Collaborative Planning and Event Monitoring Over Supply Chain Network

Sujoy Ray

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

March 2017

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Sujoy Ray**

Entitled: **Collaborative Planning and Event Monitoring Over Supply Chain Network**

and submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Pouya Valizadeh	Chair
Dr. Patrick Hung	External Examiner
Dr. Anjali Agarwal	External to Program
Dr. Todd Eavis	Examiner
Dr. Joey Paquet	Examiner
Dr. Mourad Debbabi	Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

Dean of Faculty

ABSTRACT

Collaborative Planning and Event Monitoring Over Supply Chain Network

Sujoy Ray, Ph.D.

Concordia University, 2017

The shifting paradigm of supply chain management is manifesting increasing reliance on automated collaborative planning and event monitoring through information-bounded interaction across organizations. An end-to-end support for the course of actions is turning vital in faster incident response and proactive decision making. Many current platforms exhibit limitations to handle supply chain planning and monitoring in decentralized setting where participants may divide their responsibilities and share computational load of the solution generation. In this thesis, we investigate modeling and solution generation techniques for shared commodity delivery planning and event monitoring problems in a collaborative setting. In particular, we first elaborate a new model of Multi-Depot Vehicle Routing Problem (MDVRP) to jointly serve customer demands using multiple vehicles followed by a heuristic technique to search near-optimal solutions for such problem instances. Secondly, we propose two distributed mechanisms, namely: Passive Learning and Active Negotiation, to find near-optimal MDVRP solutions while executing the heuristic algorithm at the participant's side. Thirdly, we illustrate a collaboration mechanism to cost-effectively deploy execution monitors over supply chain network in order to collect in-field plan execution data. Finally, we describe a distributed approach to collaboratively monitor associations among recent events from an incoming stream of plan execution data. Experimental results over known datasets demonstrate the efficiency of the approaches to handle medium and large problem instances. The work has also produced considerable knowledge on the collaborative transportation planning and execution event monitoring.

DEDICATION

To all my teachers, who inspired, guided and supported me throughout my education.

ACKNOWLEDGEMENTS

Support of several people helped me to conduct this research endeavor, to only some of whom, it is possible to give particular mention here. At first, I would like to express my deepest gratitude to my supervisor Dr. Mourad Debbabi. It would not have been possible to complete this research work without his guidance, motivation, support and patience. I am grateful to all the committee members of my dissertation: Dr. Patrick Hung, Dr. Anjali Agarwal, Dr. Todd Eavis and Dr. Joey Paquet for their insightful comments and advice. I would also like to thank Dr. Anjali Awasthi and Dr. Benjamin Fung for their suggestions and feedback during the preparation of this thesis.

This thesis would not have been finished without my co-authors and project partners. I express my sincere thanks to my colleagues for their friendship and encouragement. Special thanks to my team members: Andrei Soeanu, Aref Mourtada, Badr Afify and Dr. Wen Ming Liu for their collaboration, feedback and assistance for the implementation and experiments. I am indebted to the Department of Computer Science and Software Engineering and Concordia Institute for Information Systems Engineering for providing a great facility to conduct this research. Research work for this thesis was partially funded by the Natural Sciences and Engineering Research Council of Canada, Defence Research & Development, Canada in partnership with MDA Corporation. I also sincerely thank Concordia University and the Fonds de recherche du Québec – Nature et technologies (FRQNT) for their financial support to conduct my research endeavor.

In all these years, I was fortunate to work with several renowned researchers across the country. I would like to extend my gratitude to Dr. Mohamad Allouche, Dr. Abdeslem Boukhtouta, Jean Berger, Dr. Micheline Bélanger and Dr. Nicolas Léchevin from Defence Research and Development Canada for their help and everything they have taught me.

Looking back, I really cannot thank enough my wife Swagata who has been a constant source of unwavering love and support. Lastly, and most of all, I would like to acknowledge the infinite care and affection of my parents, family and friends. Without their encouragement and sacrifices, I would never have made it this far.

Sujoy Ray

March 13, 2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

The following notations and abbreviations are frequently used in various chapters of this thesis. The meaning of a notation remains same in all chapters unless otherwise stated.

Chapter 1:	
a-RFID	Active Radio-Frequency Identification
CVRP	Capacitated Vehicle Routing Problem
MDSDVRP	Multi-Depot Split-Delivery Vehicle Routing Problem
SCN	Supply Chain Network
SDVRP	Split-Delivery Vehicle Routing Problem
TMS	Transportation Management System
VRP	Vehicle Routing Problem
Chapter 2:	
COP	Combinatorial Optimization Problem
FIS	Frequent Itemsets
FLP	Facility Location Problem
ILP	Integer Linear Programming
LRP	Location Routing Problem
NP	Non-deterministic Polynomial-time
p-MP	p-Median Problem
WSN	Wireless Sensor Network
Chapter 3:	
c_{ij}	Cost of traversal along edge $\langle i, j \rangle$ of the transport network
d_i	Customer demand for single type of commodity (d_i)
D	Set of all depots
E	Set of directed edges; each edge $\langle i, j \rangle \in E$ denotes a link between source vertex i and destination vertex j
G	A complete directed graph representing an SCN; $G \coloneqq \langle V, E \rangle$
K	Set of vehicles
N	Set of all customer nodes
V	Set of all vertices including customer nodes and depots
Chapter 4:	
ANDR	Average Node Distance (Rounded)
DSE	Dominant Strategy Equilibrium
K_p	Set of vehicles under the control of participant p
MCDR	Minimum Clustering Distance (Rounded)
π_{ip}	Failure risk of participant p in monitor selection on vertex i
P	Set of participating decision makers; each member is denoted as $p \in P$
R	Total risk of failure in distributed monitor selection
VCG	Vickery-Clarke-Groves mechanism
W_p^t	A template of weight vectors where each weight vector represents the interest of a participant decision maker p to serve all customer nodes at iteration t

Chapter 5:	
c_i	Monitor deployment cost at vertex i
C	Total budget; C_p denotes deployment budget of participant p
V	Set of all vertices including relay nodes and monitors
S	A set of monitors; $S \subseteq V$
Q	A bipartite graph; $Q \coloneqq \langle V \setminus S, S, E' \rangle$ where $V \setminus S$, S and E' are relay nodes, monitors and a set of directed edges respectively; Each member $\langle i, j \rangle \in E'$ represents a communication link from relay node i to a monitor j
w_{ij}	Total number of traversal (as planned) on a directed edge $\langle i, j \rangle$
δ_{ij}	Energy consumed by i to send data to j for each traversal
γ_{ij}	Energy consumed by i to send data to j for all traversals from vertex i
Chapter 6:	
$\mathcal{A}$	Alphabet; a set of all possible items
$\mathcal{A}_p$	A subset of events/items in $\mathcal{A}$ indicating concern of participant p
ARM	Association Rule Mining
CET	Closed Enumeration Tree
ς	The value of current confidence of an association rule
c_{min}	Minimum confidence threshold
FP-Tree	Frequent Pattern Tree
MAREDS	Mining Association Rules over Event Data Stream
MCR	Maximum Confidence Rule
MFI	Maximum Frequent Itemset
PAET	Partial Association Enumeration Tree
s_{min}	Minimum support threshold
USP	Update Set Pairs

Chapter 1

Introduction

The huge technological advancement in the last few decades has brought new challenges to the conventional operations for businesses ranging from private enterprises to governmental institutions. Rapid economic changes, business expansion, infrastructure improvement and faster data sharing techniques have placed high demand for collaboration in solution generation and monitoring of business plans to efficiently execute large-scale and long-lasting operations. In general, an operation requires a complex process, constrained by dependencies and priorities, that uses allocated resources through a set of tasks in order to achieve its goal(s). Many academic and industrial efforts can be found in research literature on various aspects of automation from generating plans to monitoring execution of various operations. In this regard, this dissertation explores automated collaborative solution generation and monitoring techniques for operational plans in presence of multiple participants. Specific focus of this study is on commodity delivery plans over Supply Chain Network (SCN).

Large organizations address commodity delivery planning and monitoring at three different levels over SCN, namely: strategic, operational and tactical [179]. At the strategic level, decision makers select depots for serving the customers. Various network partitioning/clustering algorithms are popularly used to choose depot locations at the vicinity of the

customers. Once the depots are identified, operational decision makers solve the underlying routing problems as per the requirements. These sub-problems are often characterized as Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), Traveling Repairman Problem (TRP), etc. Analytic, heuristic and meta-heuristic algorithms are used to solve the routing problems typically with an objective to minimize the routing cost. The resulting solution is a set of tasks where each task forms a vehicle route. The tactical officers execute these tasks in compliance with previously taken decisions. Timely planning for logistic deployment is vital in situations like humanitarian aid, disaster relief and rescue operations or national crises. For example, after a disaster, such as earthquake, multiple disaster relief organizations (e.g. United Nations Disaster Assessment and Coordination (UNDAC), Red Cross Society, etc.) launch major rescue operations to help potential victims. While their management teams remotely decide over establishing depots, the rescue commanders decide on the routing plans and the drivers are asked to reach out the victims. Depending on specific needs, objectives of these tasks can be different. However, such an objective always represents optimization of a quantifiable (e.g. routing cost, rescue time, etc.) subjected to a set of constraints. In government and commercial organizations, commodity delivery planning significantly save cost of operations.

SCN monitoring is also a challenging problem to the decision makers. Often time, lack of sufficient tracking capability leads to missing crucial assets [32]. Holguin-Veras et al. [235] has described the importance and difficulties of establishing communication within a short duration during hurricane Katrina. A similar experience is documented by Schwartz et al. [202] on the aftermath of Haiti earthquake where multiple organizations attempted delivering necessary commodities and services over a large transport network. In this respect, efficient monitor placement and information tracking from pre-installed sensors may notably enhance situation awareness over SCN by improving tracking of movement, speed, efficiency and cost for executing tasks [72, 99].

At a larger scale, efficient plan modeling, solution generation and robust task monitoring require collaborative decision making framework and a vigilant advisory system as discussed by Allen *et al.* [11]. However, collaborations of participants are often restricted by the organizational policies over data sharing. It also demands for quick decision making

by analytical computation over collected information which is often approximately accurate. In this respect, the available off-the-shelf software solutions exhibit notable gaps in the following important areas:

- Optimal partitioning of SCN to share delivery of logistics/commodities;
- Modeling collaborative planning and delivery of logistics/commodities;
- Offering distributed algorithms for transportation problem solving;
- Elaborating collaborative approaches for plan execution monitoring.

Within the scope of this thesis, we intend to bridge these research gaps.

Large-scale multi-level commodity delivery planning has a major shortcoming. It stems from the initial selection of locations for depots at the strategic level where decisions are made without accurate knowledge of actual routing cost. Salhi and Rand have shown that the best solution for depot locations found at routing stage contrasts the results obtained from three different location-allocation methods [196]. To overcome this limitation, in this thesis, we investigate depot assignment and logistics delivery problems together in the commodity delivery planning. In Chapter 3, we introduce a new linear model of the combined problem and propose a generic solution search technique for a multi-depot vehicle routing problem that may also employ split delivery of commodities, if needed. The solutions of this problem are expected to efficiently use limited number of vehicles with predefined capacity to serve customers demands from various depots. However, the combination of these two sub-problems, namely depot selection and vehicle routing, significantly increases the problem complexity. Thus, it restricts researchers to handle large problem instances with many depots and customers centrally with current computing capabilities. Therefore, in Chapter 4, we study two collaboration mechanisms such that multiple participants can jointly generate near-optimal solutions of multi-depot vehicle routing problem.

Similarly, in order to address plan execution monitoring, we first investigate collaborative technique to collect sensor data from SCN. Timely relayed execution information may provide valuable input for decision making [29]. Additionaly, now-a-days, inexpensive a-RFID tags are being frequently used with additional sensors to communicate data to

RFID readers over SCN [160]. These readers also transmit such information to distant decision makers using in-field monitoring facilities. Fast and collaborative determination of data relaying strategy is crucial for quick deployment of these technologies in operations, such as disaster support, rescue missions, etc. [239]. However, remote RFID readers are often constrained from on-board power supply and much of their energy is used to communicate with the monitors. Furthermore, monitor deployment incurs a significant cost over a large SCN which puts an additional constraint for limited deployment of monitors within a deployment budget. Thus, obtaining an optimal strategy to minimize energy consumption in such data communication is an open research problem with applications over diverse networks, (e.g. wireless sensor networks (WSN), smart-grid, etc. [105, 249]). Moreover, in a collaborative setup, participants also share the deployment budget [72]. In Chapter 5, we investigate centralized and distributed approaches for optimal monitor deployment where the deployment budget is split among collaborative participants. Finally, in Chapter 6, we discuss collaborative monitoring of sensor generated data stream to reveal frequent associations among recent events. Such association is often crucial to the remote decision makers. For example, it is useful to analyze a recent delay event over SCN in connection to other events such as weather, accidents, vehicle failure, etc. In this regard, a major challenge lies in quickly inferring relationships among related events by mining incoming data stream(s) generated by remote in-field sensors. Typical event mining techniques heavily depend on batch processing of previous records. Batch processing needs availability of adjusted information along with significant computation time and memory. In contrast, stream mining reveals association among recent events relevant to current situation.

1.1 Motivation

In commercial sectors, trade related surface transportation is constantly increasing in North America. In the past, between 2009 and 2010, the transportation and warehousing sectors were growing approximately by 4.3% [110]. In Canada, the Gross Domestic Product (GDP) in the transportation and warehousing sectors was increased significantly in last decade from $50.2 billion in 2001 to $58.4 billion in 2010 [110]. According to the United States

Department of Transportation, the surface transportation trade between North American Free Trade Agreement (NAFTA) partners was increased by 11.5% in January 2012 compared to January 2011 at $75.5 billion [214]. The value of NAFTA freight turned $88.2 billion during November 2015 in which 51% ($45.1 billion) trade happened between the US and Canada[1]. Interestingly, almost 60.4% of this trade was carried out by large trucks. Thus, intelligent transportation planning and monitoring software hold the key for the cost saving in such trade [72]. Meanwhile, worldwide Transport Management System (TMS) software revenue also increased 20.6% from 2007 to 2008 ($538 million to $648 million) and reached to $963 million at 2012 globally [72]. Another recent Gartner report [123] forecasts that such technology-led evolution in supply chain convergence and process orchestration will continue unabated where decision makers will increasingly use mature and proven routing solutions and scheduling tools. According to their claim 65% of its respondents view supply chain management technologies as a source of competitive advantages.

The process orchestration in TMS is also quickly changing with the new generation of data communication technologies and fusion of sensor generated information. Timely relayed execution information is providing more valuable input for decision making than ever before [29]. Beside installation of sensors over SCN, more advanced decision-support methods and tools are expected to trigger innovations in real-time information gathering, changes in fleet management, tracking and re-routing of assets based on external changes (such as weather, traffic, etc.) [123]. Gartner predicts high benefits over such revolutionary changes in supply chain management and expects newer fleet routing and scheduling applications in recent future [123]. Alongside, fast and collaborative monitoring of relayed information is increasingly depending on sensor (RFID) technologies in operations, such as disaster support, rescue missions, etc. [239]. Efficient monitoring of RFID information reduces supply chain cost and operation time up to 2.8%-4.5% [197] and 57.18% [28] respectively. Quicker decision making using RFID technology may reduce shipping lead time for a single product up to 66.12%. In an experiment, WalmartâĂŹs pilot RFID implementation to replace its bar code system reduced its CO_2 emissions by 3.2% [2]. RFID

[1] https://www.scranet.org/SCRA/News_Release/Newsletter/Industry_News/Canada_regains_rank_as_top_North_America_trading_partner_with_US.aspx

[2] http://journals.isss.org/index.php/proceedings51st/article/viewFile/493/248

implementation has shortened number of trips for trucks and offered better product visibility and tractability with improved inventory management [197]. Efficient RFID sensors, better monitor placement and fast monitoring may reduce order quantity and inventory level at distribution centers up to 47% [197]. On the other hand, RFID-based information tracking is less labor intensive and represents only 0.5% of the product value [28]. Hence, in the light of the aforementioned, collaborative planning and monitoring unfold important research problems with significant business impact.

1.2 Problem Overview

In collaboration, participants work together towards a shared objective. In order to attain the objective, participants may willingly execute different workload without the evaluation of their personal interest. In this respect, collaboration differs from the concept of cooperation where self-interested participants share the workload to reach a common goal. In cooperation, participants perform together until their common goal can be achieved by mutually benefiting each party. Thus, it helps reaching a shared goal as long as individual members may pursue their own sub-goals. In what follows, we introduce three collaborative sub-problems in relation to this thesis, namely, collaborative vehicle routing, collaborative monitor deployment and collaborative plan execution monitoring.

1.2.1 Collaborative Vehicle Routing

Collaborative vehicle routing handles commodity delivery to customers (demand points). The customers can be represented as nodes in a complete graph where the graph denotes a transport network. The latter is a type of SCN where transportation activities take place (e.g. commodity delivery). Given a set of nodes (V) and a set of directed edges (E), where E is a relation in $V \times V$, a transport network is a complete graph G = (V, E). Each edge of the graph provides a traversal cost (c_{ij}) from source node i to destination node j. Usually, a transport network is composed of two different node types: Customers (N) and Depots (D). While customer nodes are characterized with deterministic demand (integer) for commodity (d_i), depot nodes have no demand. However, they host vehicles

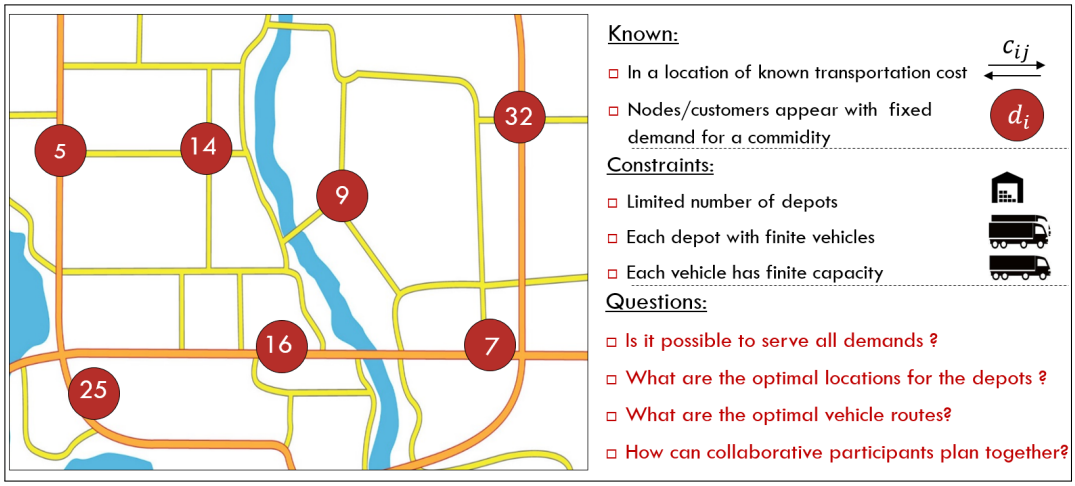


Figure 1.1: A transportation network for collaborative vehicle routing problem

$(k = 1, 2, \ldots, K)$ to supply commodities to customers. Thus with predefined depots and customers, a solution for a collaborative routing problem instance gives the routes for each vehicle that minimizes the overall routing cost to serve all customer demands. Figure 1.1 depicts a small commodity delivery problem where each node represents a geographic location as identified through a red circle. These locations are connected to every other nodes through a sequence of roads. We may abstract such connection with two directed edges between each pair of nodes. The number inside the red circle denotes an integer demand (d_i) of a commodity for the node i. Additionally, we assume an establishment cost (EC_i) to set up a depot at node i. As presented in Figure 1.1, there are three main constraints to the collaborative vehicle routing:

- *Limited Depots*: Only a fixed number of depots can be established. Each node hosting a depot will not input any demand assuming that the respective node can be directly served by the host depot without any need for vehicle routing.
- *Limited Vehicles*: There exists a fixed number of vehicles $(k = 1, 2, \ldots, K)$ to serve all customer demands. There can be a predefined number of vehicles associated per depot or they can be considered as total K vehicles that can be used from all depots.
- *Vehicle Capacity*: Each vehicle has a limited integer capacity to load commodity.

In this setting, route planning evaluates feasibility of serving all demands. An optimal solution offers best locations for the depots (if not predefined) and optimal vehicle routes with minimum cost of routing. Assuming, each newly found depot serves its own demand, the problem aims at minimizing the combined depot establishment and routing cost. We formulate this problem as Multi-Depot Split-Delivery Vehicle Routing Problem (MDSDVRP). Unlike VRP, where a node is served by one vehicle, MDSDVRP allows multiple vehicles to deliver commodity to a customer which may lower the total routing cost. Chapter 3 discusses MDSDVRP solution generation technique. Chapter 4 covers the collaboration aspect of such solution generation in presence of multiple participants planning together.

1.2.2 Collaborative Monitor Deployment

Figure 1.2 depicts a screenshot of a transport network for the aforementioned SCN where two organizations operate over a product delivery area from two depots. Each delivery is performed by agents (e.g. vehicles) who visit different vertices in sequence through connecting paths (e.g. roads). The agents are assumed to have appropriate equipment to communicate position, status, etc. to nearby relay nodes (Ⓡ) through sensors. Execution monitors (Ⓜ) deploy additional devices to collect and process relayed data and deliver information to distant decision makers. Each communication from relay-node i to monitor j consumes energy δ_{ij} from on-board power source of the relay node. More agents pass through a relay node on the their way of routing node's energy consumption turns higher. On the other hand, monitor deployment incurs setup and security cost. Limited budget puts constraint on the number of deployed monitoring facilities (or monitors). Thus, the core optimization problem of the monitor deployment aims at finding a set of optimal monitor deployment locations that minimizes weighted average energy consumption for data communicated from relay nodes to monitors under a budget.

In collaborative monitor deployment, we extend the aforementioned problem such that the participants may plan with individual monitor deployment budget. They can jointly search optimal monitor locations by locally executing distributed optimization techniques. Furthermore, in this case, they do not explicitly disclose individual budget to others but negotiate to a globally optimal solution which can be potentially achieved by combining

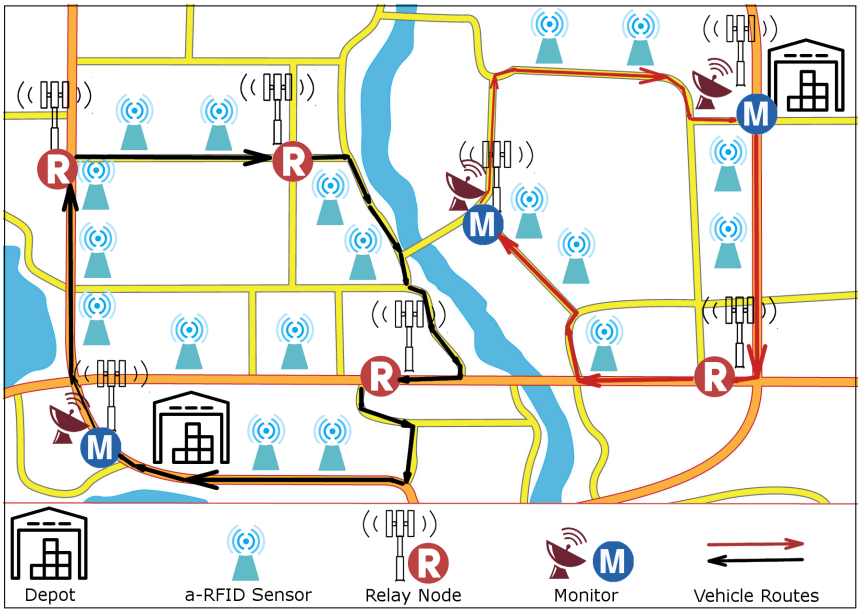


Figure 1.2: A transport network for collaborative monitor deployment

their budget. Chapter 5 describes our proposed solution finding approach.

1.2.3 Collaborative Plan Execution Monitoring

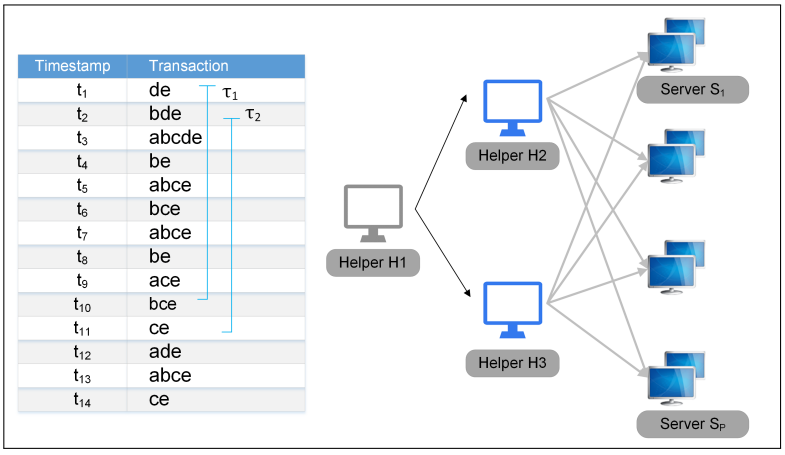


Figure 1.3: A system architecture for collaborative plan execution monitoring

Fig. 1.3 depicts a sequence of 14 transactions from an incoming data stream at a data center where each transaction represents a set of events associated to a timestamp.

For example, transaction "abcde" represents a set of five remote events {a,b,c,d,e} occurring at timestamp t_3. In a centralized setting, one server is required to investigate the relationships among these events. In data mining, these relationships are often reflected through association rules where antecedent and consequent of every rule indicates cause and effect respectively. In Chapter 6, we propose an efficient incremental association rule mining approach in order to progressively extract exact set of desired association rules from an incoming stream over a sliding window model. Furthermore, as depicted in Figure 1.3, we extend this approach to collaboratively extract these rules in multiple subsets using a number of servers. These servers are controlled by participating decision makers, each of whom has interest in a subset of all possible events. The collaboration, in turn, reduces the stream processing load and allows handling larger number of events at every update of the sliding window. Such a setting also requires a small number of entities, namely Helpers, to schedule the collection and mining of association rules in distributed setting.

1.3 Objectives

The main goal of this thesis is to contribute to a collaborative decision support framework in the area of commodity delivery planning and plan execution monitoring over SCN. In this respect, we investigate the aforementioned three sub-problems regarding collaborative route planning, monitor deployment and plan execution monitoring. Therefore, the objectives of this thesis can be summarized as follows:

- Modeling multi-depot split-delivery vehicle routing problem and developing algorithms to produce near-optimal solutions for related problem instances;
- Investigating collaborative solution technique for multi-depot vehicle routing problems and contrasting the approach against cooperative solution generation technique;
- Modeling collaborative monitor deployment problem and elaborating a distributed solution generation technique for the related problem instances;
- Designing and developing techniques for plan execution monitoring using stream mining in centralized and distributed settings;

- Conducting case studies, experiments and performing comparative studies to evaluate our solution approaches against existing techniques available in the literature.

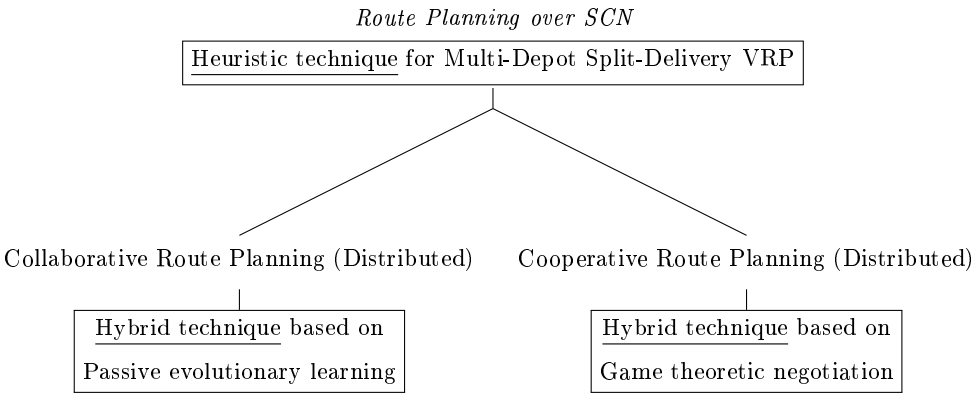
Thus, our approach is expected to bridge notable gaps of collaborative solution generation approaches among the currently available transport management systems.

1.4 Contributions

We investigate collaborative solution generation approaches for three optimization problems on route planning, monitor deployment and plan execution monitoring. We present mathematical models of the underlying research problems and develop algorithms to find near-optimal solutions. Following, we illustrate the technical contributions of this thesis:

- *Collaborative Vehicle Routing*: Our core contribution on this research problem includes the elaboration of a mathematical model for multi-depot, multi-vehicle per depot vehicle routing with split delivery. In this respect, we propose a flexible model using Integer Linear Programming (ILP). The flexibility allows customizing the model via small modifications (according to the need) to address a number of specific problems of the VRP family such as Multi-Depot Vehicle Routing Problem (MDVRP), Split-Depot Vehicle Routing Problem (SDVRP), Capacitated Vehicle Routing Problem (CVRP), etc. Moreover, the concept of location routing brings forward both location allocation and vehicle routing as part of the same objective function. It helps customizing the depot establishment cost, as well as, prior establishment of depots at specific locations. With respect to the related heuristics algorithm, it allows generating vehicle routes with near-optimal cost while serving the customers by multiple vehicles belonging to the same or different depots. However, the common solution approaches for multi-depot VRP assume a centralized setup with complete knowledge of travel cost, locations of the depots, vehicles and customers. In this thesis, we investigate collaborative route planning for multi-depot VRP problems using two main concepts result sharing and sub-problem sharing. The

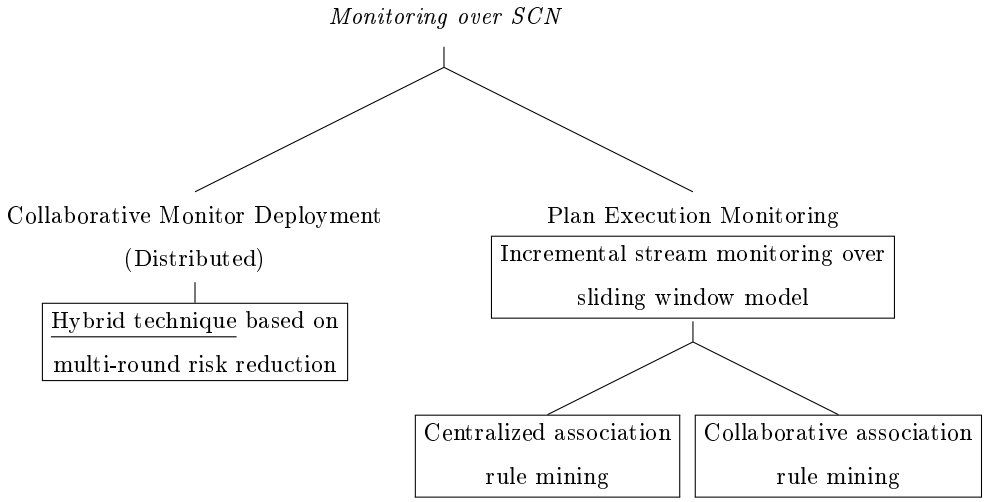
benchmark results show clear benefits in deriving routes using the proposed collaborative approaches. The following diagram summarizes our contributions for collaborative route planning where the rectangular boxes indicate our proposed techniques.



- *Collaborative Monitor Deployment*: Our contribution in this research problem is focused on collaboratively deploying monitors over a subset of relay nodes on an SCN to minimize energy consumption in information communication. Considering individual deployment budget, we propose a mathematical model for multiple decision makers of monitor deployment. Hereof, we first present a centralized heuristic algorithm to compute solutions under a common budget constraint. Second, we develop a distributed approach to automate a collaborative negotiation mechanism for near-optimal monitor deployment with individual budgets. The optimal location allocation of monitors over SCN where the total budget is divided among participants, is a distributed variant of classical facility location [75] and p-median problems [141]. Benchmark results show high accuracy and performance benefits of the proposed collaboration approach particularly through efficiently handling large problems.

- *Collaborative Plan Execution Monitoring*: Our contribution in this research effort is focused on an incremental association rule mining solution to indicate causal relationships between events from a data stream. We design a data structure to store events and apply incremental, in-memory algorithms to update rules quickly and accurately. Primarily, we propose an efficient mechanism to extract a set of desired association rules efficiently from an incoming data stream. Then, we illustrate a collaborative distributed technique to distribute the mining process in multiple servers

of participating decision makers for efficient update of these association rules. The distributed rule generation procedure also improves scalability and restricts knowledge gathering without any loss of association rules. We highlight our contributions for collaborative monitor deployment and collaborative plan execution monitoring in the diagram below. The rectangular boxes indicate our developed techniques.



Traditional logistics planning and its subsequent execution phase(s) heavily depend on human expertise for decision making. Thus, it exhibits intrinsic limitations in handling large and complex operations. Human intervention often provides erroneous inputs and incurs significant delay in decision making. The new disruptive technologies (e.g. Internet of Things), advanced data processing and increasing computing memory may improve the planning and the tracking of plans with efficient and sufficiently automated mechanism. It can also handle intricate sharing responsibilities in commodity delivery and improve the scope of situation responses. Furthermore, in current complex business environment, handling of large-scale SCN problem in a centralized setup with complete knowledge of all input parameters, is often impractical. The underlying techniques also suffer from known scalability issues. Thus, distributed solution generation for optimization problems helps in faster, efficient and collaborative operation management for elaborated business operations. However, distributing planning and monitoring problems over SCN is challenging. In this thesis, we attempt to solve three core research problems in collaborative distributed setting.

1.5 Organization

The remainder of the thesis is organized as follows:

- Chapter 2 presents an overview of the related work on three aforementioned collaboration problems. We describe the context of these optimization problems with detailed discussion in the light of the existing research literature.

- Chapter 3 discusses MDSDVRP. We present an Integer Linear Programming (ILP) model and illustrate a heuristic based algorithm to near-optimally solve MDSDVRP instances. We provide extensive benchmark results on a large number of existing problem instances. The results of this chapter appears in [193].

- Chapter 4 proposes and evaluates two distributed approaches to solve multi-depot VRP instances. We describe how such problems can be distributed among collaborative decision makers and solved using aforementioned heuristics algorithm locally. We compare the benchmark results with our solutions in a centralized setup. The results of this chapter appears in [195] and [209].

- Chapter 5 elaborates a collaborative multi-round risk reduction approach to solve a budget constrained monitor deployment problem that reduces energy consumption between the relay nodes and monitors during data communication. We propose a distributed model and a collaborative strategy to near-optimally solve monitor locations when the budget is split among participants. We present certain benchmark results on known problem instances. The results of this chapter appears in [192].

- Chapter 6 elaborates implementation of a stream mining technique that extracts association rules among the generic events from a data stream. We have demonstrated how such mining can be performed collaboratively using multiple servers. Our Experimental results clearly indicate that the proposed technique can be efficiently executed to monitor a large stream of events. A generic and scalable monitoring technique may also benefit several monitoring applications from diverse areas.

- Finally, Chapter 7 presents some concluding remarks. In this chapter, we recall and summarize the research results and discuss possible future work.

Chapter 2

Background and Related Work

Collaborative planning and monitoring of supply chain activities allow sharing of responsibilities for delivering commodities in a timely manner at the right locations. The computation complexity and practical relevance of these planning and monitoring problems have attracted many researchers for more than half a century. In this chapter, we survey and analyze previous research and development efforts to identify major collaboration challenges over two supply chain activities, namely, vehicle route planning and plan execution information monitoring. We begin with a brief overview of the related classical research problems, their complexities and various solution generation approaches. We also highlight limitations of these classical problems and solution techniques in order to achieve collaboration in aforementioned supply chain activities.

2.1 Context

As the supply chain operations are continuously being affected by business expansion and budget challenges, collaborative platforms are turning vital to the national and international interest in order to share resources (e.g. vehicles) for task execution. Such a platform helps to increase the use of common resources and sharing of responsibilities in commodity delivery by effective route planning and joint execution status monitoring [120, 147]. This, in turn, may lower the cost of operations and improve efficiency. However, planning and

monitoring over SCN involve solving combinatorial optimization problems where optimal solution generation is often intractable for large problem instances. In what follows, we first present a general overview concerning the complexity of such problems.

2.1.1 Optimization Problems

The following terms will be frequently used in this thesis. We begin by defining them in connection to previous research efforts.

Definition 1. *Optimization Problem [230]: An instance $I = (S, f)$ of an optimization problem specifies the following setting:*

- *A set of feasible solutions S for I, and*
- *A cost function $f : S \to \mathbb{R}$, where $\mathbb{R}$ represents the set of real numbers.*

In this setup, optimization can be achieved either by minimization or maximization.

- *Solution s^* is optimal for a minimization problem instance I, iff $f(s^*) \leq f(s), \forall s \in S$.*
- *Solution s^* is optimal for a maximization problem instance I, iff $f(s^*) \geq f(s), \forall s \in S$.*

Usually, S is defined over a solution space U. In applied mathematics, a solution $s \in S$ is generated using a set of decision variables that are subjected to a number of constraints. If the variables are continuous, we categorize these problems as continuous optimization problem. In contrast, if all or a subset of these variables are discrete, the category of the optimization problem is called *discrete* [169]. In this dissertation, we use only discrete binary and integer variables. Consequently, the main focus of this study is on a specific subset of combinatorial discrete optimization problems.

In this context, the solution search space (U) is created from a finite ground set E. Thus, in combinatorial optimization, $S \subseteq 2^E$ where 2^E defines all combinations (the set of all subsets) of E. If $|E|$ is large, enumerating all combinations is intractable. The phenomenon is called *combinatorial explosion*. Thus, U and S are always implicitly defined within the description of a combinatorial optimization problem.

Definition 2. *Combinatorial Optimization Problem [213]: An instance I of a combinatorial optimization problem (COP) is defined as a tuple $\langle U, P, f, o\rangle$ where:*

- *U is the solution space on which S and f are defined.*
- *P is the feasibility predicate, i.e. for any solution $s \in S$, $s \in U$ and s satisfies P.*
- *$f : U \to \mathbb{R}$.*
- *o is optimization (extremum) goal usually defined as minimization or maximization.*

Combinatorial optimization deals with a set of predicates (often represented through equations and/or inequations), whereby a number of feasible solutions (S) in U are delimited. S is discrete or can be reduced to discrete. Thus, the optimization determines a global optimal solution $s^* \in S$ where s^* satisfies P and $f(s^*)$ meets the optimization objective. It is also worthy to note that each COP includes a corresponding decision problem where the function $f(\dots)$ only evaluates feasibility of the solution. In a canonical form, COP can also be modeled as follows:

$$\text{minimize } f(x) \tag{2.1}$$

$$\text{Subject to: } g_i(x) \leq 0 \text{ for each } i \in \{1, \dots, m\} \tag{2.2}$$

$$h_j(x) = 0 \text{ for each } j \in \{1, \dots, p\} \tag{2.3}$$

$$x \in X. \tag{2.4}$$

In this model, the requirements (predicates) are expressed (programmed) through a number of equations and inequations (see Eq.s (2.2) and (2.3)). Given a set of variables $x_i \in X$ and their corresponding coefficients $a_i \in \mathbb{R}$, if $f(x)$, $g_i(x)$ and $h_j(x)$ can be represented as $\sum_i a_i x_i + b$ then it suggests that the requirements can be expressed in linear relationship and S is bounded by a polyhedron. In mathematical programming, such a programming method is called linear programming. Otherwise, the programming method is termed as non-linear. In this thesis, we mainly deal with linear programming to model our problems. Also, we consider discrete variables i.e. X represents binary ($\{0, 1\}$) or integer ($\mathbb{I}$) numbers. The corresponding modeling technique is known as Integer Linear Programming (ILP).

In order to design solution algorithm for any COP instance, we first need to ascertain the characteristics of the solution set S. In this respect, we study the convexity of the polyhedron that S is bounded with (in case of a linear programming model).

Definition 3. *Convex Optimization [31]: An instance of an optimization problem is convex if a convex objective function minimizes over a convex set of solutions (S) within a solution space where:*

- *All constraint functions (e.g. g_i in Eq. (2.2) and h_j in Eq. (2.3)) are convex, and*
- *For any two solutions $s, s' \in S$, $f(\alpha s+(1-\alpha)s') \leq \alpha f(s)+(1-\alpha)f(s')$ where $\alpha \in [0,1]$,*

The first condition ensures that S is defined as a bounded space in U. Therefore, if we consider s, s' are two points in S, the convex set guarantees that a solution represented by $\alpha s + (1-\alpha)s'$ also belongs to S. In other words, all the points in a straight line between any two elements of S also belongs to S. Since the objective function f ($f : U \rightarrow \mathbb{R}$) maps every member of U to the co-domain of a real-valued function, it is important that f is also continuously defined for all solutions represented by $\alpha s + (1-\alpha)s'$. Thus, it ensures the existence of a solution. Furthermore, f is convex as identified from the second condition. It helps to find a global minimum value by comparing local minimum solutions over different subset of solutions while covering the search space. This guarantees that the use of distributed/collaborative methods can solve certain types of convex optimization problems. It also allows defining a set of necessary and sufficient conditions to explore the solution search space [31]. In general, f can be linear, quadratic or even exponential function. However, we only address f as a linear function as a part of an ILP. Thus, all our discussed optimization problems are convex.

The computation complexity of any mathematical problem is always compared through the mathematical computation model of an abstract Turing machine[1] . A large number of discrete COP can be solved and verified in polynomial time using a deterministic Turing machine [201]. From the perspective of computation complexity theory, these are classified as $\mathcal{P}$. Alternatively, there exists another subclass of discrete COP, namely $\mathcal{NP}$, where only the verification of the solution of its members can be performed in polynomial

[1] https://courses.engr.illinois.edu/cs498374/notes/38-nondet-tms.pdf

time using a deterministic or non-deterministic Turing machine. In Mathematics, relation between $\mathcal{P}$ and $\mathcal{NP}$ is a major unsolved problem. We assume $\mathcal{P} \neq \mathcal{NP}$ in similarity to previous research efforts. Intuitively, $\mathcal{NP}$ is a set of decision problems. Beyond $\mathcal{NP}$, there exists another fundamental class of problems that can be at least as hard as $\mathcal{NP}$ to solve or even harder. It is not necessary that their solutions can be verified in polynomial time even using a non-deterministic Turing machine. These are termed as $\mathcal{NP}$-Hard problems. Among these $\mathcal{NP}$-Hard problems, if a problem is computationally $\mathcal{NP}$-Hard for solution generation but their solutions can be verified in polynomial time, this sub-class of $\mathcal{NP}$-Hard problems is known to be $\mathcal{NP}$-Complete. In this thesis, we deal with the discrete COPs where the computation complexity of their solution generation is generally $\mathcal{NP}$-Hard. In order to prove such claim, we relate each of our problems to a known COP where computation complexity of solution generation is $\mathcal{NP}$-Hard. Majority of the supply chain planning and monitoring optimization functions are typically associated to the following:

- *Cost Minimization*: Cost reduction is a common objective for a majority of decision problems [102, 103].
- *Delay Minimization*: Faster deployment, rapid sustenance and quick stock replenishment require delay minimization [147].
- *Risk Minimization*: Risk minimization is often considered vital while transporting supplies in a hostile environment [208].
- *Communication Minimization*: Communication minimization in planning is important in remote or hostile territories [175] and for sensor networks. For the latter, communication minimization offers longer life of the sensors and relay nodes [22].
- *Computation Minimization*: Execution management systems with elaborated monitoring components require time-bound response generation, therefore monitoring can be presented as an optimization problem under time constraints.
- *Energy Consumption Minimization*: Minimization in energy consumption is important for the endurance of the sensor network, RFID technologies where the deployed in-field instruments have limited source of energy [27].

- *Path-Coverage Maximization*: Supply chain management and surveillance problems also seek to maximize path-coverage to achieve maximum coverage within the limitation of constrained resources (e.g. fuel).
- *Accuracy Maximization*: Action planning requires accuracy maximization when conducting supply chain operations within specific time-windows.

In practice, optimization functions are often designed with multiple criteria of different weight or importance according to the preference(s) of the decision maker(s). Moreover, optimization problems over SCN are subjected to a set of typical constraints such as:

- *Budget Constraint*: Budget is one of the most common constraints for optimization problems. It is usually represented through monetary cost, energy, etc.
- *Resource Constraint*: Planning and execution monitoring take place based on limited resources in terms of assets, sensors, etc.
- *Demand Constraint*: Supply chain planning should respect demand fulfillment of the customer needs. This often puts restrictions on mission planning.
- *Capacity Constraint*: Plans are executed via agents (e.g. transport vehicles, personnel, etc.) which have limited carrying capacity.
- *Temporal Constraint*: Decision making and task execution are often required to take place within a corresponding time bound.
- *Spatial Constraint*: The task execution is also constrained from geographic features (e.g. rivers, lakes, etc.) and spatially dispersed hazards.
- *Communication Constraint*: Supply chain planning may require radio silence to prevent detection and conserve energy. In sensor networks, sensor-communications are typically limited in a range.

In this thesis, we first discuss route planning. The focus is on cost optimization over commodity delivery to a set of customers by a set of vehicles, from one or many depots over a transport network. The transport network is characterized by a graph where each

node represents a customer or a depot and a directed edge between any two nodes denote a traversal path between them with a deterministic cost. The constraints to the route planing involve resource constraints, demand constraints and capacity constraints. Alongside, we also discuss monitor deployment problem which involves energy minimization under budget and monitoring resource constraints. In the case of execution monitoring, we further study association among co-occurring events under memory and time constraints.

2.1.2 Near-Optimal Solution

Every combinatorial optimization problem has a corresponding decision problem which evaluates if a solution is feasible given the problem constraints. A massive number of feasible solutions may exist for a large-scale instance of an optimization problem where each solution satisfies the constraints. Thus, tracing an optimal solution for such an instance is challenging. As discussed, if the optimization problem is $\mathcal{NP}$-Hard, exhaustive verification of all feasible solutions is often intractable. In such cases, approximate algorithms are used to deal with the complexity of these problems. An approximation algorithm generates solutions within a reasonable amount of time that respects all constraints of the optimization problem but often does not guarantee the quality of the solution against the true optimal solution the problem instance. In a minimization problem, the evaluation of the objective function f over the near-optimal solution s^{no} has higher value than that of the optimal solution s^*, i.e. $f(s^*) \leq f(s^{no})$; given $s^*, s^{no} \in S$. Similarly, for a maximization problem, $f(s^*) \geq f(s^{no})$; given $s^*, s^{no} \in S$. So, we define near-optimal solution of an optimization problem as follows:

Definition 4. *Near-Optimal Solution [31]: Given a predefined ratio: gap, a solution s^{no} of an optimization problem instance is near-optimal if and only if:*

- *s^{no} satisfies all constraint functions and*
- *$\frac{\|f(s^{no}) - f(s^*)\|}{f(s^{no})} \leq$ gap where f is the objective (or cost) function of the problem*

Therefore, a specific problem instance may have a range of solutions that are near-optimal. Among them, a smaller value of $\frac{\|f(s^{no}) - f(s^*)\|}{f(s^{no})}$ indicates better solution quality. Thus, it

serves as a metric to compare approximate algorithms based on solution accuracy. However, in literature, value of optimal solution is unknown for many benchmark problem instances. In this thesis, as we tackle the minimization problems, $f(s^*)$ is replaced with the best known solution value as $\frac{f(s^{no})-f(s^{bestknown})}{f(s^{no})}$ in order to compute a near-optimal solution. Correspondingly, a negative value of the fraction indicates the finding of a better solution than previously best known solution.

2.1.3 Collaborative Optimization

Collaborative solution generation for optimization problems allows participants to jointly conduct optimization procedure in the form of distributed sub-problems. Smith and Davis pointed out that the distributed problem solving differs from distributed processing [207]. A typical assumption in distributed problem solving relates to having a high degree of group coherence [70]. This depends on decision makers' willingness and capability to work together towards a common goal. In some cases, a sufficiently effective payoff scheme can also be designed for self-interested decision makers to join a collaborative effort [41, 156]. Thus, all participants help solving an original problem with a common objective using their limited resources. Distributed participants may also require to declare partial results from their shared tasks as computed using decentralized algorithms. These findings are communicated mainly to agree upon a current solution. The *task sharing* or *task passing* strategy is often discussed in four steps in the research literature [19]:

- *Task Decomposition*: This involves decomposing a large task into sub-tasks that can be tackled by or passed to different participants.
- *Task Allocation*: This involves assigning tasks/sub-tasks to appropriate participants.
- *Task Accomplishment*: Tasked participants proceed to accomplish their assignments, which may include further decomposition and sub-sub-task assignment recursively to the point that an agent can directly execute its task.
- *Result Synthesis*: Once an agent accomplishes its sub-task, it passes the result to an appropriate agent that is able to compose partial results. The recipients compose the results toward an overall solution in an iterative manner.

Therefore, distributed planning is tightly intertwined with the distributed problem solving, being both a problem in itself and the means to solve a problem [70]. Decomposition of tasks requires efficient handling of available resources. In this respect, the distributed solvers may be hosted by the participants who are often dissimilar in terms of knowledge, capability (e.g. asset), information, expertise, etc. which results into heterogeneous sub-task creation. In research literature, these constraints are termed as *group competence*. Data synchronization plays another major role in sharing necessary information among the distributed solvers [192]. In this thesis, we focus on the decomposition of specific optimization problems in order to solve them in collaborative distributed setting. It should be also noted that, due to limited knowledge, there exists a degree of stochastic uncertainty in decision making from the side of participants. An effective framework to express such uncertainty as constraints in optimization is through chance constraint [174].

Definition 5. *Chance Constraint Optimization [129, 167]: In canonical form, a chance constraint optimization can be modeled as:*

$$\textit{minimize } f(x,\xi) \tag{2.5}$$

$$\textit{Subject to: } g(x,\xi) = 0 \tag{2.6}$$

$$Pr(h(x,\xi) \geq 0) \geq pr \tag{2.7}$$

$$x \in X. \tag{2.8}$$

where f, g and h represent the objective function, equality constraints and inequality constraints respectively. $x \in X$ is a vector of decision variables and ξ denotes the vector of uncertainty (random) variables. $pr \in [0,1]$ represents the probability threshold.

Li et al. categorized chance constraint optimization problems based on optimization process, uncertainties and constraints [140]. The optimization process can be linear or non-linear while remaining in steady or dynamic state. The uncertainty can be modeled as constant or time-dependent while the constraints can be individual or joint. They are denoted using the first letter. For example, a linear optimization process at dynamic state

with constant uncertainty and joint chance constraint is referred as LDCS problem. In decentralized setting of P decision makers, if the probability over the inequality constraints can be distributed per participant, then, $Pr(h(x,\xi) \geq 0) \geq pr$ can be reformulated as $Pr(h_p(x,\xi) \geq 0) \geq pr_p;\ \forall p \in P$. In the case of individual chance constraint, each decision maker requires satisfying individual constraint(s) to a certain probability threshold. However, for a majority of collaborative optimization problems, the chance constraint is jointly defined where the probability is set over common decision making of all participants. Joint chance constraint is formulated as:

$$Pr\left(\bigwedge_{p \in P} h_p(x,\xi) \geq 0 \right) \geq pr \tag{2.9}$$

In collaborative optimization, uncertainties are generated by the participants. It is assumed that these participants perform individually without any consultation or dependence over the action(s) of others. Thus, the uncertainty variables are uncorrelated. Normal (Gaussian) distribution is often used as an adequate assumption to model such uncertainty [140]. Also, each objective function of our optimization problems (*see.* Section 1.2) is expected to be a deterministic function that minimizes a quantifier such as: routing cost, energy consumption, etc. If such an objective function is also linear, then all our collaborative optimization models optimize a deterministic convex objective $f(x)$ instead of $f(x,\xi)$ as mentioned in Eq. (2.5). Furthermore, in this thesis, we only handle linear constraints for all our optimization problems. Thus, we expect $h(x,\xi)$ as a set of randomly perturbed linear constraints which means that the decision variables and the random variables can be decoupled. Finally, we treat the joint chance constraint in terms of risk. Let R be the total risk bound of all participants to fail satisfying any optimization constraint. Then, we treat a linear joint chance constraint as $Pr(\bigwedge_{p \in P} h_p^T \cdot x_p \leq g) \geq 1 - R$. The superscript T denotes transpose of a matrix. Many researchers analyzed the value of R to be $(0 < R < 0.5]$ for the convexity of the constraints [167, 174]. Nemirovski and Shapiro described this computability of the probability and the convexity of the corresponding feasible set as a *"rare commodity"* in majority of the chance constraint optimization problems [140]. Campi and Garatti discussed feasibility and optimality of the chance-constrained optimization using

randomization [37]. In Chapters 4 and 5, we particularly follow a finite-horizon optimal control technique, as discussed by Ono and Williams [174]. We jointly optimize a dynamic chance constrained optimization problem and generate computationally tractable near-optimal solutions by iterative reduction of risk. Our interest lies in addressing convex collaboration problems. Finite-horizon optimal control technique considers two sets of decision variables, namely control variables ($\mathcal{X}$) and state variables ($\mathbb{U}$). The control variables are evaluates at every state of the system. For a finite number of iterations (τ), the technique computes values for control variables at each iteration while satisfying all state and control constraints. State variables describe the mathematical state of the dynamic chance-constrained optimization problem itself. A typical dependency of state variables is as follows:

$$u^{t+1} = A \cdot u^t + B \cdot x^t + E \cdot \omega^t \tag{2.10}$$

where, state vector u^{t+1} is updated from its previous corresponding state vector u^t. x^t is a control variable. ω^t denotes an additive disturbance at step t. A, B and E are user chosen constants. The process seeks minimizing or maximizing the control variables at each step. In this setting, collaborative decision making requires decomposition, allocation, accomplishment and synthesis of the tasks. Among these, task accomplishment requires task specific activities. In progression of this thesis, we discuss a generic procedure of task decomposition, allocation and synthesis in the view of finite-horizon optimal control [173].

- *Risk Decomposition*: First, it should be noted that, in collaborative setup of P participants, chance constraint in finite-horizon optimal control depends on the state variables. The values of the control variables are determined at every step while the mathematical state of the iterative collaboration is only held by the state variables. Based on this, we reformulate the chance constraint in the presence of N constraints and decompose the risk based on Boole's inequality i.e. $Pr(\bigcup_i A_i) \leq \sum_i Pr(A_i)$ [174]:

$$\left(Pr\Big[\bigwedge_{p\in P}\bigwedge_{i\in N} h_{ip}^{T}\cdot\mathbb{U}_i \le g_{ip}\Big] \ge 1-R\right) \le \left(\forall_{(p\in P, i\in N)}, Pr\Big[h_{ip}^{T}\cdot\mathbb{U}_i \le g_{ip}\Big] \ge 1-\pi_{ip}\right) \tag{2.11}$$

$\mathbb{U}_i = [u_i^{0T}, \ldots, u_i^{tT}, u_i^{\tau T}]$ denotes a set of variables of all state vectors for constraint $i \in N$ ranging from step $t = 0$ to $t = \tau$. T denotes transpose matrix. π_{ip} is the individual risk bound of participant p on constraint i ($\pi_{ip} \ge 0$ and $\sum_{i\in N}\sum_{p\in P} \pi_{ip} \le R$).

- *Risk Allocation*: To allocate individual risk bound for each participant, finite-horizon optimal control technique uses an inverse cumulative distribution function over an uni-variate distribution of risk, identified by $-m_{ip}(\pi_{ip})$. Ono and Williams defined this function as convex and monotonically decreasing non-negative function for $\pi_{ip} \in (0, 0.5]$ and formulated it accordingly [173, 174]. This function helps allocating the individual chance constraints of decomposed risks as presented in Eq. (2.11). It changes the individual chance constraint into inequations per participant using nominal (expected) state of the state variables, represented by $\overline{\mathbb{U}}_p = [\overline{u}_p^{0T}, \ldots, \overline{u}_p^{tT}, \overline{u}_p^{\tau T}]$. Thus, in effect, allows operating over expected state of the state variables removing the dependence of the state variables over the additive disturbance as shown in Eq. (2.12). In the collaborative setting of multiple participants and constraints, the risk allocation can be performed in multiple iterations over a finite horizon τ as follows:

$$\overline{u}_p^{t+1} = A_p\cdot\overline{u}_p^{t} + B_p\cdot x_p^{t} \tag{2.12}$$

$$x_{min}^{t} \le x_p^{t} \le x_{max}^{t} \tag{2.13}$$

$$h_{ip}^{T}\cdot\overline{\mathbb{U}}_p \le g_{ip} - m_{ip}(\pi_{ip}) \tag{2.14}$$

$$\sum_{i\in N}\sum_{p\in P}\pi_{ip} \le R \tag{2.15}$$

In Eq. (2.12), $\overline{u}_p^{t}$ is a vector representing a nominal state of state variables u_p^{t}. Eq. (2.13) limits the value of control variables. Over the set of state vectors ranging from iteration 0 to τ, this approach reformulates and relaxes the chance constraints over the expectation of a summation of indicator random variables (state vector) as generated by Eq. (2.11). The newly formulated deterministic constraints in Eq.

(2.14) are based on the nominal state of the $\mathbb{U}_p$, measured as $\overline{\mathbb{U}}_p$. Thus, the chance constraints are relaxed into deterministic constraints. Eq. (2.14) also relates to the failure of the whole system in case at least one participant fails to satisfy all the state constraints.

- *Result Synthesis*: The aforementioned risk allocation can be formulated in a decentralized collaborative optimization problem for each participant $p \in P$ as follows:

$$\text{minimize} \quad f_p(X_p) + \rho \sum_{i \in N} \pi_{ip}, \forall p \in P \tag{2.16}$$

$$\text{Subject to:} \quad \overline{u}_p^{t+1} = A_d \cdot \overline{u}_p^t + B_p \cdot x_p^t \quad \forall t = [0, ..., \tau - 1] \tag{2.17}$$

$$x_{min}^t \leq x_p^t \leq x_{max}^t \quad \forall p \in P, t = [0, ..., \tau - 1] \tag{2.18}$$

$$h_{ip}^T \cdot \overline{\mathbb{U}}_i \leq g_{ip} - m_{ip}(\pi_{ip}) \quad \forall i \in N, \ \forall p \in P \tag{2.19}$$

Finally, this approach allows treating the allocation of risks from joint constraint in Eq. (2.15) as a part of the optimization objective itself. Based on a centrally defined value of constant ρ, the iterative approach aims at minimizing individual risk for each constraint. ρ is termed as "*price of risk*" [174]. The modeling essentially reformulates the bounded risk into a form of penalty over the minimization. However, this unbounds the individual risk over the modeling approach. Thus, the individual participant takes an amount of risk based on the value of ρ. Ono and Williams proved that the amount of risk depends on a single-valued, continuous, monotonically decreasing function $m_{ip}(\pi_{ip})$ as shown in Eq. (2.19). They also showed that the generic decentralized optimization problem, as presented in Eq.s (2.16)-(2.19), has an optimal solution. This solution also satisfies all constraints ranging from Eq.s (2.12)-(2.15) including the limit over the probability of failure defined by Eq. (2.15).

2.1.4 Classification of our Research Problems

In this dissertation, we investigate a collaborative commodity delivery setup where multiple participants engage in operating vehicles from multiple depots (possibly owned by different organizations), sharing delivery of customer demands, and distributing computation load of the route generation process. Furthermore, we discuss a collaborative information monitoring framework whereby distributed decision makers can participate in monitor deployment in supply chain and track task execution using the sensor generated data stream. The network of supply chain involves flow of products from producers/distributors to customers. Such a network consists of physical locations and traversal paths among these locations. Formally, these locations can be represented as vertices of a graph while the directed edges between the vertices stand for the traversal paths (arcs) among locations. Thus, we assume the transport network of SCN as a complete directed graph where a direct edge (arc) exists between every two nodes. Therefore, in route planning for commodity delivery and monitor deployment within SCN, the number of feasible solutions increases in terms of combinatorics (such as sets, subsets, combinations or permutations) with the insertion of node(s) in network. In addition, the aforementioned problems deal with the optimization of different quantifiers (e.g. cost) over the SCN under a number of constraints. For example, route planning typically deals with cost optimization over commodity delivery to a set of customers by a set of vehicles, from one or many depots over a transport network. The constraints ensure serving every customer with limited vehicle capacity while vehicles are tasked through a tour from a depot. Similarly, monitor deployment also requires minimization of energy consumption while constrained by the limited budget. We consider that execution monitors deploy additional devices to collect and process such relayed data and deliver information to distant decision makers. Such a monitor deployment incurs setup and security cost. Limited budget places constraints on the number of deployed monitoring facilities (or monitors). Thus, it is also a Combinatorial Optimization Problem (COP). On the other hand, information monitoring is used to provide statistical interpretation on the current state of the executing processes. This statistics is generated by monitoring and control functions [245] by using well-chosen process parameters (also called *global aggregates* [244]). An increased number of parameters along with the faster

data collection may represent the current state of the work-flow process more accurately. However, huge amount of data processing diminishes punctual situation awareness due to bounded memory and computing power. Therefore, the classical monitoring objective can be also characterized as an optimization trade-off between timely response and extensive data gathering. The computational complexity of COPs depends on the underlying network. Given the characteristics of our problems and SCN, the solution generation of these problems require handling of COP where the computational complexity of optimal solution generation is $\mathcal{NP}$-Hard. Multiple previous research articles have discussed on the $\mathcal{NP}$-Hard computation complexity for the class and variants of our problems. Optimal solution generation for these problems requires exhaustive search in the solution space and evaluation of every feasible solution. This makes the optimal solution generation intractable for problem instances having large number of nodes in a complete network. Near-optimal solution generation techniques for $\mathcal{NP}$-Hard problems include greedy methods, branch and bound heuristic, primal-dual heuristic, dual descent heuristic, node partitioning and substitution, techniques, etc. The other class of programming techniques involve meta-heuristic such as Ant Colony optimization, Evolutionary Algorithms, etc. where a population of solutions continuously adapt/mutate to form better solutions. Distributed problem solving for optimization problems depends on the result sharing or the problem sharing. In result sharing, participants solve the same problem but attain different results based on different solution search criteria. Such result sharing approaches are instrumental in resolving high computational load and risk mitigation of central decision making. However, decision makers are sometimes restricted from sharing complete problem input with others due to organizational policy and security constraints. In such setting, the distributed solution techniques require dividing a central problem into multiple sub-problems among the participating decision makers and sharing computational loads based on treating the sub-problems in their respective domains [195, 209]. In both cases, sharing policies may impose no communication, partial communication or full communication of participating entities [88]. Also, participants may access a common repository in the network to store the results for evaluation purposes.

2.2 Literature Review

We review the existing research and development efforts on vehicle routing for commodity delivery, monitor deployment and event monitoring to illustrate the gap between the state-of-the-art and our proposed collaborative solution generation techniques for these problems.

2.2.1 Collaborative Vehicle Routing

In general, coordination of the vehicle routing is performed considering proximity among geographically located areas. Commodity delivery is particularly planned from one or more coordinating center(s). In the past, large scale commodity delivery plans were mostly prepared by applying transport network partitioning [77] and vehicle routing [228] in sequence. In such multi-stage approach, network partitioning helps choosing facility locations and dividing SCN at the first stage. Then, in the latter stage, VRP variants are solved (near) optimally to deliver commodities over a set of sub-networks where each sub-network has smaller size than the original SCN. So, they are often referred as "cluster-first, route-second" approach in literature [196].

Problem Elaboration

The transport network partitioning problem belongs to the more general graph partitioning problem class where the objective is to approximately partition a graph into clusters having least number of interconnections among each other. This, in essence, corresponds to the workload distribution. The graph partitioning problem is known to be $\mathcal{NP}$-complete [77]. Thus, there is no general tractable procedure that would allow efficiently performing the optimal graph partitioning for large problem instances. Nonetheless, specific approaches allow using graph partitioning for the geographical information systems, telecommunication networks, clustering, image processing, and many other areas [126, 172, 200], including operations research. This has resulted to the development of heuristic methodologies [113] that show the application of partitioning techniques using contiguous geographic clustering by network route segmentation. Many algorithms for network partitioning exist in the literature, such as K-Means clustering [119], DB-Scan algorithm, shortest path algorithms

[13], etc. However, an important limitation of the "cluster-first, route-second" approach stems from the fact that the optimal locations for depots as obtained by partitioning at the strategic level do not always optimize the cost at the operational level since the depot locations are generally chosen without considering the potential routing cost. Salhi and Rand [196] showed that the best solutions for facility locations obtained in first stage do not necessarily lead to the lowest cost solution at the routing stage. VRP aims at commodity delivery to a set of customers by a set of vehicles, from one or many depots over a transport network characterized by a full mesh graph. However, VRP models deal with the routing issues only from pre-established depots. The usual solution consists in a route for each vehicle that begins and ends at the same depot. Dantzig and Ramser [57] formally introduced the vehicle routing problem in their pioneering work on truck dispatching. VRP entails combinatorial optimization to reach optimal routing cost solution. In 2002, Toth and Vigo elaborated an extensive classification of VRP family [228]. Subsequently, Golden et al. documented the more recent advancements of the last decade [91]. The vehicle routing problem can be mainly classified into static and dynamic VRP. The static vehicle routing problem is often extended in several directions, namely: Distance Constrained VRP (DCVRP), VRP with Back Haul (VRPB) VRP with Pickup and delivery (VRPPD), VRP with Time-Window (VRPTW), etc. [91, 228]. The dynamic counterpart of the conventional VRP, commonly known as DVRP, focuses on the open-ended, time-dependent policies that focuses on the evolution of the routes as a function of inputs that evolve in real-time [229]. DVRP can be subjected to constraints with respect to quality, availability and processing of information, etc. Many variants also include probability (for example: Probabilistic TSP, Probabilistic VRP), apriori optimization (for example: APriori DTSP) and stochastic event occurrences. The vehicle is often considered uncapacitated. The DVRP problem is also modeled as a dynamic version of the VRP with time-windows.

In this thesis, we focus on a variant of a static capacitated-VRP (CVRP) which involves delivering commodities to customers from multiple depots using vehciles with limited capacity to carry commodities. However, unlike CVRP, we consider serving commodities to a customer nodes by multiple vehicles. In CVRP, demand of each customer node is served by single vehicle. It involves an additional bin packing problem, which is

needed to be solved along with the routing in order to optimally use the available capacity of the vehicles and optimally serve the customers. To avert such limitation, we investigated Split-Delivery Vehicle Routing Problem (SDVRP). Like CVRP, SDVRP also aims at minimizing the total travel cost for commodity delivery but it allows to serve individual customer demand by more than one vehicle. Therefore, SDVRP instances observe relaxed bin-packing constraints and a feasible solution always exists if the overall customer demand is less or equal to the overall vehicle capacity available. The concept of split-delivery was first introduced by Dror and Trudeau [69] and later further elaborated by Archetti and Speranza [17]. Archetti et al. [18] recently published a survey on the progress in SDVRP where the benefits of shared delivery are illustrated on various problem instances. In computation theory, CVRP and SDVRP are both expressed as a linear optimization problem with several constraints represented through linear equations and inequalities. In this respect, most of the problem models adopt Vehicle Flow Model, Commodity Flow Model or Set Partitioning Model [228]. However, an important limitation of SDVRP is the availability of the single depot.

A collaborative setting needs to overcome the limitation of SDVRP by allowing service from multiple depots through a collaborative platform. The Multi-Depot Vehicle Routing Problem (MDVRP) was introduced in 1972 by two separate research efforts. Tillman and Cain [227] published an article on multiple terminal scheduling problem while Wren and Holliday [12] published another similar article on scheduling vehicles from multiple depots. An important sub-problem in this regard is also to identify locations for these depots. Only a few research initiatives aims to model the location allocation and routing problem together [179]. This problem category is known as Location Routing Problems (LRP). It involves determining the depots and routes for the optimal number of vehicles to reach customers. Jossef Perl [187] first introduced the multi-depot routing allocation problem. Although LRP problem definition is elaborated, previous works have key modeling limitations such as candidate depots being often predefined, each depot is assumed to have one vehicle, etc. Moreover, LRP precludes shared delivery of customer demand at a node. Toward this end, we intend to incorporate idea from the Location Routing Problem (LRP) [179] for shared commodity delivery which combines the location allocation and the

vehicle routing. This involves determining depots and routes for a fixed number of vehicles to serve customers.

Recently, Gulczynski et al. [100] investigated a version of Multi-Depot Split-Delivery Vehicle Routing Problem (MDSDVRP) by extending SDVRP. Their work relates to our problem to some extent. The authors offered a mixed integer programming optimization model which addresses route cost minimization by applying split delivery among vehicles from the same or different depots. However, the employed approach is multi-stage as it first considers an assignment of customers to the depots using a distance based approximation, then solves the split-delivery VRP for each depot. Thereafter, further improvements are pursued by creating inter depot routes.

However, in MDSDVRP, the decision makers rely on a centralized planning. In this setting, a decision making center is required to have all input information to solve the problem. Furthermore, the center becomes solely responsible to solve the problem as a whole. Therefore, it is expected to have control over all assets (e.g. task executing participants/vehicles). In a business coalition, the centralized approach imposes significant limitation and often violates intelligence gathering, knowledge sharing, privacy preservation policies, etc. Moreover, as these problems are $\mathcal{NP}$-Hard, solution generation for medium and large problems often turns memory and time consuming. In this regard, collaborative planning and solution generation offer an important alternative for collaborative operation management in several sectors ranging from humanitarian aid distribution to fleet management in transport operations where joint solution generation is vital. Furthermore, decentralization eliminates the single point of failure and allows distributed self-organizing decision makers to pursue specific goal(s) based on their capabilities while aiming toward overall efficiency. In this approach, the computation loads of the decision makers also become smaller in compare with the centralized solution generation procedure. However, complete decentralization of vehicle routing problem is subjected to some critical challenges from the algorithmic and knowledge sharing perspectives. Figure 2.1 depicts the classification of collaborative vehicle routing problems. In distributed solving of the optimization problems, decentralization can be achieved through result sharing [19, 210] and/or problem sharing [195, 209].

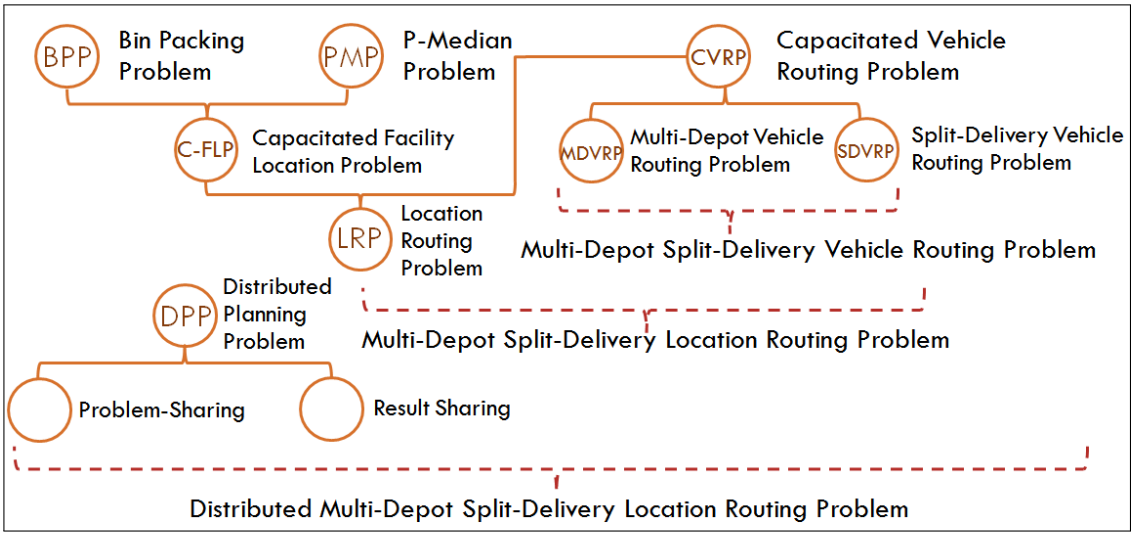


Figure 2.1: Classification of collaborative vehicle routing problems

Solution Generation

As the emerging technologies are unfolding newer possibilities for the organizations, responsibilities of mission planners and decision makers are significantly extending almost on yearly basis. Overwhelming data, coming from sensor networks, equipment and personnel, is raising a serious challenge in pursuit of getting the most benefit in proper decision making and situation awareness assessment. Additionally, global challenges, fiscal restrictions and lack of expertise in new emerging technologies are also forcing decision makers to consider distributed setting of problem solving by collaborating with partners and other stakeholders. Thus, an automated distributed decision-support framework with distributed information processing, planning and monitoring is a necessity to conduct future business operations effectively.

VRPs represent a well studied class of $\mathcal{NP}$-Hard problems [228] where analytic solution techniques, such as branch and bound [91], are intractable for medium and large scale problems. In such cases, solution generation requires various supervised solution search procedures such as: heuristic and meta-heuristic algorithms. They are commonly used to find the near-optimal solutions for $\mathcal{NP}$-Hard problems [97]. Heuristic methodologies usually involve a directed search procedure based on knowledge gathering. Meta-heuristic

algorithm generally involves comparing and iterative enhancing of candidate solutions with respect to a defined quality function measure. Some examples of these search techniques are Tabu Search [86], Ant Colony [158], Genetic Algorithms [107], etc.

The problem and solution generation of vehicle routing and scheduling under various logistics constraints have been investigated extensively [228]. In 1964, Clarke and Wright [51] first introduced a heuristic for VRP by comparing possible cost saving in vehicle route generation. Zhou et al. [262] used a genetic algorithm based approach for customer allocation to their distribution centers. Heuristics and mathematical programming for cargo loading were also studied in [97] and [20]. Concerning customer allocation, Chan and Kumar [39] developed Ant Colony based meta-heuristic optimization for managing customer demands. Ahuja et al. proposed mathematical programming based approaches for planning fastest paths under dynamic traffic conditions [8]. In similar context of logistics planning for natural disasters, Ozdamar et al. presented an approach of planning in emergency situation [178]. Yi and Kumar also used an Ant Colony based approach for optimizing disaster relief operations [252].

In SDVRP, Archetti et al. showed that the SDVRP can be solved in polynomial time if and only if common vehicle capacity (C) is 2 [18]. However, the problem becomes $\mathcal{NP}$-Hard for C > 2. Dror et al. described a local search algorithm based on specific SDVRP properties [68]. Archetti et al. obtained improved results using Tabu search [17]. Chen et al. also proposed a hybrid algorithm using the standard Clarke and Wright saving algorithm in order to solve SDVRP. Gulczynski [101] combined a mixed integer programming in conjunction with a variable length record-to-record travel algorithm to solve MDSDVRP instances. Accompanying experimental results show the benefits obtained in traversal while splitting the deliveries to the customers by vehicles starting from multiple depots.

However, in collaborative vehicles routing, decision making requires combinatorial optimization. Dividing a decision problem into sub-problems is challenging from many perspectives such as objective function, constraints, global knowledge, etc. To avoid such challenges, previous researchers often used a centralized setting to aggregate input data from the collaborators. Game theory was also studied to interact with self-interested participants with an objective to determine the best strategy for every participant in

transportation over SCN and in resource pool fragmentation [247]. Martinez et al. [151] published an incentive-based allocation to coordinate operations during aid distribution.

2.2.2 Collaborative Monitor Deployment

Information monitoring can provide valuable knowledge on the progress of the tasks and task executing participants [29]. Ensuring continuous gathering of task-execution data is also essential to attain high level supply chain visibility and enhanced situational awareness. In this context, active Radio-Frequency Identification (a-RFID) sensors are used to tag high-value items. These sensors transmit item related data to nearby readers that relay collected information to distant monitor(s). Monitoring task execution over the SCN is important in many activities. Lack of sufficient tracking capability often leads to missing crucial assets [32] and even loss of human lives [239]. Moreover, remote RFID readers are constrained by the on-board power supply and much of their energy is used to communicate with the monitors. Multiple researchers have characterized the adverse impact of larger communication distance on radio signal strength [146, 253].

Problem Elaboration

Monitor deployment and similar problems are well studied in sensor deployment [226], network management [3], facility location [141] and object localization [248]. Detailed classification of these location problems has been discussed in previous articles [75, 76, 127]. Several monitor deployment problems are captured through a graph to represent the underlying network in order to optimize path coverage [104], reduce energy consumption [181], minimize deployment [94] and transportation cost [75] under various constraints. Depending on the model and objective function, deployment locations are selected either on vertices [75] or edges [104]. Over SCN, monitor deployment is usually performed on a subset of vertices taking in consideration of aspects such as security and maintenance. Deployment is also possible between two locations (e.g. road, railway, etc.). Since, our research effort is mainly concerned about monitor deployment in transportation environment including emergency situations such as disaster relief, in this dissertation, we focus on deploying monitors over a subset of vertices. These vertices can also be treated as relay nodes with

limited source of energy as they are often established in remote places.

Monitor deployment on the vertices are usually categorized as an optimization problem derived from the classical mini-max Facility Location Problem (FLP) and p-Median Problem (p number of facilities) [67]. FLP aims to minimize the total cost of facility deployment and transport to facility nodes. These facilities can be capacitated or uncapacitated, based on their limitation in serving capacity. The transportation cost is also optimized in the p-Median Problem while such a problem restricts the number of facilities to a predefined number p.

In our setting, commodity delivery routes are already established. Therefore, given a predefined routing for commodity delivery among sensor-equipped relay nodes, we investigate optimal locations for the monitors to minimize energy consumption in data communication between monitors and relay nodes. In this setting, the monitors may serve a number of relay nodes but their total deployment cost is restricted by the total budget.

In this context, the problem of collaborative monitor deployment relates to the optimal selection of monitor locations where the total budget is split among participants. Thus, it is a distributed problem derived from the classical Facility Location Problem [75] and p-Median Problem [141] which have $\mathcal{NP}$-Hard complexity. In a centralized setting, monitor deployment problems are addressed with known budget which simplifies the formulation. If the budget is distributed among the participants then the exact value of the total budget is unknown to any participant. In this context, the formulation requires coupling through a joint chance constraint [174] to limit the probability of constraint violation by the participants.

In Section 5.3.3, we investigate collaborative monitor deployment to explore how the distributed decision makers with individual budgets may collaborate to minimize the weighted average energy consumption in the data communications.

Solution Generation

As the p-Median Problems and uncapacitated FLPs both belong to the class of $\mathcal{NP}$-Hard problems, The Heuristic or meta-heuristic techniques are extensively studied in the literature to efficiently solve the p-Median Problems and uncapacitated FLPs [75, 141,

218]. The p-median problem is often addressed using greedy approaches, node partitioning, node substitution, branch and bound, primal-dual heuristic, meta-heuristic techniques, etc. [212, 237]. Uncapacitated FLP is usually approached with the greedy algorithms, branch and bound, dual descent heuristic and other programming techniques [67]. Farahani et al. [75] and Laporte et al. [127] surveyed models, classification and approximation algorithms to solve various FLPs. Various approximation algorithms for such problems have also been surveyed [203]. Otto, and Kókai [177] discussed an evolutionary algorithm to solve p-median problem by generating a population of solutions which continuously adapt/mutate to form better solutions. Rault et al. [191] elaborated sensor placement in WSNs as Multi-Criteria Optimization and classified related energy-conservation schemes. An approximate centralized algorithm was proposed [141] to solve the facility location by considering facilities near every customer node within a specified distance followed by a rounding technique to choose among a smaller set of solutions.

In our context, we need an automated collaborative negotiation mechanism toward the near-optimal monitor deployment with individual budgets. Thus, we prefer a problem specific heuristic approach that can be locally used by the collaborative participants to reach a common near-optimal solution. For larger problems, heuristics can provide better solution quality than the greedy algorithms and partitioning techniques. However, heuristics have intrinsic limitation in guaranteeing solution quality. Recently, Shi Li [141] has published an approximate centralized algorithm to address facility location with a theoretical guarantee for the solution quality.

On the other hand, energy consumption in communications has turned extremely important with the revolution in RFID-based tracking since the RFID transmitter-receivers (also called readers) host limited on-board power supplies [181]. Gong et al. [94] applied Particle Swarm Optimization to remove redundant RFID readers for RFID tag coverage. Tian et al. [226] addressed distributed network design of RFID middleware. Palensky and Dietrich [181] discussed efficient demand management of intelligent energy systems. He et al. [105] analyzed how to quickly redeploy spare sensors in order to maintain high quality sensing by replacing the unavailable sensors [105]. These strategies in sensor allocation can be partially applied to monitor deployment problem [249]. A centralized solution for

localization problem was discussed by Xin et al. [246] which considered sensor assignment for a set of targets based on their target localization performance. The authors proposed an iterative sampling technique which is also useful for assigning relay nodes to the monitors based on energy consumption. However, monitor deployment is a more complex problem as the locations of the monitors are unknown. Moreover, if the budget is distributed among the participants, additional negotiation becomes crucial to reach a collaborative decision.

In a collaborative setup, monitor deployment is seen as a joint responsibility of multiple stakeholders who negotiate on such facility deployment. Thus, the corresponding models require addressing chance-constraint over the negotiations among decision makers to reach optimal monitor deployment. Only a small number of research efforts have addressed distributed solution generation for facility location problems. Anussornnitisarn et al. [14] presented the decentralized control of cooperative and autonomous agents for solving the distributed resource allocation problem. Pantazopoulos et al. [183] elaborated a distributed service facility placement to support user demands over a data network. The proposed heuristic locally computes partial solution (e.g. 1-median problem) and uses local traffic measurements to converge to a global solution. Smaragdakis et al. [206] also proposed a distributed solution for the web-service facility location by migrating, adding, or removing servers within a sub-network using local information about topology and demand. The algorithm starts with an initial set of facilities and progressively converges toward the final set of facilities while gradually converges to a near-optimal solution cost. Partial solution computation involves selecting vertices by various sampling methods using neighborhood search. The inclusion of neighbor vertices forms clusters around the initially selected vertices. A typical problem in this regard lies in cluster convergence when a subset of vertices in a cluster is latter found as better fitting into others. Gong et al. [93] surveyed population-distributed models along with associated algorithms for distributed evolutionary multi-objective optimization. In this case, every distributed process generates a population of solutions which continuously adapt/mutate to form better solutions.

2.2.3 Collaborative Plan Execution Monitoring

An execution monitor observes the behavior of the task execution(s) and detects if the execution(s) is consistent with the earlier planning of the tasks. Monitoring is studied with planning as a four-step process [71, 87, 232]. First, it starts with *modeling* of task execution flow, plan execution environment and task flow (projection, anticipation, perception, etc.) expectation. Second, it performs *observation* by collecting information from the results of actions, partial or complete state of the environment and qualitative/quantitative measure of its impact over task flow. Third, it conducts *state evaluation* to analyze discrepancy between the expected and the actual states while forecasting possible and forthcoming exogenous events. Thus, it learns inconsistencies in the current task flow and asserts a new estimate of (near) optimal performance for the executing plan in its current state. Finally, it carries out *plan repair* based on preexisting template or decides for *re-planning* if the execution does not meet the requirements. Monitoring is also used for learning inconsistencies in the current task flow. The scientific concept of *monitoring* and *re-planning* was introduced by McCarthy in 1977 [154]. This work discusses the impossibility of listing all preconditions in the execution environment to have its intended effect. Thus, optimization plays a vital role in monitoring.

Networking and database research communities work extensively on efficient monitoring frameworks with a typical centralized Network Operations Center (NOC) [109] where all the nodes send information. A relevant taxonomy is presented in [63] with respect to the state-of-the-art in monitoring approaches and techniques. These approaches are often related to the specification-languages, monitoring characteristic, and event-handlers. The specification-language defines properties to be monitored, abstraction level of the specification and the expressiveness (property type and monitoring level) of the language. Monitored properties can be domain specific or category specific, such as plan and policy constraints, action opportunities, adversarial activities, projections, contingency planning, reporting requirements and system fault detections [240]. They can also be general-purpose or domain agnostic, like safety, liveness, security, performance related properties, etc. [50].

The monitoring language uses algebra, automata and/or logic. Taxonomy can be

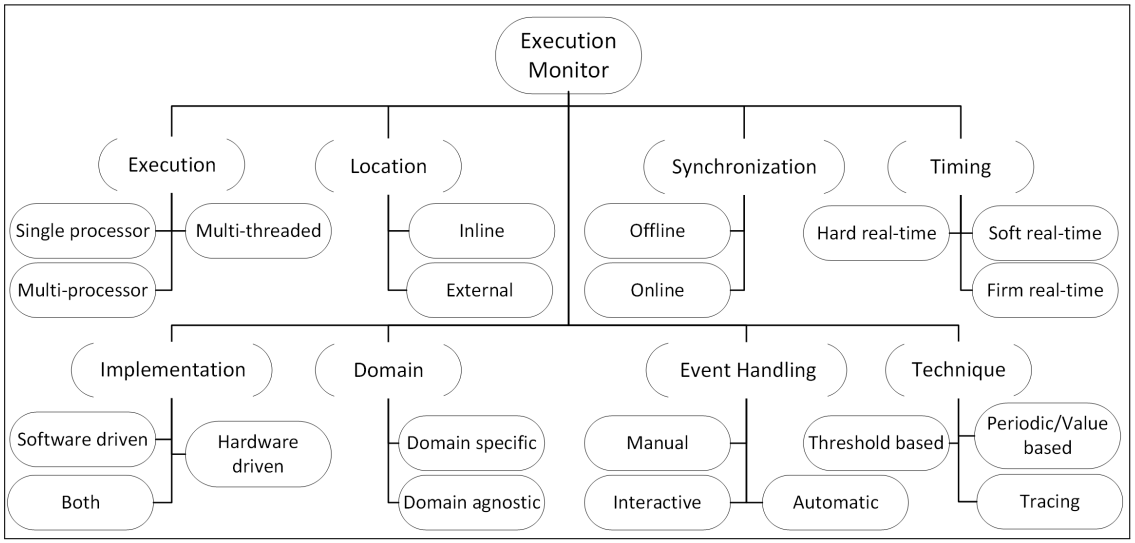


Figure 2.2: Classification of Execution Monitors

found based on the common issues of monitoring systems such as the type of software targeted by the monitoring system, the platform constraints, and code maturity level. They can be collectively categorized as operational issues. Platform-wise, monitors can be categorized as software and hardware monitors [188]. A software-driven monitoring system uses code to observe and analyze. Monitoring code can be embedded to perform in-line checking which uses the resources of the monitored application. Conversely, monitoring can be also performed externally where each monitor executes a separate thread, on the same or on a different machine. If the monitored application waits for the checking to be performed, then the monitor is called synchronous, otherwise, it is asynchronous. From the perspective of state, monitor falls into two categories: (i) offline- when the processing of the collected data is performed after execution and (ii) online- when the monitor affects the execution of processes to gather and process information. An offline monitor analyzes collected information using a larger buffer over accumulated historical data. The classification may further span over the source program type, dependencies, and the generational level. Specifically, the source type is related to the type of programs on which the monitoring activities are envisioned, including general-purpose, domain-specific, and category-specific monitoring. Monitors are generally implemented in (i) single process environment- the monitor executes

in the same process as the target program, (ii) multi-threaded environment- the monitor and target application are executing as separate threads on the same processor or (iii) multiprocessor environment- the monitor and its target program are executed on different processors. Figure 2.2 depicts a classification of the execution monitors.

Problem Elaboration

Given a set of objectives, the planning process generates a schedule for commodity delivery tasks based on available amount for resources over the SCN. The execution of these tasks often requires monitoring key plan parameters for their successful completion or adaptive re-planning. As these tasks need to comply with certain dependencies and priorities to use the available resources, during the plan execution, occurrence of exogenous events prompt change(s) in the course of actions within the execution environment. Such events can be stochastic in nature or may follow a particular pattern as discussed by Nicola et al. [61] concerning the rate-based transition systems. An effective monitoring technique identifies the requirements, analyzes the impact and carefully design transparent and sound changes for a subset of tasks such that the original task-dependencies remain unaffected with the minimal impact over the planning objectives. Monitoring process may optimize the use of resources in the planned schedule. More precisely, the roles of monitoring are: (i) to detect discrepancies between the predictions/expectations and the observations, (ii) to assess and diagnose these discrepancies, and (iii) to decide about the recovery actions in pursuit of plan repair. Thus, monitoring requires verifying sub-goals, usability, correctness, etc. [233]. Monitoring can be active or passive.

An active monitor effectively intervenes to rearrange the courses of actions using its available resources and available actions. Thus, an active monitor is aware of a set of actions. Active monitoring, also termed as monitoring enforcement, relies on two abstract principles to verify accuracy of its proposed mechanisms [142, 219], namely, soundness and transparency. The monitoring mechanism is sound when it produces valid results that always satisfy desired plan properties and dependencies. The enforcement of the monitor is transparent when it preserves the semantic validity for the executing plan. In such case, a monitor should execute a sequence of tasks unaltered for a valid input task sequence. The

application of active monitoring of plan execution has been gaining momentum among research communities in the last few decades. A survey of such monitoring techniques is presented by Pettersson [189]. In 1998, De Giacomo et al. introduced a calculus for monitoring execution [60]. Veloso et al. [233] introduced the concept of rationale based monitoring that captures planning related information and creates alternative choices for subtasks during plan execution. Fichtner et al. [78] developed Fluent Calculus to express dynamic domains in first-order logic then this was later implemented as Flux [224]. Levine [135] and Fritz et al. [83] also presented a general perspective of the active execution monitoring. They identified a number of monitored properties, such as: plan feasibility, validity, optimality, etc., and defined them as predicates to verify before and after every action [135]. Security researchers also monitor properties such safety, liveness, security, performance, etc. [142, 205]. However, active monitors often try to analyze and enforce properties without considering the presence of an untrusted execution environment that may impact over the availability of resources (for example: edit automata [142]). Furthermore, monitors, as discussed in various research areas, mainly analyze history of previous actions or events (such as: bounded-history automata) to intervene on the executing action(s) [219]. Thus, in execution monitoring, we only trust the correctness of the executed action sequences for the current schedule and intervene over actions that are currently failing or likely to fail in future. Modeling of execution environment is a challenging problem. Initiatives have been found in modeling environment as an ambient [34, 114] that host an executing process or as an interacting process that communicates with all plan executing processes. Nielson et al. elaborated it as a context [33] also. However, majority of these previous research efforts ignore the algorithmic complexities of the active monitoring and repairing plan execution and also consider the intervention instantaneous. Table 2.1 presents a comparison of some of the aforementioned initiatives as well as other research works.

A passive monitor observes important parameters of the execution process of a plan and the status of the engaged resources. It identifies discrepancies between the expected and the actual values and generates alerts, if necessary, after a quick analysis. Monitoring can be performed against either a known threshold, expected values of monitored variables

Table 2.1: Comparison of the highlighted articles on active monitoring

Articles	Architecture	Technique	Location
Allen et al. [11]	Distributed	Correlation, Impact assessment and Messages	External
Al-shaer [9]	Distributed	Logic-based Filtering	Inline
Fabre et al. [73]	Distributed	Petri-Net Unfolding	External
Fritz et al. [82]	Centralized	Action Theory and Situational Calculus	Inline
Lesser and Corkill [130]	Distributed	Knowledge-based Problem Solving	External
Steven J. Levine [135]	Centralized	Algorithmic	External
Ligatti et al. [143]	Centralized	Specialized automata	External

or over external events. In the first case, the monitor generates alerts when the value of it objective function crosses a predetermined threshold [54]. Value monitoring refers to a periodic tracking process to generate approximate values of the monitored parameters. On the other hand, event-driven monitors actually track a dynamic behavior of a program or work-flow through event sequences. It is often called tracing. Tracing is a highly memory consuming technique [10]. It notifies simultaneous occurrences of events within a certain time interval. Many frameworks implement it using periodic sampling technique with a defined interval of time [109].

Alerts are used to express different aspects of the monitored execution. For example, Threshold Crossing Alert (TCA) are triggered while an observed value crosses a threshold resulting from plan and policy constraints, action opportunities, adversarial activities, projections, contingency planning, reporting requirements, etc. Alerts are often domain specific [240] and susceptible to situation (context), dependency, relevance, reactivity and geographic location. The dependency context is elaborated in [98] while reactivity and its real-time related aspects are discussed in [163]. Francalanza et al. characterized the interaction among agents to respond occurences of event(s) across location boundaries in the absence of a global clock [80]. In this regard, interactive alert management during a plan execution represents an interesting and challenging topic for response generation during the course of action.

Our underpinning motivation for execution monitoring is to obtain a domain agnostic and fast procedure to attain high situational awareness among the decision makers with a competitive computation cost and minimal exchange of monitored data. We focus on external, passive monitoring of data-stream(s) generated by in-field sensors over

the SCN. Analysis of the data streams evolving from various systems has been researched extensively [216]. The application areas include real-time stock monitoring for financial applications [122], vehicle monitoring in roads [121], Chlorine monitoring in water streams [184], vote monitoring for elections [176], intrusion detection in large information network [148], weather and environment monitoring [85], etc. The mining of events from historical data is common in literature in order to extract relations for predicting future data values [125]. Association rule mining is one such powerful technique that can reveal interesting correlations, frequent patterns, associations or casual structures between data in large datasets [6]. In association rule mining, alert thresholds are set over (i) *support*: frequency of events (or set of events) and (ii) *confidence*: dependency or coexistence of a set of events in presence of another set of events, in the captured sequence of data stream.

Event monitoring is often performed offline on large datasets [6]. However, plan execution monitoring requires analyzing timed events from external sources. These events comes in an orderly and timely manner within a shorter interval of time to exhibit patterns and may produce valid alerts for presently executing plans [116]. Rajaraman and Ullman have pointed out the core concerns over matching the speed of data stream and computation complexity of the mining algorithm [190]. Their recommendation of in-memory single-pass real-time processing has also been supported by Jiang and Gruenwald [116]. It is also important to accurately analyze when a frequent event, also termed as itemset, turns infrequent and vice-versa [49]. In Chapter 6, we consider processing distinct items from the streams in a general form. We focus on incrementally maintaining *association rules* of our interest within the limitation of computing resources (memory and time).

Monitoring also requires adjustable sensitivity (adaptivity) of the monitor to meet the behavioral requirements during the execution of a plan. A common problem of this setup is the frequent cascading alerts, as the unfolding events bring the execution further away from expectations. Cai et al. [36] presented a practical system to illustrate the feasibility of identifying alerts from data streams in a centralized scenario. A good number of relevant research efforts have been conducted in the literature on handling reactivity during real-time response generation [163].

A full-scale collaborative plan execution monitoring requires decentralized collection

and/or mining of items. It needs partitioning of its requirements under the constraints of information access, analysis procedures and sharing mechanisms. Such partitioning needs (i) knowledge to carry out monitoring activities, (ii) information filtering, (iii) diagnosis techniques (e.g. identifying possible causes for detected discrepancies), (iv) recovery mechanisms (e.g. fast re-planing from current state) and (v) integration of monitoring process with the plan execution (e.g. plan self-repair). A suitable distributed architecture with robust design, efficient algorithms and protocols may help end-to-end data gathering and exchanging of information among the distributed nodes.

Solution Generation

The stream data capturing techniques depend on model, data structure, algorithms and architecture. Usually, event stream mining is performed using *landmark model*, *damped model* or *sliding window* [116]. In *landmark model*, data is captured from a defined point of time also known as landmark. In *damped model*, higher weight is assigned to each new transaction of the incoming data streams. The *sliding window model* evaluates a transaction if it is among the recent τ transactions. The τ is called the window size [49]. In Section 6, we consider a sliding window model for stream data mining to identify relationships among recent events.

In this regard, we have investigated the suitability of On-Line Analytical Processing (OLAP) and On-Line Analytical Mining (OLAM) for plan monitoring and re-planning. Several articles have applied OLAP and OLAM based approaches in plan execution monitoring and re-planning. Nguyen et al. [170] published an OLAP-oriented architecture tailored for providing proactive and timely response to unexpected situations by sensing and interpreting events in the environment. Furthermore, in the context of logistics, Hoa et al. [106] presented a mixed OLAP approach to enhance logistics work-flow. A data mining approach for generating fastest paths on a large routing network has been published by Awasthi et al. [23].

In data mining, rules are generated through a two phase technique. First, frequent itemset mining extracts itemsets that meet a minimum support threshold in a dataset. In its second phase, association rules are formed from the extracted frequent itemsets. Rules

can be filtered afterward as per user requirements. Most commonly used algorithms are A-priori [118, 243] and FP-Growth [215] algorithms. Several variants of these algorithms exist to mine itemsets [47, 222], e.g. Direct Hashing and Pruning (DHP) algorithm, Fast Distributed Mining (FDM) algorithm, etc. However, these algorithms have high computational complexity which limits their ability to handle data streams. The first frequent itemset mining over a data stream was proposed by Manku and Motwani in 2002 [149]. Afterward, several itemset mining algorithms were published based on clustering, classification, pattern mining, change detection, cube analysis, etc. [4]. However, researchers often prefer using tree-based data structure to concisely mine itemsets e.g. FP-Tree [96, 215], Prefix tree [48], Decision tree [164], Hoeffding tree [65], etc. Among these data structures, Decision tree and Hoeffding tree are used of data classification and decision making purposes. They can only produce approximate solutions where the solution quality largely depends on stream characteristics and computational configurations. Prefix tree and FP-Tree may host stored events for actual association rule generation. Among them, prefix tree consumes larger memory to store necessary information. However, for exact solution generation it performs better than FP-Tree during stream mining since latter keeps reorganizing branches with the changing support for tree nodes (representing itemsets). Gaber et al. [85] surveyed complexity of extracting reliable samples for different item distribution characteristics in various data-based, task-based and mining based event capturing algorithms over the stream data. Other mining approaches apply graph model [43], exponential histogram [59], weighted feature vector [184], etc. Moreover, notable other research works include tilted time window model [45], lossy counting approach [149], classification approaches [255], etc. Cheng et al. introduced semi-frequent closed itemsets to count support approximately [46]. Computation of the proper window size is also challenging for large-scale data-stream mining [4]. Also, researchers emphasize on mining selective set of important events such as Top-K monitoring [182] while handling a large volume of information. The quality of approximate solutions is evaluated using chernoff bound [149, 255] or other experimental measures [45].

Leung and Khan [133] proposed one of the first algorithms to mine frequent itemsets from streams by batch using a canonical ordered prefix tree to store transactions of the

current and previous window. Li and Lee [138] applied a bit-sequence to store items in a sliding window. This technique is called as MFI-TransSW. Left bit-shifting is used to add new transactions and remove the old ones, while "AND" operation extracts frequent itemsets [138]. LDS algorithm [64] extends the concept by using three lists to store items over the sliding window. The first and second lists store items that are respectively present and absent in a transaction. The third list captures the occurrences of items as a bit-string. Frequent itemsets are extracted from these lists, upon user request, using either Eclat[257], dEclat [258] or bEclat [30] algorithms. The choice of algorithm depends on the properties of these lists. Chi et al. [48] proposed *Moment* to mine closed frequent itemsets over a sliding window. Moment stores transactions and closed frequent itemsets in an inverse FP-Tree and a prefix tree structure (namely CET) respectively. CET keeps boundary nodes to address state changes such as: infrequent itemset becoming frequent and vice versa. NewMoment [137] and TMoment [171] extend Moment using additional hash table for easy update and quick query. A prefix tree stores single items and their occurrences at the first level. NewMoment represents the occurrences of the itemsets using a bit-string while TMoment uses an integer array of transaction IDs. Child nodes hold closed frequent itemsets and their support. Jiang and Gruenwald [115] proposed CFI-Stream to store all itemsets in a prefix tree from all transactions. Frequent itemsets are extracted upon user request by applying minimum support threshold. Yen et al. [251] proposed CloStream over sliding window. CloStream maintains two tables to store current transactions and items separately along with a list of closed itemsets. QMINE [166] also uses similar tables but the second table keeps a set of bit-vectors to track items of the first table. Using landmark model, Li et al. [139] proposed DSM-FI which processes stream data by batch and stores it in a prefix tree. However, infrequent itemsets are periodically pruned from the tree. Zhi-Jun et al. [261] divided frequent itemsets to equivalent classes. All classes of itemsets and their borders are maintained in an enumeration tree. Liu et al. [145] also mined closed frequent itemsets but stored potentially frequent itemsets of each batch in a prefix tree. Yu et al. [256] proposed an approximate frequent itemset mining algorithm using false-negative values. Using a damped model, Chang and Lee [40] proposed estDec algorithm to mine frequent itemsets from stream. estDec holds potentially frequent itemsets (of the

near-future) in a lexicographic tree with weight values assigned to each node. Additionally, Woo and Lee [242] extended estDec to EstMax algorithm that mines maximal frequent itemsets. Leung and Jiang [132] mined frequent itemsets and stored them in an FP-Tree like data structure. An expected support value is computed incrementally after processing each batch to eliminate older itemsets.

Aggarwal and Yu [5] introduced a framework for online mining of frequent itemsets and generating association rules. Their technique produces association rules with different support and confidence values without any additional computation cost but it ignores any dataset update as required for streams. Shin and Lee [204] used estDec to generate association rules from frequent itemsets over damped model. They used a stack traversal approach but these rules are always generated from the scratch, upon user request, by traversing estDec tree. Thakkar et al. [223] proposed a stream management system to mine frequent itemsets, extract and save association rules after a number of elapsed transactions over the sliding window. Saved rules help in further analysis and comparison but are not used in subsequent rule generation steps. Optional pruning eliminates duplicate and uninteresting rules. Su et al. [42] proposed FFI_Stream which is an association rule mining technique from a data stream based on quantitative attributes. Dang et al. [56] divided stream data into clusters using Sliding Window with Expectation Maximization (SWEM) technique. Itemsets are also mined using a modified version of UF_Streaming algorithm [131]. Here, a Membership Function Bias, also called MFB_Measure, explores interesting frequent itemsets that may help generating association rules but the rules are searched on request (not during the mining process). Thool and Voditel [225] mined Top-K frequent items using space-saving algorithm, by [157]. Single item $[1-1]$ rules are generated by (re)scanning the extracted itemset list. Corpinar and Gundem [52] proposed PNRMXS algorithm to mine positive and negative association rules from XML streams using landmark model. Frequent Pattern-growth (FP-Growth) is used to extract $[1-1]$ rules from each batch from scratch. Paik et al. [180] proposed mining maximal frequent itemsets from XML data streams. Association rules are generated for each batch separately and filtered using a minimum confidence threshold. Generated rules are accumulated for the entire stream in a landmark model fashion, yet the rule extraction is performed from

the scratch every time. Vijayarani and Prasannalakshmi [234] conducted an analysis on association rules generation over the data streams.

Collaborative plan execution monitoring requires a continuous planning, monitoring and re-planing framework [165, 241]. Fabre et al. [73] discussed on distributed monitoring of plans in the context of concurrent and asynchronous systems using partial unfolding approach over a Petri-Net. Al-Shaer [9] presented an active distributed monitoring strategy where a set of re-configurable and self-directed management tasks can be modified automatically at run-time by evaluating expression generated from a set of predicates. Likewise, Lavine [135] studied distributed monitoring algorithms and proposed a set of offline and online monitoring algorithms in the area of robotics. Micalizio [159] elaborated the autonomous recovery of plans operated by multiple agents using control loop of three tasks: plan monitoring, agent diagnosis, and the plan repair. Francalanza et al. [80] proposed a monitor semantics using labeled transition systems and formalized the aspect of partitioning monitors based on the locations. Babcock and Olston presented a cost-effective on-line monitoring technique [24] that involves the use of distributed constraints, insertions and deletions from dataset as well as updates of numeric values. In connection to plan repair or re-planning, new algorithms were proposed by Gerevini and Serina to detect re-planning inconsistencies [87]. This work is especially remarkable for two reasons. First, the authors conclude that a failed plan can be repaired more quickly than building a new one. Second, they show that re-planning for one inconsistency may introduce new inconsistencies later. McNeill et al. first used ontology in plan execution monitoring via multi-agent systems in 2003 [155]. They considered that the re-planning failure may occur from the faulty ontology during plan execution by multi-agent system. In 2004, Eiter et al. showed a technique for offline generation of re-planning libraries [71]. They proposed the use of reinforcement learning for choosing action less likely to fail. Van der Krogt and de Weerdt showed another advanced guided plan repair approach by non-deterministic removal of the actions from the plan and then added new actions for plan recovery based on a heuristic technique.

In connection to our scope of data mining based approaches, items are usually collected from various streams in a central location where the mining process takes place.

In the distributed setting, data stream is mined at distinct locations and the outputs are aggregated at the end. Many published articles focused on the data partitioning and the information exchange while mining stored data. Data is partitioned horizontally (row-wise) or vertically (column-wise) [231, 118]. Cheung et al. introduced a Fast Distributed Mining (FDM) algorithm with local pruning of infrequent items [47]. Otey et al. [176] developed a customizable distributed outliers detection method for continuous and categorical data to identify frequent itemsets. Park and Gupta [185], Sawant and Shah [198] and Zeng et al. [259] have investigated approaches for distributed data mining. It should be noticed that majority of the surveyed algorithms assume mining over the stored data in the distributed locations across the network. These algorithms mainly focus on itemset mining rather than rule generation. Only few articles can be seen on distributed association rule mining over the stream data. Manjhi et al. [148] have proposed distributed extraction of the frequent items using different monitors for each stream. Items are then collected centrally through the communications among monitors in a hierarchical way. Another framework has also been proposed by Sun et al. [217] to extract global frequent itemsets from the distributed data streams by aggregating locally generated frequent itemsets through adaptive filtering. The framework then communicates back with the outcome to local streams for verification and refinement of newer itemsets. In connection to pattern mining, Huang et al. [108] have proposed two Map-Reduce functions in a distributed sequential pattern mining algorithm to extract candidate patterns locally and aggregate final output from the first function. Zhang and Mao [260] have used a combination of decision trees to extract patterns locally and applied naïve bayes classifiers to form global patterns from the distributed data streams. Their algorithm uses statistical summaries to approximate the support values. Cesario and Mastroianni [38] proposed a hybrid single-pass/multiple-pass framework for mining frequent items and itemsets from the distributed data streams in multiple layers. The outcome is communicated forward and backward fashion across different layers to refine the output and minimize the error. Wu et al. [243] proposed a decentralized approach to mine event association rules over multiple streams. Frequent itemsets are filtered locally and then merged centrally using a map-reduce function. Association rules are generated upon user request using another map-reduce function. However, generated rules are neither

stored nor used in subsequent requests.

Most of the distributed techniques require reliable and extensive sharing of information through communication. Many researchers simply dealt with the maintenance of aggregation views without emphasizing the communication cost across the data sources. Several published distributed monitoring techniques, such as Marian et al. [150] and Fagin et al. [74], are affected by the limitations of numerous distributed and remote lookups. Kaminka et al. [117] also discussed on the issues encountered while coordinating a plan among geographically distributed agents in a dynamic environment. Several examples are also discussed by Cormode et al. [54] for algorithmic distributed functional monitoring problem. An improvement over communication can be achieved by setting a precision gradient as proposed by Manjhi et al. [148].

2.3 Gap Analysis

Numerous articles have been published on vehicle routing problems for more than five decades. Analysis of these articles reveals a lack of research effort particularly on the collaborative handling of route planning. Figure 2.3 depicts the research gap on the collaborative vehicle routing from the findings of the literature review. As shown in the figure, only 5-10% of the recent publications deal with multi-depot vehicle routing problem while only 2-5% discuss split-delivery vehicle routing problem. As per our analysis, published articles combining multi-depot and split-delivery vehicle routing problem are less than 10. Furthermore, most of these articles on MDVRP and SDVRP consider a centralized setup. Thus, investigation for collaborative approaches to solve VRP is an important challenge.

Recent work on Gulczynski et al. [100] relates partially to the modeling of shared delivery of commodities in multi-depot setting. However, this work exhibits some limitations in the field of application. First, it employs an apriori, rule-based allocation of customers to depots by favoring customer assignment to the nearest depots. In case of insufficient vehicle capacity, this may lead to a solution of higher cost. Such concerns were addressed in Soeanu et al. [209], where a distant depot is required to serve a customer that is closest to another depot in order to achieve overall cost reduction. Second, the

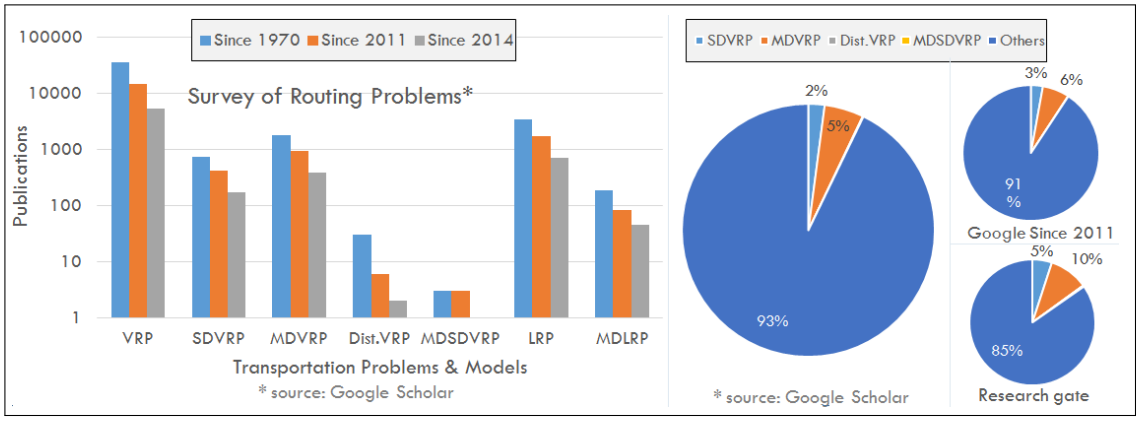


Figure 2.3: Survey of vehicle route planning problems

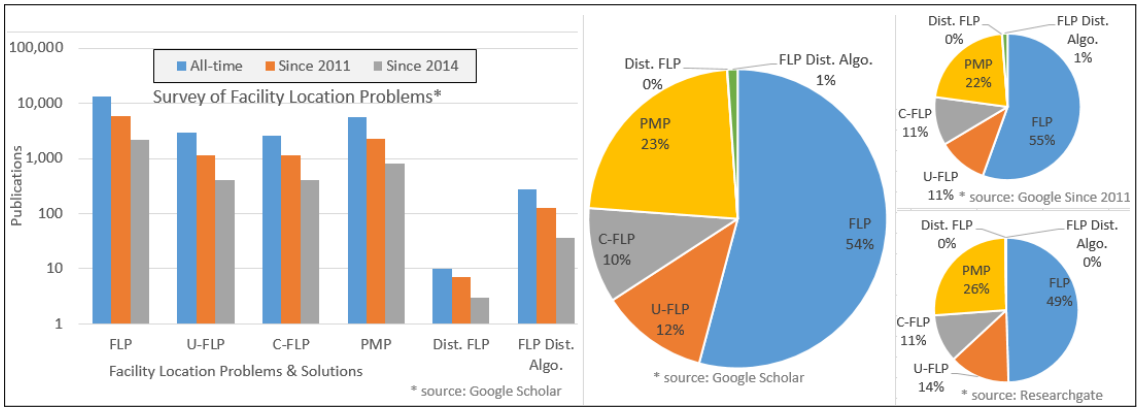


Figure 2.4: Survey of facility location problems

proposed solving technique leverages the Clarke and Wright (CW) saving mechanism that has only limited applicability to the situations where the triangle inequality is satisfied [79]. Furthermore, the model of Gulczynski et al. considers predefined locations for depots on the transport network. In contrast, we contribute by finding a novel strategy to near-optimally solve MDSDVRP in one stage and then to search for a distributed solution generation technique that can help collaborative vehicle route planning in the presence of multiple decision makers.

An analysis of the existing research literature also reveals a gap in collaborative handling of plan execution monitor deployment. As we discussed before, monitor deployment can be seen as an FLP variant. Figure 2.4 clearly depicts that while a large amount of

publications can be found on FLP, Uncapacited FLP (UFLP) and pMP, only 0-1% of this research is dedicated to the distributed algorithm generation for FLP or similar problems. Most of these existing techniques do not analyze how participants can locally compute compatible partial solutions and then collaboratively converge to a global solution in the absence of exact knowledge of input parameter(s). In this thesis, we address the important collaboration aspect of problem modeling, application development and platform deployment on plan execution monitor deployment. Chapter 5 bridges the gap in terms of jointly generating near-optimal global solution by collaborative budget contribution adjustment.

Table 2.2: Comparison of highlighted articles on itemset and association rule mining

Reference	Algorithm Name	Setting	Model	Itemsets	Caching	Rules
Wu et al.[243]	-	Distributed	Sliding Win.	1 & 2	Stored	1-1 Rules
Thakkar et al. [223], Su et al.[42]	SWIM, FFI_Stream	Centralized	Sliding Win.	All FIS	Stored	All
Paik et al.[180]	-	Centralized	Landmark	Maximal	Stored	All
Thool and Voditel[225]	Streaming-Rules	Centralized	Landmark	Top-k	Stored	1-1 Rules
Corpinar and Gundem[52]	PNRMXS	Centralized	Landmark	All FIS	Stored	1-1 Rules
Shin and Lee[204]	-	Centralized	Decay	All FIS	On req.	All: on req.
Chang and Lee[40], Leung and Jiang[132]	estDec, DUF-streaming	Centralized	Decay	All FIS	On req.	NA
Woo and Lee[242]	estMax	Centralized	Decay	Maximal	On req.	NA
Li et al.[139]	DSM-FI	Centralized	Landmark	All FIS	On req.	NA
Liu et al.[145]	FP-CDS	Centralized	Landmark	Closed	On req.	NA
Zhi-Jun et al.[261], Yu et al.[256]	-	Centralized	Landmark	All FIS	Stored	NA
Leung and Khan[133], Li and Lee[138], Tanbeer et al.[220], Deypir and Sadreddini [64]	DSTREE, MFI-TransSW, CPS, LCS	Centralized	Sliding Win.	All FIS	On req.	NA
Jiang and Gruenwald [115], Yen et al.[251], Naik and Pawar[166]	CFI-Stream, CloStream, QMINE	Centralized	Sliding Win.	Closed	On req.	NA
Chi et al.[48], Ao et al.[15], Li et al.[137], Keming Tang et al.[221], Nori et al.[171]	Moment, FPCFI-DS, NewMoment, Stream_FCI, TMoment	Centralized	Sliding Win.	Closed	Stored	NA
Manjhi et al.[148], Zhang and Mao[260]	-, -	Distributed	Decay	All FIS	Stored	NA
Cesario and Mastroianni [38]	-	Distributed	Sliding Win.	All FIS	Stored	NA
Wang and Chen[236]	-	Distributed	Landmark	Maximal	On req.	NA

Furthermore, in the vast scope of information monitoring we focus on capturing recent frequent events and their associations in a distributed setting. Table 2.2 compares highlighted research articles extracting frequent itemsets and association rules from the

stream data. Information monitoring of the data stream may reveal frequent associations among the recent events in a supply chain e.g. delay and weather condition, accidents, vehicle failure, etc. Such association is often crucial to the remote decision makers. External monitoring of an executing process requires physically distributed, autonomously operated agents that participate in a monitoring process bounded by their local and shared information that is collected within a dynamic and uncertain environment. In this respect, a major challenge lies in quickly inferring relationships among related events by mining an incoming data stream generated by the remote sensors. It can be clearly seen that only a small number of distributed approaches exists to mine frequent itemsets and association rules on the stream data. Moreover, many previous research works ignore the algorithmic complexities in extracting these itemsets and association rules which exhibits a serious limitation to practically deploy them to capture association among events over a fast data stream. Wilkins et al. [240] identified the main challenges under four categories: (i) Adaptivity: sensitivity of the monitor must meet the requirements and system's ability to extract high-value alerts and suggestions. (ii) Plan and situation-specific analysis: commonly shared plan requires communicating only relevant tasks to participants as well as accompanying instructions for coordination [98]. (iii) Provision of high-value, relevant alerts: frequent alerts issued on every occurrence of a monitored condition may overload the system and represents a known problem. The challenge also consists in minimizing and eliminating, if possible, the False Positives (FP) along with limiting the flow of unwanted or redundant alerts. (iv) Reactivity: an execution monitor should give enough time and means (techniques) to react to the events and situations in a dynamic data-rich field at a high rate of incoming events.

2.4 Summary

Logistics delivery may benefit from notable operational cost savings by increasing reliance on shared serving of customer demands by multiple agents. However, traditional logistics planning and supply chain monitoring exhibit intrinsic limitation in implementing a collaborative platform of decision making and information processing for better transportation

management. In this chapter, we thoroughly review the existing research and development efforts and analyze the gaps in the context of four core research problems. In common, all these problems are computationally hard. In the next chapters, we elaborate a set of proposed approaches to solve these core research problems of our interest. More precisely, in this thesis, these research problems are addressed in the following order:

- *Multi-Depot Split-Delivery Vehicle Routing*: We elaborate an approach on how vehicles can share responsibilities for commodity delivery in route planning.
- *Collaborative/Cooperative Route Planning for Commodity Delivery*: Next, we propose how multiple participants can jointly plan routes for commodity delivery.
- *Collaborative Monitor Deployment*: We also present how collaborative participants may deploy monitors to collect and send sensor generated information from the SCN.
- *Collaborative Event Monitoring over Data Stream*: Finally, we discuss a general purpose distributed execution information monitoring technique to track association among events from the sensor generated data stream during plan execution.

Apart from these aforementioned problems, we have identified a number key challenges for planning and execution monitoring over the supply chain network, as found during the literature review. These challenges include modeling of complex information sharing framework, reducing communication cost among agents, handling low processing power of distributed (mobile) agents, preventing simultaneous access to shared resources and devices, time synchronization, managing information privacy, security and safety issues and joint evaluation of information.

Chapter 3

Multi-Depot Split-Delivery Vehicle Routing Problem

In this chapter, we investigate an advanced decision-support platform to address a combined problem of depot assignment and logistics delivery planning. Currently available transport management systems exhibit notable gaps in optimal partitioning of transport network to share/collaborate on delivery of logistics/commodities [72]. In order to bridge the gap, we introduce a linear model of a combinatorial optimization problem and propose a generic solution search technique for multi-depot vehicle routing problems that may employ shared delivery of commodities, if needed. The experimental results show that the proposed algorithm exhibits very good performance when solving small and medium size problem instances and reasonable performance for larger instances.

3.1 Introduction

A collaborative commodity delivery planning requires operating vehicles from multiple distribution centers (depots). In such a setup, location of the depots and vehicle routes are required to be derived together while respecting predefined global constraints and limited vehicle capacity. Furthermore, vehicles are expected to cooperate for shared delivery arrangement among participating organizations. In this respect, we introduce a new problem

namely Multi-Depot Split-Delivery Vehicle Routing Problem (MDSDVRP). MDSDVRP is an extension of classical vehicle routing problem that allows multi-depot vehicle route generation with possibility of jointly serving customer demands. Furthermore, our model and solution approach also consider establishing depot at most suitable locations in the transport network which helps minimizing depot establishment and routing cost together.

In actuality, vehicle routing can be seen as a core problem in supply chain/logistics planning, with conceptual, empirical and behavioral aspects. A holistic view of the supply chain process offers an overarching perspective spanning over various facets such as facility location, vehicle routing and environmental impact. In this respect, focusing on a single aspect, for example minimizing the routing cost without considering facility locations may result in higher warehousing cost and larger externalities such as: pollution, congestion, etc. In the usual setup, the problem of multi-depot split delivery vehicle routing is considered with the common assumptions of Split-Delivery VRP (SDVRP) and Multi-Depot VRP (MDVRP) under which we essentially consider a vehicle routing problem involving commodity delivery as an abstract conceptual optimization problem [26] with few empirical details. The participating entities are depots (as starting/ending points for vehicles), customers (with deterministic demand) and vehicles (with predefined and available capacity). Typical abstractions are observed in terms of unlimited route length (not considering required stop-overs for rest, etc.) as well as deterministic infrastructure analysis (fixed traversal cost across transport network nodes, etc.). Another prevalent abstraction is to consider the problem of facility location separate as specifically employed by cluster first-route second approaches [196]. However, in this work, we emphasize the importance of considering together the problems of location allocation and vehicle routing.

The optimization goal of VRP is the overall cost minimization based on the cost assigned on each edge of the transportation network. In the literature the deterministic capacitated-VRP (CVRP) is a well-studied $\mathcal{NP}$-Hard combinatorial optimization problem having several variants and extensions [228]. In fact, the CVRP is composed of two problems: Bin-Packing and Routing. The Bin Packing Problem (BPP) addresses an optimal allocation of commodity to vehicles having deterministic capacity. The routing problem deals with the most efficient routing possible using the loaded vehicles. We may note that

in shared commodity delivery settings (which represent practical aspects at the requirements level), it is possible to determine the feasibility of a problem instance by requiring the total vehicle capacity to be greater or equal to the total demand. In other words, MDSDVRP will always yield a solution if the total available capacity is equal or more than the total demand. In this respect, MDSDVRP is less restrictive than some of the other VRP variants for which there may be no feasible solution (e.g some customers having demands larger than the capacity of a single vehicle). However, MDSDVRP still belongs to the $\mathcal{NP}$-Hard class of problems [16, 100] and is therefore intractable when approached with an exact algorithm. It is worthy to mention that it has a notable larger solution space since splitting the delivery among different vehicles is subject to combinatorial explosion. Consequently, we detail an effective heuristic technique that yields good near-optimal solutions. The contribution of this chapter can be summarized as follows:

- Elaborating a model for MDSDVRP to search optimal depot locations vehicle routes.
- Proposing a heuristic-based mechanism to quickly solve MDSDVRP near-optimally.
- Generating solution benchmarks on known problem instances and comparing with existing results.
- Analyzing performance and providing notable insights of the proposed solution.

Another notable achievement relates to the flexibility of the proposed model. This allows to customize it via small modifications (according to the need) in order to address specific problems of the VRP family that are within the scope of the proposed model. These include MDVRP (no split delivery), SDVRP (only one depot), CVRP (no split delivery and only one depot), etc. Moreover, we explain location routing through the same model that allows to consider both location allocation and vehicle routing as part of the same objective function. In this context, it is also possible to customize the depot establishment cost values such as to defining depots at specific locations. With respect to the related heuristics algorithm, it allows to generate vehicle routes with near-optimal cost while serving the customers by multiple vehicles belonging to the same or different depots.

The remainder of the chapter is organized as follows. Section 3.2 elaborates the problem and propose an Integer Linear Programming (ILP) model for MDSDVRP. Section 3.3

describes our solution generation approach. It illustrates a generic heuristic based searching mechanism designed to solve vehicle routing problem instances, MDSDVRP instances in particular. Along with the algorithm, we also discuss two improvement techniques over the initially derived solutions. Section 3.4 presents a relevant case study problem illustrating CVRP, MDVRP and MDSDVRP in order to demonstrate solution generation using the proposed approach. In Section 3.5, we provide the results and compare them to existing benchmark values. We further conduct an analysis of the results in Section 3.6 to determine appropriate ranges for the parameter values used in the solution approach. Finally, we summarize our findings in Section 3.7 by highlighting the advantages and the limitations of the proposed procedure and highlight possible with future work.

3.2 Problem Description and Modeling

In the proposed MDSDVRP model, we employ split-delivery vehicle routing and also address facility location by choosing depots from a subset of customer nodes (based on their corresponding depot establishment cost). The model also allows the use of pre-established depots by setting the corresponding depot establishing cost to zero.

3.2.1 Problem Statement

Multi-Depot Split-Delivery Vehicle Routing Problem (MDSDVRP) handles commodity delivery to customers (demand points) that are represented as nodes in a complete graph named as transport network. Given a set of nodes (V) and a set of edges (E), where E is a relation in ($V \times V$), a transport network is a complete graph G = (V, E). Each edge of the graph provides the traversal cost (c_{ij}) between the corresponding two nodes i and j. Usually, a transport network is composed of different node types: Customers (N) and Depots (D). While customer nodes are characterized with deterministic demand (integer) for commodity (d_i), depot nodes (having no demand) alternatively host vehicles ($k = 1, 2, \ldots, K$) to supply customers. In case of predefined depots and customers, a solution for an MDSDVRP instance gives the routes for each vehicle that minimizes the overall routing cost to serve all customer demands. In our proposed formulation, we further

consider that if the depots are not predefined, the solution to MDSDVRP will determine the optimal location(s) of the depot(s) within the set of customer nodes. In this case, assuming that the newly found depot(s) will serve their own need(s), the goal of problem is then to minimize the combined depot establishment and routing cost.

3.2.2 Assumptions

Classical VRP generally refers to the capacitated vehicle routing problem (CVRP) where each location (customer) has a finite demand (integer value) for the same type of commodity. A maximum number of vehicles having finite capacity (integer value) can start from and return to a single depot, with no restriction on the route length. In addition, each location is served by only one vehicle and the sum of the demands served by a vehicle does not exceed the vehicle capacity. It is also customarily assumed that the vehicle fleet is homogeneous, that is all vehicles have the same capacity. Moreover, MDVRP changes the assumption of single depot and considers the availability of more than one depot, each of which can serve any of the customers. In this setup it is also customarily assumed that the vehicle fleet is homogeneous across depots (the same maximum number of homogeneous vehicles). Capacitated VRP with split delivery (SDVRP) shares most assumptions of CVRP except that each customer can be served by one or more vehicles that jointly satisfy the total customer demand. This allows to address problems where the individual capacity of the vehicles can be less than some (or all) of the customer demands. MDSDVRP combines the assumptions of MDVRP with those of SDVRP such that each customer can be served by one or more vehicles, each of which can belong to the same or different depots. In addition to the aforementioned constraints, we intend to address the situation where there is are no pre-established depot(s) in the problem. The depots locations are then chosen from the set of customer locations. Consequently, the demands at the node locations chosen as depot locations are considered to be served by the respective depot itself. In order to solve the problem, the heuristic procedure may be exercised by one or more decision maker(s). The corresponding MDSDVRP solution algorithm assumes that all input information is exact. We assume that each decision maker has the knowledge of all available vehicles in every depot along with their capacities. S/he also knows the cost

of routing across every edge of the transport network. In case of more than one decision maker, we consider the existence of a centralized result sharing platform where multiple decision makers can share information while searching a partially different solution space.

3.2.3 MDSDVRP Modeling

VRP families of problems are usually expressed as linear optimization problem with several constraints represented through linear equations [228]. The common models are named as follows:

- *Vehicle Flow Model*: VRP is most commonly modeled using Integer Linear Programming (ILP). It uses integer variables, associated to each arc or edge of the graph. These variables track the number of traversal in each arc with respect to the constraints. The summation over the product of the number of traversals and the cost associated to each arc represents a possible outcome. The optimized solution yields the minimum value of all possible outcomes.

- *Commodity Flow Model*: The commodity flow model uses additional integer variables on the arc to represent the flow of commodities along the paths traveled by the vehicle. The model is also used more recently to get exact solution of CVRP.

- *Set Partitioning Model*: Set partitioning model is used to formulate the VRP problem as a set-partitioning problem where we consider every feasible circuit possible with respect to the constraints. The objective is to determine the collection of circuits that generates the minimum cost, serves all the customers and satisfies other additional constraints. This model generally requires a large number of variables.

In what follows, we present a vehicle flow model our problem which extends original CVRP for split-delivery and multi-depot operations. It also addresses depot localization. In a complete directed graph G = (V, E) of a transport network, let c_{ij} be an input cost matrix derived from a composed cost function (depending on various parameters) for all node pairs. We extend this transport network graph G to $G' = (V', E')$ wherein an arbitrary node 0 is added in the transport graph such that $V' = V \cup \{0\}$. The concept was

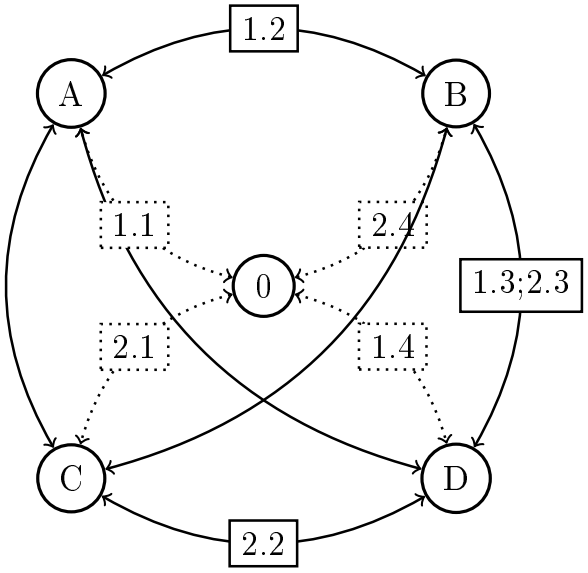


Figure 3.1: An example transport network of customers and depots

earlier introduced by Yu *et al.* [254]. The intent of including node 0 as a virtual node is to carefully capture the facility (depot) location subproblem as a part of the routing model. The inclusion of additional edges between 0 and existing customer nodes is associated to two new pair of cost values (c_{i0} and c_{0j}). The establishment cost for node j is represented as (EC_j) where $EC_j = c_{0j}$. Consequently, we write the cost function c'_{ij} as follows:

$$c'_{ij} = \begin{cases} c_{ij} & \text{if } \{i,j\} \subset V \\ EC_j & \text{if } i = 0 \text{ and } j \in V \\ c_{i0} & \text{if } j = 0 \text{ and } i \in V \end{cases}$$

Figure 3.1 shows a sample graph in the proposed setting with customers (B and D), depots (A and C) and two vehicles. Actual delivery routes can be assumed as: AB→BD→DA and CD→DB→BC. The changes to the original transport network is considered with addition of a virtual node 0 and the inclusion of dotted paths. In this case, a solution from the MDSDVRP model would result in the paths: 0A→AB→BD→D0 and 0C→CD→DB→B0. The proposed model evaluates a cost function based on its original first set of routes rather than the later derived set of routes. Actually, we incorporate additional routes 0A and 0C to reflect EC_A and EC_C in the cost function and replace the cost of

paths D0 and B0 with DA and BC respectively. Toward this consideration, we associate an additional set of boolean variables w_i on top of the usual SDVRP formulation [16] to determine the depot locations.

In formulation of the problem, we consider three given input parameters. First, the establishment cost for creating depot on node i is termed as EC_i. Second, the demand level at each node i is: d_i for all $i \in V$. Finally, the let us consider that the maximum vehicles available is K and each of them has capacity C_k. The aims of the modeling can be summarized as follows:

- Determining an optimal number of depots and their locations, in absence of predefined depots;
- Computing the optimal cost of the overall customer serving;
- Evaluating optimal number of tasked vehicles;
- Elaborating routes (potentially shared) for each tasked vehicle for overall optimal delivery cost.

The decision variables are as follows:

- $x_{ijk} \in \{0,1\}$ are boolean variables to determine routes (1 if the edge (i,j) is taken by vehicle k).
- $y_{ik} \in N$ is an integer amount of resource deposited at node i by vehicle k.
- $w_i \in \{0,1\}$ is 1 if node i is a depot.

We present the ILP model formulation as follows:

$$min \sum_{i \in V} EC_i w_i + \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k \in K} x_{ijk}, \qquad \forall i \neq j \tag{3.1}$$

Subject to:

Flow conservation:

$$\sum_{i \in V'} \sum_{k \in K} x_{ijk} \geq 1, \quad \forall j \in V', i \neq j \tag{3.2}$$

$$\sum_{j \in V} \sum_{k \in K} x_{0jk} \leq |K| \tag{3.3}$$

$$x_{0ik} = x_{i0k} \ \forall i \in V \ and \ k \in K \tag{3.4}$$

$$\sum_{i \in V'} x_{ihk} = \sum_{j \in V'} x_{hjk} \ \forall h \in V' and \ k \in K, \ i, j \neq h \tag{3.5}$$

Sub-tour elimination:

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} - \sum_{j \in S} x_{0jk} \leq |S| - 1, \quad S \subseteq V, \ |S| \geq 2, k \in K \ and \ i \neq j \tag{3.6}$$

Capacity restriction:

$$\sum_{i \in V} y_{ik} \leq C_k, \quad \forall k \in K \tag{3.7}$$

$$\sum_{k \in K} y_{ik} = d_i(1 - w_i), \quad \forall i \in V \tag{3.8}$$

$$y_{ik} \leq d_i \sum_{j \in V'} x_{ijk}, \quad \forall i \in V \ and \ k \in K \tag{3.9}$$

Depot assignment:

$$\sum_{k \in K} x_{0ik} \geq w_i, \quad \forall i \in V \tag{3.10}$$

$$x_{0ik} \leq w_i, \quad \forall i \in V, k \in K \tag{3.11}$$

Variables:

$$x_{ijk} \in \{0, 1\}; \ where \ i, j \in V', i \neq j, k \in K \tag{3.12}$$

$$w_i \in \{0, 1\}; \ where \ i \in V \tag{3.13}$$

$$y_{ik} \geq 0; \ where \ i \in V, i \neq j, k \in K \tag{3.14}$$

The objective function Eq. (3.1) minimizes total depot establishment and routing costs. With respect to the constraints, Eq. (3.2) and Eq. (3.5) impose that each customer is visited by at least one vehicle. Eq. (3.3) and Eq. (3.4) set the limit of maximum vehicles

that can be used in solution and make sure that all vehicles in operation finally return back to node 0. Eq. (3.6) is a modified version of generalized sub-tour elimination constraint from [229] in order to accommodate that all routes start and finishes at node 0 and passes through a determined depot before reaching node 0. Eq. (3.7), Eq. (3.8) and Eq. (3.9) impose that serving a customer on a route takes place if and only if the route is selected and the total on-route serving does not exceed vehicle capacity, while ensuring that the total demands of each customer are met. Finally, Eq. (3.10) and Eq. (3.11) assure that a vehicle, if serving at least a customer, must start and finish through a determined depot location. These ILP constraints further satisfy the following compound relations:

- If x_{0jk} is 1 then $\sum_{i \epsilon v} x_{ijk}$ = 1 and x_{j0k} = 1; i.e. a route for vehicle k will start and end with through a proposed depot j.
- $w_i = 1$ if x_{oik} = 1, i.e. a node i is a depot if and only if it is directly connected to node 0 in the solution.
- $\sum_{k \epsilon K} y_{ik}$ = 0 if $w_i = 1$, i.e. the demand of a prospective depot is 0 during computation of the routes.

The proposed model is flexible and extensible, allowing to capture real-world problems.

- MDSDVRP model can handle pre-established depots with inputs of low establishment cost for favored nodes and high establishment cost for others (*see* Table 3.4).
- With many vehicles and one depot in configuration, this model expresses an SDVRP.
- One can provide product or service through the same model. For simplicity, we assume that service delivery (e.g. surveillance) resembles product delivery but the vehicle capacity (C_k) is not reduced after visiting the demand nodes. However it must meet the service requirements (y_{ik}). Thus, we change Eq. (3.7) as follows:

$$y_{ik} \leq C_k, \quad \forall k \in K \; and \; \forall i \in V \tag{3.15}$$

3.3 Proposed Approach

In the following, we present an overview of the approach to solve the aforementioned research problem. Under a set of assumptions, we propose an ILP model for the multi-depot vehicle routing problem allowing joint serving of customer demands using vehicles from multiple depots. The model allows to identify the problem requirements in terms of variables and parameters. However, MDSDVRP belongs to the problem class $\mathcal{NP}$-Hard [100]. Therefore, no scalable exact solution algorithm exists to efficiently find the optimal solution. Consequently, we investigate a heuristic algorithm that can efficiently explore a large portion of the solution space in order to find a good near-optimal solution. The proposed search procedure is guided by a learning mechanism that allows to steer the search toward the most probable area of the solution space where near-optimal solutions are likely to be found. We also employ a stochastic technique to prevent the premature convergence of the algorithm to a local optimal solution.

3.3.1 Solving MDSDVRP

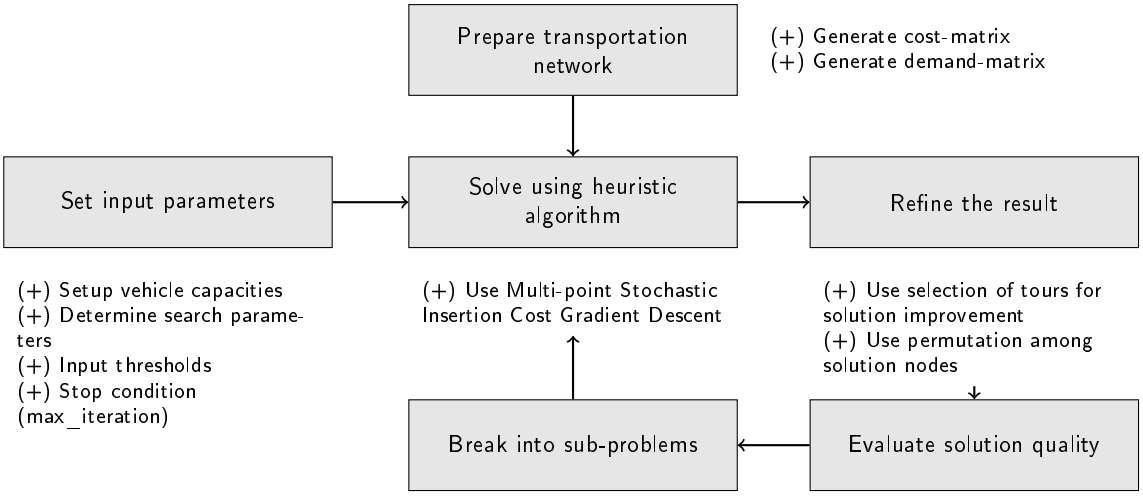


Figure 3.2: An overview of solution generation technique

MDSDVRP can be solved either analytically or using heuristic and meta-heuristic techniques. Analytically, MDSDVRP requires solving a set of linear equations as created in the model. The procedure is practical as long as the problem is smaller. Usually, ILP

models are represented using a suitable language used to describe a set of linear equations in a readable manner by both human and machine. It is also necessary to use analytical techniques like Branch and Bound, Branch and Cut to tighten the initial linear programming relaxations. We initially selected AMPL (*A Mathematical Programming Language*) to represent the problem. Then, we chose GNU Linear Programming Kit (GLPK) as a freely available solver module for AMPL based ILP formulation. GLPK uses the revised simplex method, the primal-dual interior point method for non-integer problems and the branch-and-bound algorithm along with Gomory's mixed integer cuts for (mixed) integer problems. We may additionally employ MIR cut [58], Cover cut and Clique cut [1], which are helpful when solving ILP models. However, the complexity of MDSDVRP increases exponentially with problem size. Therefore, we investigate solution finding mechanism through generative heuristics. This essentially involves the exploration of candidate solutions which are "grown" from dynamically generated solution fragments ranked on their cost. The process involves a guided search whereby the potentially good (cost effective) "fragments" are marked beneficial for subsequent exploration and retained in the data structures. The costlier fragments are continuously discarded. In this way, the grown solutions are also cost effective since only the cost effective fragments have been retained during the search.

The solution generation technique requires a preparation procedure which analyzes the transport network graph and the customer demands at the nodes in order to establish an ordered traversal map (sorted based on cost) and respectively a demand map for all customer demands. Moreover, different solution search input parameters are also required to be set before the algorithm run. After a careful analysis on various heuristic algorithms, we arrived at a modified multi-point stochastic insertion cost gradient descent algorithm to address solution search from multiple depots. The search allows the insertion of customer nodes in the explored set of route fragments, subsequently boosting the more cost effective set of routes iteratively.

Figure 3.2 presents the solution generation approach which assures that a ready solution is always available after the first pass. The algorithm is also expected to help in

cooperative solving of compound routing problems by a team of potentially remotely located agents. In this setting, during the search process, progressively better upper bounds found by different agents can be exchanged for improved convergence. The heuristic solution can be further improved using meta-heuristic like techniques such as permutation of adjacent nodes in routes, etc. Also, the approach allows the use of a divide and conquer policy in order to handle large scale problems whereby sub-problems involving a subset of the routes will be subjected to the same algorithm with the potential to yield better overall results. In the following, we discuss the details of the multi-point stochastic insertion cost gradient descent algorithm.

3.3.2 Algorithm Design

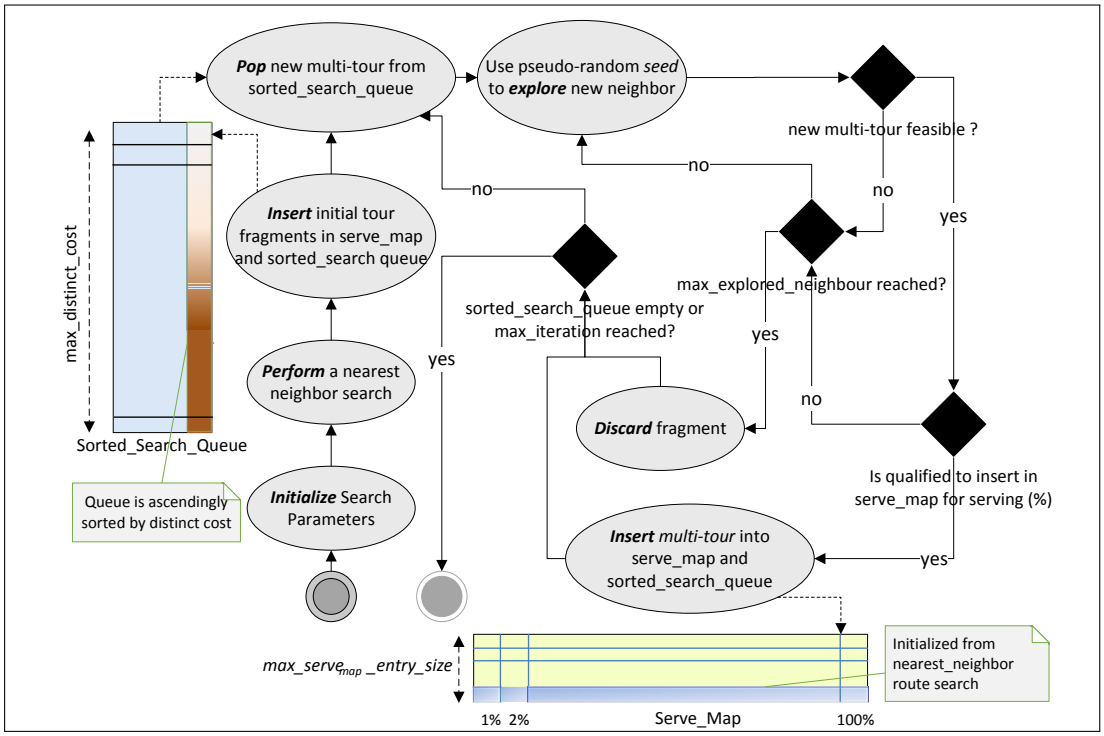


Figure 3.3: Heuristic procedure of route generation

Heuristics is employed to obtain near-optimal solution where exact algorithms and equation solving are expensive in terms of memory and time allocated to the computation. The stochastic multi-point insertion cost gradient descent is one such search technique.

It uses a seed based pseudo-random number generator to steer the solution search while allowing to reproduce (for a given problem instance) the same solution by using the same seed to solve the problem. Figure 3.3 presents the components of the high-level procedure. It starts by determining a nearest neighbor-based solution (which is computationally inexpensive to obtain), denoted by (S_{nn}) and representing an initial upper bound reference. Then, the main search starts with a base fragment (multi-tour consisting of initial vehicle locations in their depot(s)) inserted in $sorted\_search\_queue$. The latter keeps the fragments inserted into it in an ascending cost oder while the fragments with the same cost are arranged in descending order of the amount of their total demand served.

At each iteration, the topmost fragment in the $sorted\_search\_queue$ is selected by popping it out and exploring it in order to insert a neighbor node not yet served or partially served into one of its considered tours. The neighbor is identified among the unserved demand nodes by exploring them progressively up to a bound of $maxnbr$ in an ascending order of traversal cost from the last inserted element in the considered tour. An neighbor that is explored can be inserted in the tour as per the vehicle's ability to serve the node (enough remaining capacity when split delivery is not used or non-empty capacity otherwise). After inserting the neighbor in the selected fragment, the latter is updated with a corresponding increased cost of serving and increased amount of serving. The updated fragment is then qualified for storing in the $sorted\_search\_queue$ by examining if its cost of serving fits within the bounds of the corresponding $serve_{map}$ entry. The $serve_{map}$ data structure is essentially employed to build up and represent the knowledge related to the specific topology and serving availability characterizing the problem instance being solved. The knowledge gathered is represented by a set of adjustable cost bounds corresponding to particular percentages of total demand serving as discovered during fragment generation. This knowledge is used to continuously guide the search procedure by qualifying or disqualifying potential fragments while they are being explored. Thus, the $serve_{map}$ keeps entries related to the cost of serving at each related serving percentage (granularity dependent). Each entry holds a set of different cost values (for the same serving percentage) with a maximum cardinality of $max\_serve_{map}\_entry\_size$. A qualified fragment will update the corresponding $serve_{map}$ entry.

Algorithm 1 : MDSDVRP Heuristics

```
1: Input: max_iteration,S_nn,max_distinct_cost(mdc),max_explored_neighbor(maxnbr),
2:          max_serve_map_entry_size(msset),init_fragment,seed,usesplit
3: Global Knowledge: transport_network_graph(G),demand_map(dmap)
4: Output: S*
5: Initiate: S* ← S_nn; sorted_search_queue(s_que) ← ∅; serve_map ← {}; D* ← GetAllDemand(dmap);
6: Insert(init_fragment, s_que);
7: while max_iteration ≥ 0 and s_que is not empty do
8:     Pop MultiTour s from top of s_que;
9:     if s contains more than one tour then
10:        Use Shuffle(seed) to randomize their order;
11:    end if
12:    for selectedTour in s do
13:        Find next customer nextDst ← GetNextCustomer(G,LastInsertedElement(selectedTour));
14:        maxNN ← maxnbr;
15:        while maxNN > 0 and CountDistinctCostEntries(s_que) ≤ mdc do
16:            Find demand to be served: nextServeNeed ← GetDemandOf(nextDst, dmap);
17:            if nextServeNeed > 0 then
18:                if usesplit or nextServeNeed ≤ GetRemainingCapacity(selectedTour) then
19:                    InsertInTour(nextDst, selectedTour) ;
20:                end if
21:                if CostOf(s) > CostOf(S*) then
22:                    continue;
23:                end if
24:                if GetServeAmt(s) = D* or GetRemainingCapacity(s) = 0 then
25:                    S* ← s;
26:                    Remove each multi-tour(s') fragments from s_que where CostOf(s') > CostOf(s);
27:                end if
28:                if SizeOf(GetEntry(GetServeAmt(s),serve_map)) < msset or
                   CostOf(s) ≤ GetMaxValueIn(GetEntry(GetServeAmt((s),serve_map)) then
29:                    Insert(CostOf(s), GetEntry(GetServeAmt(s), serve_map));
30:                    Insert(s, s_que);
31:                end if
32:                if SizeOf(serve_set(GetServeAmt(s)))> msset then
33:                    RemoveMaxValueIn(GetEntry(GetServeAmt(s),serve_map));
34:                end if
35:            end if
36:            maxNN ← maxNN - 1;
37:        end while
38:    end for
39:    max_iteration ← max_iteration - 1;
40: end while
41: return S*;
```

During solution search, the $serve_{map}$ entries are populated by progressively smaller cost bounds in ascending order of cost. When the maximum entry size is reached, the highest value is removed from the entry set updating the knowledge related to serving the corresponding percentage of total demand. This in turn places tighter selection pressure on subsequently explored fragments with the same serving amount. The fragments placed in the $sorted\_search\_queue$ are stored until the $max\_distinct\_cost$ bound is reached. Subsequently, a fragment is discarded if its updated cost is higher than the maximum cost value of the stored fragments. When a fragment is updated such that is forms a complete solution, any member of $sorted\_search\_queue$ that has a higher cost can be removed since a complete solution with lower cost has been found. The procedure continues as long as the $sorted\_search\_queue$ is not empty and alongside progressively lower cost complete solutions can be identified, with the one having the lowest cost remaining as the final result of solution search. The effectiveness of this solution generation procedure stems from the following. First, the heuristic employs an evolving selection pressure to identify better quality fragments leveraging knowledge gathering based on the corresponding cost values stored in the $serve_{map}$. The fragments selected in this manner are potentially more able to eventually develop good near-optimal solutions. Second, the procedure exhibits a thorough local search characteristic since all the fragments in $sorted\_search\_queue$ are explored and updated according to their cost. Finally, the gradual solution generation trajectory leverages bounded local neighbor exploration which allows for faster convergence.

Algorithm 1 elaborates in pseudo-code the aforementioned concept by extending our previous work [210]. We describe next the notation used for the input parameters and the output. An upper bound solution denoted by S_{nn} provides an initial reference that can be quickly determined using the nearest neighbor. A demand map ($dmap$) holds the demands of each node. The $sorted\_search\_queue$ is an ascendantly sorted queue of solution fragments based on the cost. The $serve_{map}$ represents an associative array used for knowledge gathering where each entry contains an ordered set (of parameterized maximum cardinality - $max\_serve_{map}\_entry\_size$) containing fragment serving cost values corresponding to a related percentage of total serving. Seed ($seed$) represents a unique number used to generate repeatable (for the same seed value) pseudo-random choices. The

maximum number of neighbors to be considered in fragment exploration is represented by *maxnbr*. The *usesplit* is a binary input that selects whether the heuristic algorithm considers split-delivery. The algorithm is presented at a high level of abstraction with self explanatory names for the called procedures which follow the convention of having the first letter capitalized.

3.3.3 Property Analysis

Heuristic algorithms provide practical means to approximately solve optimization problems in short time and bounded memory with a trade-off in solution quality [124]. Moreover, specific challenges are faced during an extensive assessment of the properties characterizing heuristic algorithms. In this respect, our technique has a similar profile. Thus, in the scope of this chapter, we provide three important insights with respect to the termination, convergence and solution quality.

- *Termination: Every execution of MDSDVRP heuristic will eventually stop.*
- *Convergence: Any execution of MDSDVRP heuristic for a feasible problem will converge toward a competitive solution if the search is not stopped by the maximum iteration count.*
- *Solution quality: The solution found by executing the MDSDVRP heuristic represents the lowest local optimal within the scope of the solution search space delineated by the underlying search parameters.*

With respect to the first property, every selected multi-tour fragment is restricted to explore only within a set of customer nodes that are among the closest unserved (or partially served) *maxnbr* neighbors. Therefore, for a feasible problem, the search procedure only evaluates and stores distinct fragments that can grow at most to full solutions (all customers fully served). In this respect, the fragment exploration procedure either reduces (eventually down to 0) the remaining demand unserved or discards the disqualified fragments. Since the solution space of MDSDVRP is finite, albeit potentially very large, at the extreme (for a sufficiently large values of the search parameters), the algorithm will stop after an

exhaustive evaluation of all competitive solutions within the search space. However, with reasonable parameter values, the heuristic will only search a subset of the solution space, bounded in memory and time.

Concerning the second property, given the dynamics of the search technique, the potentially promising multi-tours will evolve similarly, (with respect to their granularity based serving percentage) before growing to a full solution. This growth characteristic is stemming from the fact that the *serve_map* restricts the storage of multi-tour fragments over a cost bounded percentage of serving and the *sorted_search_queue* stores the multi-tours in an ascending order of serving cost. Therefore, for each subsequent solution found, the probability of finding a better solution within a fixed delineated search space decreases successively and the solution improvement margin follows a natural logarithmic path. Figure 3.9 depicts the convergence characteristic . Our analysis on various problem instances reveals empirically that the solution cost (y) convergence curve over time (t) can be approximated as: $y = -C_1 \times ln(t) + C_2$, ($C_1$, C_2 are positive constants) with Pearson Coefficient of Determination (R^2) value of a few percentage points under unity.

Finally, the algorithm handles premature convergence by competitively ranking different potentially promising multi-tours based on their cost. The corresponding fragments are qualified by the bounds maintained in *serve_map* according to the percentage of serving. In essence, *serve_map* supports a guided learning over the heuristic procedure in order to promote the growth of potentially good multi-tour fragments from diverse exploration points within evolving tightness bounds. This guidance benefits the multi-point gradient descent such that each of the growing multi-tours leads toward a local optimal solution bounded by the search constraints. Therefore, the final solution emerges as the lowest one among all the local optimal solutions, generated from the diversely explored multi-tours.

3.3.4 Refinement Technique

The initial heuristic technique we introduced in [210] included solution refinement techniques for improving the routing cost, including localized node permutation and a Density

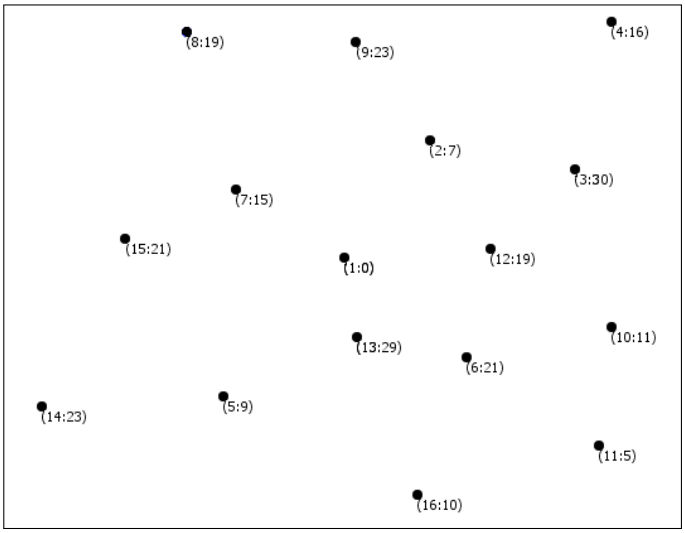
Based Clustering tour refinement. The latter was used to dynamically generate traversal cost (distance equivalent) clusters over vehicle tour nodes. This was aimed at inter-dependent route identification using incremental clustering distances over related complete solution tour pairs until all nodes of a tour belong to the same cluster. Then, if any node (except for directly density connected ones) binds two otherwise separate clusters, then the two tours are likely to allow for solution improvement by solving the corresponding sub-problem. Thus, better (lower cost) routing is likely to be identified if available. In this work, we retain the localized node permutation refinement and introduce an alternate (more scalable) non-deterministic tour delineated sub-problem refinement. Both of these refinements are detailed next. We employ the following schemes in order to locally improve the heuristic solution as follows:

- *Selective Localized Permutation Refinement*: We generate node permutations (up to a predefined threshold) around adjacent tour nodes trying to obtain a lower routing cost in the scope of a given vehicle tour. The permutation procedure is continued successively around adjacent neighbors until no further gain can be achieved.

- *Non-deterministic tour delineated sub-problem refinement*: we proceed to delineate tour pairs in a non-deterministic way for iterative improvement. In each iteration, we select a tour pair in pursuit of cost saving. If the cost saving is obtained, we attempt further improvement on all the other pairs that share a member with one of the tours previously improved. This way, better solution can be progressively identified while reducing the number of tour pairs selected for further improvement over multiple iterations until no further cost savings can be obtained.

3.4 Case Study

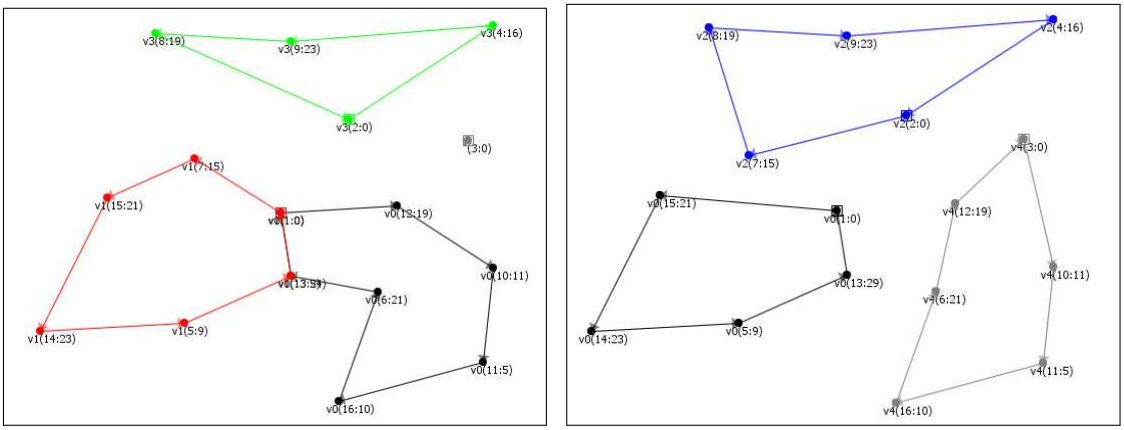
In this section, we apply the proposed algorithms on a running example of a transport network in various experimental setups. The selected problem is modified from the original CVRP problem instance: (E016-03m) as published by Golden *et al.* [90]. The configuration of the transport network and customer demand of this new problem are presented below.



Node	X	Y	Demand	EC
1	300	400	0	0
2	370	520	7	0
3	490	490	30	0
4	520	640	16	1000
5	200	260	9	1000
6	400	300	21	1000
7	210	470	15	1000
8	170	630	19	1000
9	310	620	23	1000
10	520	330	11	1000
11	510	210	5	1000
12	420	410	19	1000
13	310	320	29	1000
14	50	250	23	1000
15	120	420	21	1000
16	360	160	10	1000

Figure 3.4: Transport network and customer demands **Table 3.1:** Case study Problem Instance

Figure 3.4 presents the example problem in a 2-Dimensional Euclidean graph. The customer nodes and their demands are presented in the format of ([node no.]:[serving]). We formed the problem such that the depots may use at most two vehicles. All of them have capacity of delivering 90 units of commodity.

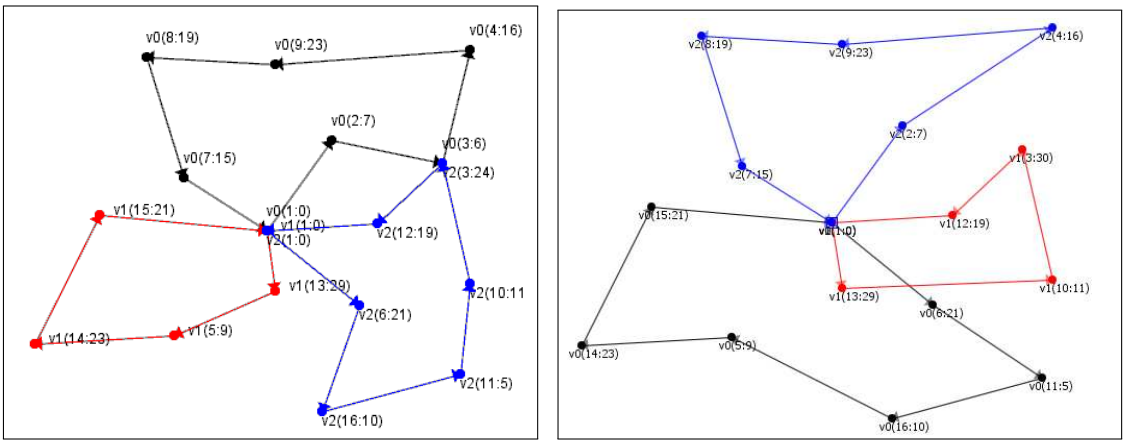


(a) Using Split-Delivery in heuristics (Cost: 2373) (b) Using No-Split Delivery in heuristics (Cost: 2402)

Figure 3.5: 3-depot heuristic solution on modified-*E016-03m* problem.

With no restriction on the number of depots, the proposed heuristic mechanism

considers nodes 1, 2 and 3 as depots. Therefore, the heuristic algorithm starts finding routes after removing the demand from these nodes after considering that they are self-served. The cost of the near-optimal solutions found with and without using split-delivery are 2373 and 2402 respectively. Figure 3.5(a) depicts the solution computed with split delivery. For this solution, it should be noted that a split delivery is formed at customer node 13 and that depot 3 was not used in serving any customers. Figure 3.5(b) represents the solution found without split-delivery which uses all depots. Afterward, we test the same example with the restriction of single depot. The setup allows to verify the performance of the proposed procedure on SDVRP and CVRP problem instances.

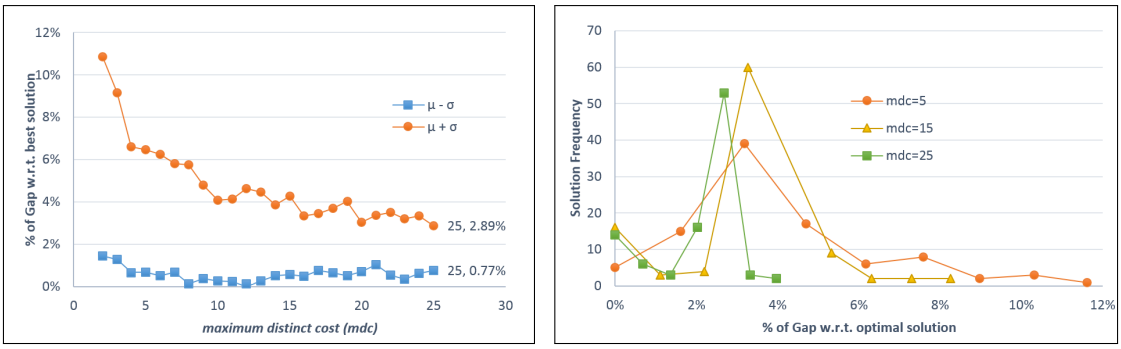


(a) Using Split-Delivery Heuristics (Cost: 2721) (b) Using No-Split Heuristics (Cost: 2786)

Figure 3.6: 1-depot 3-vehicle solution of modified-*E016-03m* using MDSDVRP.

With a restriction allowing one depot, the heuristic algorithm may perform on the case study problem instance similar to SDVRP. In such situation, we may additionally opt out for split delivery and use the same heuristic algorithm to solve problem instance as CVRP. We compare such solutions as presented in Figure 3.6. In this example, we setup a Split-Delivery VRP with one depot where depot establishment cost is 0 for node 1. The solutions found using heuristic algorithm are 2721 (*see* Figure 3.6(a)) and 2786 (*see* Figure 3.6(b)) using and without using split delivery respectively. The split delivery is slightly beneficial here as it can create better solution (with lower cost) than the optimal value achieved using CVRP [111]. The lower cost is achieved due to splitting the delivery in

node 3 where vehicles v0 and v2 deliver 6 and 24 respectively to meet the demand.



(a) Convergence Analysis of Case Study (b) Detailed Convergence Frequency Distribution

Figure 3.7: 3-depot 1-vehicle/depot convergence study on modified-*E016-03m* instance.

Figure 3.7 depicts the results obtained after performing convergence analysis on the proposed heuristic approach for the MDVRP setup of the case study problem (*see* Figure 3.5(b)). In this setting, we explore an increasingly larger search scope of the solution space by increasing the value of the maximum distinct cost (mdc) parameter. The latter represents the dominant factor in delimiting the scope of the solution search. The maximum explored neighbors ($maxnbr$) parameter is set to 3 since this value was found to perform well in benchmarks. The maximum serve map entry size ($msset$) is set to 50 accordingly. In Figure 3.7(a), we can see the solution generation evolution profile, in terms of standard deviation (σ) excursions from the mean (μ), corresponding to successive solution population batches. Each batch consists of 100 individual solutions obtained by applying the heuristic procedure repeatedly for the same mdc value but with different randomly selected seeds. We can initially note large $(\mu - \sigma)$ and $(\mu + \sigma)$ excursions that progressively narrow and finally flatten for the larger values of the mdc. In addition, it is worthy to emphasize that early on, the $(\mu - \sigma)$ excursions indicate that competitive solution are also being found albeit dispersed in population with less competitive mean value. Moreover, for the larger mdc values, the $(\mu - \sigma)$ and $(\mu + \sigma)$ excursions are distinctly narrow and positioned around competitive mean solution values. Figure 3.7(b) provide further insight with respect to the convergence of the procedure to near-optimal solution. It depicts the solution frequency histogram for small (5), medium (15) and large (25) mdc values. We can see

Table 3.2: Benchmark on known MDSDVRP problem instances [101]

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (bestHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
SQ1 (32)	2400 (2)	12 (100)	1.0 (yes)	1058 (1048)	1072 (1056.38)	-0.10 (1.00)
SQ2 (48)	3600 (3)	12 (100)	1.0 (yes)	1589 (1588)	1607 (1596.25)	0.51 (1.13)
SQ3 (64)	4800 (4)	12 (100)	1.0 (yes)	2131 (2116)	2182 (2152.25)	1.01 (2.63)
SQ4 (80)	6000 (5)	12 (100)	1.0 (yes)	2662 (2665)	2706 (2692.13)	1.16 (5.63)
SQ5 (64)	4800 (2)	25 (100)	0.96 (yes)	3422 (3446)	3481 (3461.50)	1.16 (8.63)
SQ6 (96)	7200 (3)	25 (100)	0.96 (yes)	5135 (5153)	5235 (5197.75)	1.27 (24.38)
SQ7(128)	9600 (4)	25 (100)	0.96 (yes)	6860 (6929)	7028 (6970.25)	1.61 (57.88)
SQ8(160)	12000 (5)	25 (100)	0.96 (yes)	8573 (8638)	8787 (8729.25)	1.84 (101.63)
SQ9 (96)	7200 (2)	36 (100)	1.0 (yes)	7051 (7047)	7074 (7062.75)	0.21 (41.13)
SQ10(144)	10800 (3)	36 (100)	1.0 (yes)	10578 (10587)	10668 (10638.25)	0.63 (127.75)
SQ11(192)	14400 (4)	36 (100)	1.0 (yes)	14117 (14152)	14296 (14234.13)	0.89 (302.75)
SQ12(240)	18000 (5)	36 (100)	1.0 (yes)	17645 (17780)	17886 (17829.25)	1.07 (566.75)

that for mdc = 5, the solution frequency distribution contains a wide spectrum spanning over many less competitive solution with few hits on the best solution and many hits on poor solutions. For mdc = 15, we note that the solution frequency distribution spectrum is less wide, having more hits on the best solution albeit it still includes less competitive solutions. Finally, for mdc = 25, we observe an even narrower spectrum exhibiting the most competitive solution frequency distribution along with notable hits on the best solution.

3.5 Experimental Results

We present our result in Table 3.2 by applying the proposed algorithm on known MDSDVRP instances published previously by Gulczynski *et al.* [100, 101]. The first, second and third column define the problem instance. The `totalDem` parameter represents the combined demands of all customers while `vehCnt` and `vehCap` provide the maximum number of vehicles and related capacities. In the fourth column, the tightness of an instance

Table 3.3: Benchmark on known MDVRP problem instances [53, 55]

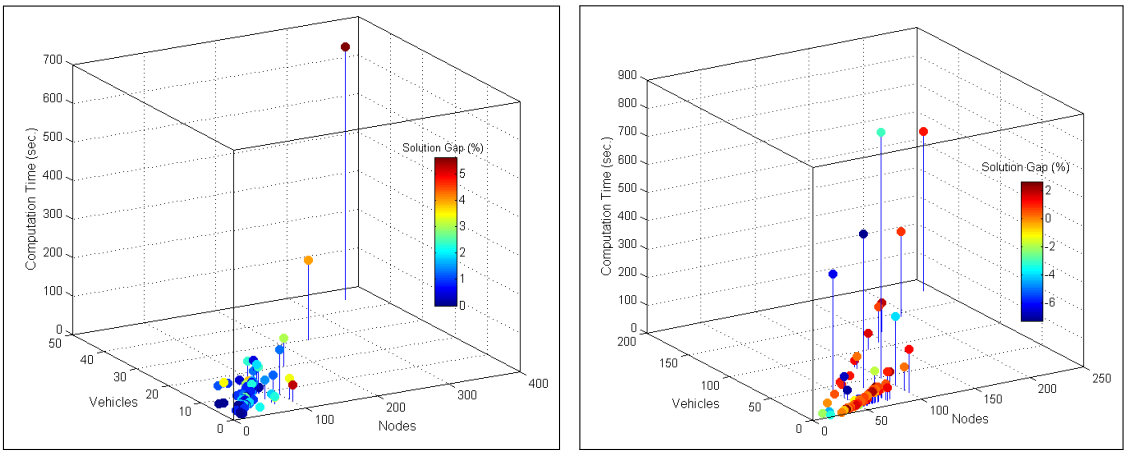
Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
p01 (50)	777 (4)	4 (80)	0.607 (no)	577 (577)	588 (583.38)	1.13 (2.38)
p02 (50)	777 (4)	2 (160)	0.607 (no)	474 (472)	484 (477.88)	0.86 (3.75)
p03 (75)	1364 (5)	3 (140)	0.649 (no)	641 (638)	648 (643.13)	0.38 (12.75)
p04 (100)	1458 (2)	8 (100)	0.911 (no)	1002 (997)	1014 (1007.13)	0.52 (52.13)
p05 (100)	1458 (2)	5 (200)	0.729 (no)	750 (749)	774 (758.63)	1.17 (41.00)
p06 (100)	1458 (3)	6 (100)	0.81 (no)	877 (890)	906 (897.75)	2.36 (40.63)
p07 (100)	1458 (4)	4 (100)	0.911 (no)	886 (883)	909 (897.00)	1.25 (25.13)
p12 (80)	432 (2)	5 (60)	0.72 (no)	1319 (1314)	1331 (1319.88)	0.11 (13.75)
p15 (160)	864 (4)	5 (60)	0.72 (no)	2505 (2539)	2614 (2583.25)	3.08 (74.13)
p18 (240)	1296 (6)	5 (60)	0.72 (no)	3702 (3835)	3872 (3855.75)	4.03 (206.25)
p21 (360)	1944 (9)	5 (60)	0.72 (no)	5475 (5737)	5862 (5799.25)	5.64 (657.13)

represents a ratio between total customer demands and total capacity available [21] while `(split)` conveys whether the heuristic solution employs shared delivery. The fifth and sixth columns offer results from our proposed approach and compare with currently best-known values. In every run, the search is invoked eight times in parallel with different seed values in eight cores of an *Intel core i7* machine. The `bestHeurVal`, `maxHeurVal` and `avgHeurVal` denotes the best, worst and average routing cost for a problem instance. The `avgGap[%]` in last column defines the percentage of the average gap of our solution with respect to the best known value. `avgTime[sec]` is the average time taken to solve the problem instance. The underlined values in column five and seven indicate finding of better result and average than previously known solutions of the corresponding problem instances. The heuristic solutions are refined by performing localized permutation on up to 4 adjacent nodes in a route. The routing details for the underlined results are presented in the appendix. To solve the SQ problem series, the proposed algorithm uses $mdc = 5$, $msset = 100$ and $maxnbr = 1$.

Similarly we solve known MDVRP instances [62] produced by Cordeau *et al.* [53].

Table 3.4: Benchmark on known SDVRP problem instances [68]

Problem (nodes)	totalDem	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
eil22 (21)	22500	4(6000)	0.937 (no)	375 (375)	379 (378.00)	0.82 (1.00)
eil23 (22)	10189	3(4500)	0.754 (no)	569 (570)	570 (570.00)	0.20 (1.00)
eil30 (29)	12750	3(4500)	0.944 (yes)	510 (510)	511 (510.50)	0.10 (1.25)
eil33 (32)	29370	4(8000)	0.917 (no)	835 (841)	843 (842.33)	0.93 (3.00)
eil51 (50)	777	5 (160)	0.971 (no)	521 (521)	533 (525.67)	0.90 (13.67)
eilA76 (75)	1364	10 (140)	0.974 (yes)	832 (831)	841 (836.25)	0.55 (57.63)
eilA101 (100)	1458	8 (200)	0.911 (no)	817 (822)	831 (827.25)	1.29 (115.75)
eilB76 (75)	1364	14 (100)	0.974 (yes)	1023 (1010)	1032 (1024.63)	0.17 (34.38)
eilB101 (100)	1458	14 (112)	0.929 (yes)	1077 (1088)	1095 (1090.60)	1.28 (155.40)
eilC76 (75)	1364	8 (180)	0.947 (yes)	735 (741)	747 (745.00)	1.40 (47.00)
eilD76 (75)	1364	7 (220)	0.885 (no)	683 (691)	695 (692.63)	1.46 (49.38)
S51D1 (50)	402	3 (160)	0.837 (no)	458 (464)	481 (467.75)	2.13 (4.75)
S51D2 (50)	1415	9 (160)	0.982 (yes)	726 (707)	715 (711.00)	-2.06 (5.00)
S51D3 (50)	2275	15 (160)	0.947 (yes)	972 (953)	970 (959.75)	-1.22 (8.00)
S51D4 (50)	4317	27 (160)	0.999 (yes)	1677 (1561)	1581 (1569.75)	-6.79 (75.00)
S51D5 (50)	3645	23 (160)	0.99 (yes)	1440 (1337)	1351 (1344.25)	-7.09 (31.88)
S51D6 (50)	6459	41 (160)	0.984 (yes)	2327 (2182)	2196 (2187.25)	-6.35 (418.63)
S76D1 (75)	614	4 (160)	0.959 (no)	594 (601)	628 (612.38)	3.04 (17.63)
S76D2 (75)	2383	15 (160)	0.992 (yes)	1147 (1091)	1108 (1099.25)	-4.29 (36.37)
S76D3 (75)	3542	23 (160)	0.962 (yes)	1474 (1440)	1456 (1448.25)	-1.74 (82.00)
S76D4 (75)	5765	37 (160)	0.973 (yes)	2257 (2096)	2115 (2102.25)	-7.31 (547.25)
S101D1 (100)	788	5 (160)	0.985 (no)	716 (733)	748 (740.80)	3.40 (53.80)
S101D2 (100)	3064	20 (160)	0.957 (yes)	1393 (1383)	1403 (1395.00)	0.20 (82.63)
S101D3 (100)	4841	31 (160)	0.976 (yes)	1975 (1889)	1904 (1897.38)	-4.05 (244.63)
S101D5 (100)	7679	48 (160)	0.999 (yes)	2915 (2814)	2866 (2828.63)	-3.00 (874.63)



(a) Time (sec.) taken to solve instances without split delivery

(b) Time (sec.) taken to solve instances with split delivery

Figure 3.8: Comparative study of solution quality and time.

We run the proposed heuristic algorithm by setting (*usesplit*) input parameter false in the heuristic procedure. For these problem instances, the algorithm uses $mdc = 25$, $msset = 100$ and $maxnbr = 3$. During the solution enhancement, we perform localized permutation up to 4 adjacent nodes in a route. Table 3.3 shows the results. In certain cases, we find similar or better results than the best known solutions published in literature [62]. With the same input parameters as used for solving the aforementioned MDVRP instances, Table 3.4 elaborates the result of applying heuristics over SDVRP instances introduced by Dror *et al.* [44]. These problem instances are carefully designed such that capacitated vehicle routes require sharing of commodity delivery in order to reach optimal routing. However, all the problem setups consist of one depot. In order to solve SDVRP instances, we place a restriction over depot deployment cost and start heuristic search directly from a known depot. With the presented input parameters, we achieve better results for many of these instances. Finally, we also solve CVRP Augerat *et. al* [21] A, B and P problem set by restricting search from a given depot and without using *split*. Table 7.1, Table 7.2, Table 7.3, Table 7.4 and Table 7.5 elaborate the results presented in the Appendix.

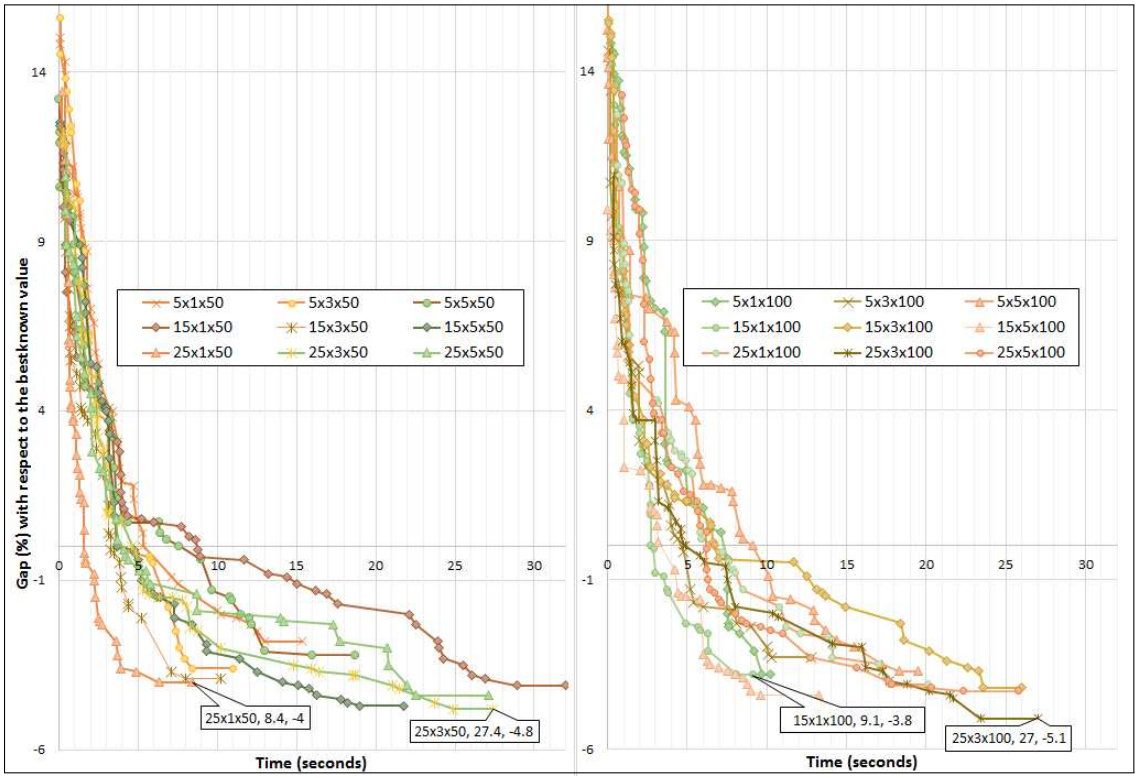


Figure 3.9: Convergence study on S76D2 instance [44] for multiple parameter values

3.6 Results Analysis

Figure 3.8 depicts an overall estimate of the time taken in solving all the problem instances considered in Section 3.5. In both sub-figures, the solution time has been calculated for all the solved problem instances with respect to number of customer nodes and vehicles. We depict the results by category based on the use of split-delivery in solution. Figure 3.8 shows that the proposed technique is successful in solving CVRP, SDVRP, MDVRP and MDSDVRP instances reasonably fast for small and medium scale problems. The solution generation is faster especially in the cases where split-delivery is not used (see Figure 3.8(a)). However, split-delivery (see Figure 3.8(b)) allows to generate good quality solutions which are some times better than the best known values for these instances. After a careful analysis of the results, it becomes apparent that the solving time increases notably with respect to customer nodes. On the other side, the increase in vehicles also adversely affects the solution time.

In analyzing the proposed procedure, we tested its performance using 18 different parameter combinations for *mdc*, *maxnbr* and *msset* as follows: *mdc*: {5,15,25}; *maxnbr*: {1,3,5}; *msset*: {50,100}. We selected a representative SDVRP instance (S76D2) [44] consisting of 76 nodes and 15 vehicles. Figure 3.9 illustrates our findings for the best solution values obtained from 8 execution runs for each parameter combination. The results are represented in two separate graphs corresponding to *msset* 50 and 100. We can notice that the algorithm converges reasonably fast during the solution search and improvement. To further analyze the convergence characteristic, we evaluate the trend-lines for the parameter combinations $(25 \times 3 \times 50)$ and $(25 \times 3 \times 100)$. Both trend-lines represent logarithmic curves: $y = -4.213ln(t) + 7.5327$ and $y = -3.468ln(t) + 6.0566$ with R^2 value 0.9609 and 0.9867 respectively. The findings indicate that (i) the general nature of convergence curve is approximately logarithmic and (ii) the coefficients (corresponding to the search parameters) determines the approximate speed of convergence and quality of the final solutions.

In fact, faster convergence corresponds to diminished solution quality. Conversely, longer search time leads to better solutions for appropriate parameter combinations. The lowest computation time is obtained with parameter combinations $(25 \times 1 \times 50)$ and $(15 \times 1 \times 100)$ in the left and the right sub-figures respectively. Likewise, the best solutions are obtained with parameter combinations $(25 \times 3 \times 50)$ and $(25 \times 3 \times 100)$ in the left and the right sub-figures respectively. We may notice the level of dissimilarity with respect to the solution finding trajectory when comparing the left (less similar) and right (more similar) sides of the figure. Thus, we emphasize the selection of parameter combinations depending on the need in terms of time and quality. We favored the combination $(25 \times 3 \times 100)$ for conducting the bulk of our benchmark experiments.

Figure 3.10 shows a performance evaluation with respect to average gap values on 18 parameter combinations over a set of 3 CVRP series (A, B and P-series [21]) consisting of known problem instances for which optimal solutions are available in the literature. We aimed at finding appropriate parameter combinations that may lead the solution generating procedure closer to optimality for a large number of problem instances. In the upper half of Figure 3.10, we can see that the larger values for the 3 parameters used for solution

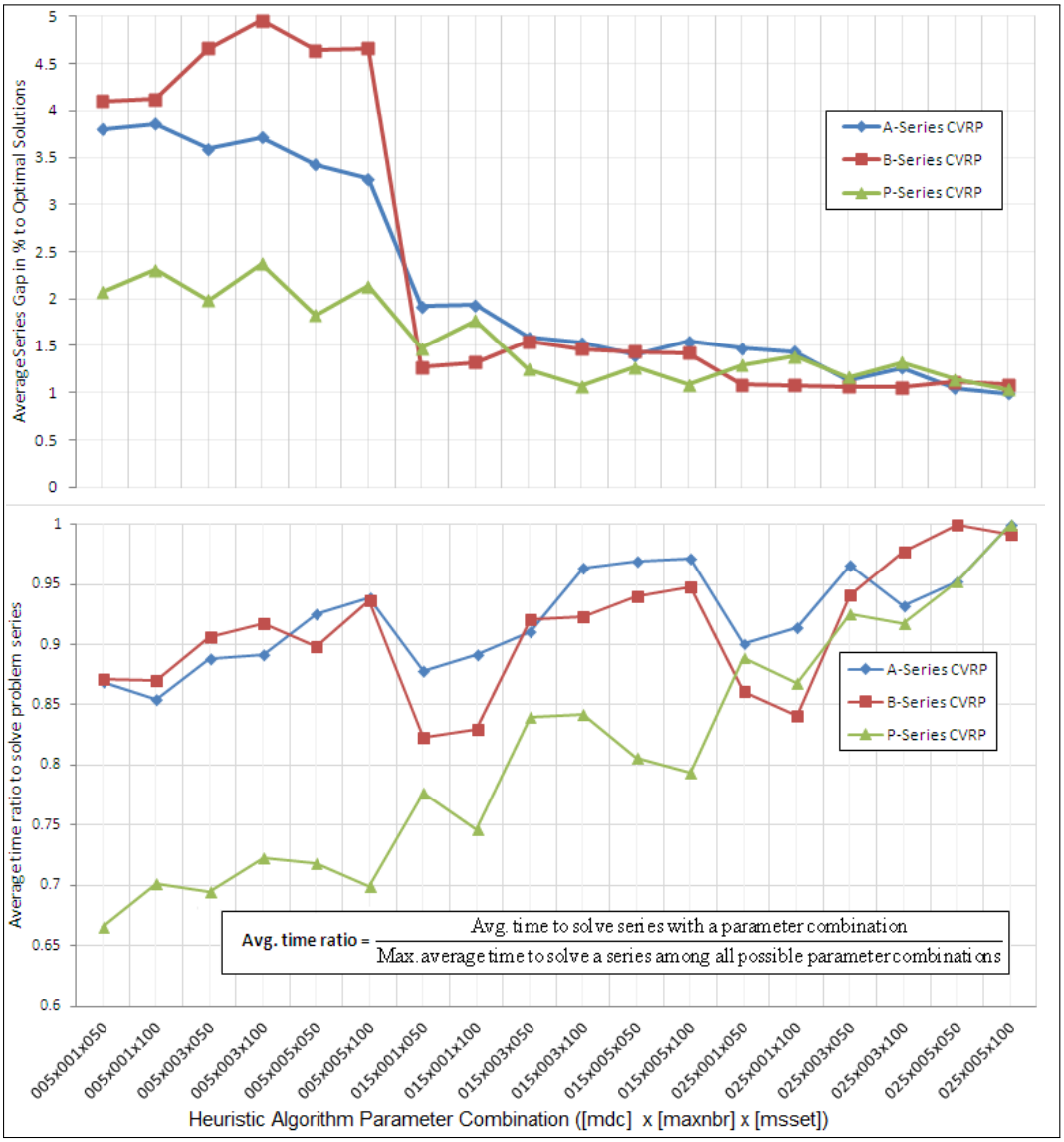


Figure 3.10: Performance comparisons of input parameters on CVRP instances

generation help in bringing average gap close to 1% for each of the 3 series. However, we can notice a gradual increase in the average time for larger values of the parameters as depicted in Figure 3.10-lower half. With respect to the latter, the y-axis represents the average computation time ratio normalized by the maximum average computation time which was obtained for the larger values of the parameters. Since we observe a plateau of the average gap values in the neighborhood of 1% while reaching a *mdc* of 25 and *maxnbr*

of 3, we favor the combinations for which the average computation time ratio is lower.

Thus, from the experiments conducted, a defined range of parameter values can be seen to correspond to finding good near-optimal solutions. For the *mdc*, we note a snap region for values over 15 which gradually reaches a plateau around a value of 25. With respect to *maxnbr*, a range from 3 to 5 appears to be most beneficial. In this context, we can estimate that values larger than 5 would lead to a certain amount of fragments grown from more distant neighbors, many of which will not eventually lead to competitive solutions. Concerning the *msset*, we can note that in some cases the lower value of 50 corresponds to better results while in other cases, the value of 100 is better suited. This indicates that a more strict (smaller *msset*) guided search may be more appropriate than a less strict (higher *msset*) for some problems and vice versa.

3.6.1 Advantages and Limitations

The benchmark results show clear advantages in deriving routes using the proposed MDS-DVRP solving approach. More precisely, the underlying heuristic is fast in producing competitive solutions. In addition, it generates good quality near-optimal or optimal solutions for many problem instances. Furthermore, the technique handles a diverse range of problems from the VRP and LRP families. Finally, it offers configurable parameter setting for the solution search to reach a user-desired trade-off between faster convergence and improved solution quality.

The proposed approach also has a number of limitations. First, the underlying heuristic does not provide a hard guarantee of the solution quality. However, we initially populate the data structures of the algorithm by generating first the nearest neighbor solution. The latter can be quickly produced by extending each vehicle route in a manner that successively incorporates unserved/underserved customer nodes among the neighbors of the last served customer node of the route. Afterward, the nearest-neighbor solution serves as initial accepting reference for the new solutions such that each new solution will be accepted if it has lower cost compared to the current accepting reference. The latter is also updated each time a better solution is found. Furthermore, the proposed solution approach is appropriate for single type of commodity delivery. Multiple commodity types

requires handling an additional multi-dimensional bin-packing problem while searching for the vehicle routes. Finally, this heuristic approach works in centralized setting where all problem data is available to a single decision maker. However, it can be used to quickly generate local solutions from the perspective of each decision maker participating in distributed solution generation.

3.7 Summary

In this chapter, we presented a generalized VRP model (MDSDVRP) suitable for multi-depot, multi-vehicle and split delivery along with a heuristic solution generation approach with efficiency refinements. The proposed approach can provide competitive solutions both in terms of cost as well as computation time for diverse instances of the VRP family. We illustrated the approach with an instructive case study example which allowed to compare different solutions of the problem variants. In this respect, we evaluated the proposed approach by generating extensive benchmark results for known problem instances belonging to different VRP variants, including CVRP, MDVRP, SDVRP and MDSDVRP. Location routing represents another important feature allowing to optimize depot location and vehicle routing in a single objective function. Finally, we also thoroughly analyzed the trade-off between faster convergence and improved solution quality.

The proposed approach has some applicability limitations in terms of single commodity delivery and the absence of time-windows, which are the subject of future work. Other future work directions include extending the technique to handle maximum vehicle tour cost and stochastic customer demands.

Chapter 4

Collaborative Multi-Depot Vehicle Routing Problem

In this chapter, we discuss two distributed approaches where participants may divide an original MDVRP instance into sub-problems and jointly reach a near-optimal global solution. The first approach presents a collaborative evolutionary learning mechanism where each participant aims to continuously improve its preference for customers by aggregating results from a number of assigned sub-problems in order to reach the near-optimal solution for the original MDVRP. The second approach involves a cooperative solution generation mechanism where self-interested participants jointly find a near-optimal solution for an MDVRP instance while each participant has its own individual objective of cost effective commodity delivery. In both cases, the underlying setup allows every participant to decide on serving a set of customers based on predefined locations for its depot(s) and fleet(s). Both approaches produce competitive solutions while having their own advantages and limitations.

4.1 Introduction

The common solution generation approaches for multi-depot vehicle routing problem (MDVRP) assume a centralized setup with complete knowledge of travel cost, depot locations,

total number of vehicles, vehicle capacity and customers [12, 179, 187]. Such assumptions are often impractical and the existing solution techniques also suffer from scalability issues to handle medium and large problem instances. In contrast, distributed solution generation algorithms help in collaborative operation management for vehicle routing problems with multiple partners. These algorithms can be designed using result sharing and/or problem sharing. The result sharing approach involves jointly searching optimal or near-optimal solution in the same solution search space of the original problem. In Section 3.3.2, we have presented a multi-point stochastic insertion cost gradient descent algorithm where each participant may collaboratively explore different regions of same search space of a problem instance based on different input seeds. This helps to simultaneously execute the search procedure and mitigate risk of computation overload by one or more participants in solution generation since the final result depends on the minimum solution value obtained across all participants. Thus, such a collaborative result sharing yields near-optimal solution in presence of multiple decision makers. However, in result sharing based decentralized approaches, each participant solves the whole problem. This requires allocating larger amount of computation resources to deal with medium and large-scale problems as the solution search space becomes larger and larger. In contrast, collaborative and cooperative problem sharing approaches can be beneficial to handle large VRP variants. Such an approach allows executing distributed algorithms at each participant's location using their computation setup. This, in turn, reduces the computation load at every participant since they solve a part of the whole problem. It also opens the scope to observe certain organizational policies.

Distributed setup of problem sharing first requires a mechanism to divide the MD-VRP instance into sub-problems. Second, it needs a supervised procedure to combine the results from the sub-problems. The combined output reflects a solution to the original problem. The supervised procedure ensures progressive convergence toward a near optimal solution. As the divided sub-problems are smaller in size compared to the original problem, solution search algorithms may perform more efficiently on the sub-problems. However, it is hard to find a proper mechanism to divide VRP customers among the participants as the most appropriate partitioning cannot always be identified without solving the original

problem itself [196]. In what follows, we elaborate the problem, present a decentralized model and solve multi-depot vehicle routing problem from the perspective of collaborative decision makers. Furthermore, we design an alternative cooperative distributed setup, where rational self-interested participants jointly solve the original problem by putting together partial solutions of their interest to deliver commodities to customers based on their vehicle capacities.

The remainder of the chapter is organized as follows. Section 4.2 describes and presents a mathematical model for Collaborative Multi-Depot Vehicle Routing Problem. Section 4.3 elaborates a collaborative solution generation approach, namely evolutionary learning (also called passive learning), in two phases. The first phase elaborates on how to divide the original problem into sub-problems. The second phase presents an evolutionary learning procedure to combine results in a distributed setup. In this approach, each participant repetitively shares routing cost of vehicle routing sub-problems and an evolutionary learning mechanism finds progressively better solutions by analyzing these continuously shared cost and partial solutions. Finally, these solutions represent near-optimal vehicle routes obtained by progressively applying heuristic techniques locally on the individual sub-problem instances. Section 4.4 illustrates a negotiation based distributed solution generation approach whereby participants actively choose customers for commodity delivery based on a game theoretic setup. In contrast to a pure collaboration approach, in negotiation, decision makers are assumed to be rational but self-interested to optimize their cost of operations. Section 4.5 describes and compares results obtained by applying techniques from both approaches on known problem instances. Finally, we summarize our findings in Section 4.6 by highlighting the advantages and the limitations of the proposed techniques.

4.2 Problem Description and Modeling

In what follows, we elaborate the problem setup. The latter considers pre-established depots in the transport network each of which is owned by a decision maker. In a distributed setting, every decision maker plans serving customers from the host depot(s) using vehicles that are associated to the depot(s). Each of these vehicles has a maximum capacity

of serving a commodity. Each depot has a number of vehicles that determines depot's capacity to serve customer demands. However, the exact number of vehicles at a depot is only known by the depot itself. We propose two mechanisms to divide MDVRP instances into multiple vehicle routing problems one for each decision maker. The solution generation approaches allow each decision maker to locally generate partial solution and share their cost related information with other participants through an iterative procedure. The procedure combines the solution cost and progressively converges toward a near-optimal solution of the original MDVRP instance. In the final solution, each participant operates independently over the transport network with its own depot(s) and its allotted customers.

4.2.1 Problem Statement

MDVRP handles commodity delivery to customers (demand points) over a common transport network. Given a set of nodes (V), representing depots (P) and customers (N), and a set of edges (E), a transport network is a complete graph G = (V, E) where E is a relation in $(V \times V)$. Each edge $\langle i, j \rangle$ has a traversal cost (c_{ij}) between corresponding nodes i and j. Similar to all vehicle routing problems, customer nodes of MDVRP are characterized with a deterministic demand (integer) for commodity (d_i). In contrast, depots do not have any demand and each of them locally hosts vehicles ($k = 1, 2, \ldots, K_p$) to supply the customers. Each vehicle k has a defined capacity (C_k) of carrying single type of commodity.

Unlike other VRP variants, in collaborative setting, information of vehicles and their individual capacities is known only to its host depot. Thus, the aim of the problem is to generate a set of vehicle routes per depot ($p \in P$) where each route starts and ends in the same depot and the total routing cost of all routes is minimum to serve all the customers.

4.2.2 Assumptions

In a collaborative MDVRP, a decision maker on each depot knows the complete transport network, all customer demands and its own capacity of commodity delivery through available vehicles. In absence of a centralized setting, a decision maker can only collaborate with other decision makers. Architecturally, such a collaboration can be implemented with

a shared-memory system or through peer-to-peer communication. In this setup, we consider that each depot shares its own interest of serving customers and divides the whole problem progressively into multiple (single or multi-depot) vehicle routing problems by owning responsibilities of serving a subset of customers. Thus, the procedure performs a decentralized problem sharing.

Without any loss of generality, we assume that a decision maker represents/owns only one depot. This simplifies the problem modeling as one participating decision maker controls the vehicles of one depot. Thus, for each depot p, the underlying transport network consists of $N' := N \cup \{p\}$ nodes. Each decision maker computes solution to its own capacitated VRP or Split-Delivery VRP instance and learns from the combined outcome of customer assignment. Progressively, they converge to a near-optimal solution of the original problem instance. A limitation to this problem design is the following. In this approach, it is only possible to consider split delivery, if required, while solving the individual subproblem instances with respect to every decision maker. In other words, the possibility of split delivery in serving customer demands with other depots is restricted since the proposed approach requires every depot to commit in advance the delivery of customer demands for its preferred customers.

4.2.3 Problem Modeling

Unlike previous formulation of MDSDVRP, in this setting, the depots are already established and every vehicle is associated to a depot. Therefore, the decision variables can be expressed as follows:

- $x_{ijkp} \in \{0, 1\}$ determines vehicle route. If the edge $\langle i, j \rangle$ is traveled by vehicle k of depot p, x_{ijkp} is 1. Otherwise, x_{ijkp} is 0.
- $y_{ikp} \in N$ denotes an integer amount of resource deposited at node i by vehicle k of depot p.

With these two sets of variables, we divide the model of MDVRP or MDSDVRP into individual capacitated VRP or split-delivery vehicle routing problem (SDVRP) instances respectively. In this regard, the optimal serving of a subset of customers from every

individual depot suffers two major challenges. First, there is a probability that certain customer demands (d_i) may remain under-served due to the individual decision making of the depots solely based on transportation cost. Second, summation of the near-optimal partial solutions from individual SDVRP instances, each of which is generated by a decision maker from its individual depot based on its vehicle capacity, does not guarantee a high-quality near-optimal solution for the original problem instance. This relates to the issue of appropriate customer assignment as mentioned before.

Mathematically, the first challenge can be addressed through probability and risk. Given a global risk factor R, the probability of serving customer demands can be captured through a joint chance constraint as previously discussed in Section 2.1.3. Since the depots are established, each decision maker aims at serving its preferred customers using its own vehicle(s). In a capacity constrained collaboration environment, we assume that every collaborative decision maker contributes in commodity delivery based on its expectation over the contribution of others. Let a state variable u_{ip} denote the contribution of participant p using its own vehicles. Then, at every state of the solution generation, each participant locally requires the amount of commodity delivery to be more than u_{ip}, i.e. $u_{ip} \leq \sum_{k \in K_p} y_{ikp}$. On the other hand, in order to have a solution to the collaborative problem setup, each customer demand must be fulfilled, i.e. $(d_i - \sum_{p \in P} u_{ip}) \leq 0$ for each customer node i. This extends Eq. (3.8) of the MDSDVRP model in Section 3.2.3. Now, in collaborative decision making, participant p actually knows only the value of u_{ip} while it can estimate the values for $u_{ip'}$ where $p \neq p'$. Such estimation helps the participant p to assess the value of its state variable u_{ip}. Therefore, in this setup, we express the joint chance constraint as follows:

$$\mathbb{P}r\left[\bigwedge_{p \in P} \bigwedge_{i \in N} \left[d_i - \sum_{p \in P} u_{ip} \leq 0\right]\right] \geq 1 - R \tag{4.1}$$

Eq. (4.1) denotes that, the risk of any customer not to be served in full, should not exceed R in order to produce a solution to this collaboration problem. However, evaluation of such a joint constraint is hard during individual solution computation [174]. To overcome the first challenge, we propose reformulating the joint chance constraint into individual chance constraints. Such a decomposition is already discussed in Section 2.1.3. Let individual risk

of every participant p to break constraint Eq. (4.1) be π_{ip} on each node $i \in N$. Then, $\pi_{ip} \geq 0$ and $\sum_{i \in N} \sum_{p \in P} \pi_{ip} \leq R$. This offers two specific advantages to the task decomposition. Reformulation helps to express the joint chance constraint using individual constraints. More importantly, it allows decomposing the risk of failure to every individual decision maker alongside the decomposition of an MDSDVRP instance into multiple SDVRP instances. Moreover, the decomposition helps to individually handle the risk through a uni-variate, convex and monotonically decreasing distribution function of risk. We denote it using function $-m_{ip}(\pi_{ip})$ similar to previous research efforts [174, 173].

The other challenge for optimal solution search relates to iterative evaluation among collaborative decision makers. Since, no participant can determine the optimal sharing of responsibilities in serving the customer demands, each participant needs to share its own solution cost of its current sub-problem with others and progressively converge to the most appropriate overall solution. Iteratively, it helps all participants to reach the optimal routing and serving of all customers. Similar to previous research efforts [174], we propose applying finite horizon optimal control where involved participants perform multi-round optimization. In order to adapt with this iterative procedure, we extend the decision variables x_{ijkp} and y_{ikp} to x^t_{ijkp} and y^t_{ikp} respectively where t denotes the round. In each round, the optimization procedure locally determines the values for decision variables $(x^t_{ijkp},\ y^t_{ikp})$, also termed as control variables. Likewise, a set of state variables $\mathbb{U}_p = [u_p^{0T}, \ldots, u_p^{tT}, \ldots, u_p^{(\tau-1)T}]$ holds the mathematical states of the multi-round optimization problem as presented before in Section 2.1.3. In the set of state variables $(\mathbb{U}_p)$, τ represents a finite horizon for the multi-round optimization while T indicates the transpose of a vector. Thus, u_p^{tT} is a vector that determines the expected contribution from a participant p for all customer nodes at a round t. The states are updated as $u_{ip}^{t+1} = A_p \cdot u_{ip}^t + (B_p \cdot (\sum_{k \in K_p} y_{ikp}^t))$ for all $t = [0, \cdots, \tau - 1]$ from round t to $t+1$ where A_p and B_p are user chosen constants. Thus, the update of future state for each customer node depends on expected contribution and the actual delivery by every participant. The following model captures the distributed problem through a system of equations.

The objective function for the distributed multi-depot split-delivery vehicle routing

problem can be formulated over the transport network as follows:

$$min \sum_{t=0}^{\tau-1} \left(\sum_{i \in N'} \sum_{j \in N'} c_{ij} \sum_{k \in K_p} x_{ijkp}^t \right) + \rho \sum_{i \in N} \pi_{ip}, \qquad \forall i \neq j \text{ and } p \in P \tag{4.2}$$

Subject to:

Flow conservation:

$$\sum_{j \in N} \sum_{k \in K_p} x_{pjkp}^t \leq |K_p| \tag{4.3}$$

$$\sum_{i \in N'} x_{ihkp}^t = \sum_{j \in N'} x_{hjkp}^t \ \forall h \in N' and\ k \in K_p,\ i, j \neq h \tag{4.4}$$

Sub-tour elimination:

$$\sum_{i \in S} \sum_{j \in S} x_{ijkp}^t - \sum_{j \in S} x_{pjkp}^t \leq |S| - 1, \quad S \subseteq N',\ |S| \geq 2, k \in K_p \text{ and } i \neq j \tag{4.5}$$

Capacity restriction:

$$\sum_{i \in N} y_{ikp}^t \leq C_k, \quad \forall k \in K_p \text{ and } p \in P \tag{4.6}$$

$$y_{ikp}^t \leq d_i \sum_{j \in N'} x_{ijkp}^t, \quad \forall i \in N \text{ and } k \in K_p \tag{4.7}$$

$$\sum_{i \in N} u_{ip}^t \leq \sum_{k \in K_p} C_k \quad \forall p \in P \tag{4.8}$$

State evaluation:

$$0 \leq u_{ip}^t \leq d_i \quad \forall i \in N \text{ and } p \in P \tag{4.9}$$

$$\sum_{t=0}^{\tau-1} (d_i - \sum_{p \in P} u_{ip}^t) \leq -m_{ip}(\pi_{ip}), \qquad \forall i \in N \tag{4.10}$$

$$u_{ip}^{t+1} = A_p \cdot u_{ip}^t + B_p \cdot \sum_{k \in K_p} y_{ikp}^t, \quad \forall k \in K_p \text{ and } p \in P \tag{4.11}$$

Variables:

$$x_{ijkp}^t \in \{0, 1\};\ where\ i, j \in N', i \neq j, k \in K, p \in P \text{ and } t = 0, \ldots, \tau \tag{4.12}$$

$$y_{ikp}^t \geq 0;\ where\ i \in N, k \in K, p \in P \text{ and } t = 0, \ldots, \tau \tag{4.13}$$

$$u_{ip}^t \geq 0;\ where\ i \in V, p \in P \text{ and } t = 0, \ldots, \tau \tag{4.14}$$

In this model, two sets of control variables are used as denoted by x^t_{ijkp} and y^t_{ikp}. At each iteration t, x^t_{ijkp} indicates whether a vehicle k from depot p moves from node i to node j. y^t_{ikp} captures the actual contribution of vehicle k of depot p at node i as determined in iteration t. Alongside, the set of state variables, u^t_{ip} represents the expected contribution from each depot p at node i in iteration t.

Eq. (4.2) represents the objective function where we iteratively minimize the routing cost for a split-delivery vehicle routing problem (SDVRP) along with individual risk of collaboration. However, in collaborative setting, SDVRP model differs from its usual modeling approaches as each participating depot is not required to serve all customer demands. Thus, the guarantee of all customers being served is only handled by the chance constraint. Nevertheless, in this setting, SDVRP routing cost optimization also represents a linear function and all SDVRP constraints (Flow conservation, Sub-tour elimination, Capacity restriction constraints) are linear as well. Thus, the SDVRP optimization is convex. Ono and Williams proved that if the total risk R is bounded by $0 \leq R \leq 0.5$, the optimization problem remains convex even under the joint chance constraint of a global risk value R [174]. This offers decomposing the global risk into individual risk π_{ip} for each participant p at each node i. Eq. (4.2) also shows customizing the influence of risk ("price of risk") using a globally determined penalty constant ρ.

Eq. (4.3) and Eq. (4.4) are flow conservation constraints similar to MDSDVRP model as discussed in Section 3.2.3. In this chance-constrained model, a depot may partially serve customers using its vehicles. However, Eq. (4.3) asserts that the total number of tours used in the solution should not exceed the limit of maximum vehicles. Eq. (4.4) indicates that the total incoming vehicles to a node are exactly equal to the total outgoing vehicles from that node. Eq. (4.5) is a generalized sub-tour elimination constraint assuring that vehicles starting from depot p are returning to the same depot. Eq. (4.6)-Eq. (4.8) deal with the amount of commodity to be delivered at a customer node. Eq. (4.6) assures that no vehicle can deliver more than its capacity while Eq. (4.7) indicates that a customer node must be visited by a vehicle in order to be served and the total amount of delivery by a vehicle does not exceed the demand of the customer node. Similarly, Eq. (4.8) assures that expected total serving for all customer nodes does not exceed total vehicle capacity. Eq.s (4.9),

(4.10) and (4.11) evaluate the mathematical state of this multi-round optimization. Eq. (4.9) assures that the expected delivery of commodity from a participant on a customer node never exceeds customer demand. Eq. (4.10) presents the risk allocation over the full serving of every customer demand. Finally, Eq. (4.11) denotes updating of the expected contribution by a participant over a node based on the current participation and the current serving at that node.

The proposed model presents three main concerns for a distributed solution technique of collaborative mutli-depot split-delivery vehicle routing problem. First, in order to handle proper risk allocation, as required in Eq. (4.10), the solution generation needs communication of state related information among participants. Second, the solution generation requires designing a specific approach to update the states at each round while locally using heuristic techniques to optimize the single depot split-delivery vehicle routing problem. Finally, finding appropriate A_p and B_p is computationally challenging for evaluation of Eq. (4.11) since y^t_{ikp} and u^t_{ip} are both integers.

4.2.4 Running Example

In order to explain the proposed approaches, we consider the same example of the CVRP instance presented as (E016-03m) in Section 3.4. The configuration of the transport network and customer demands of the problem are kept same. However, in distributed collaborative setting, we consider three participating decision makers from depot node 1, 2 and 3 respectively. Each participant has 2 vehicles of 90 units capacity for commodity delivery.

4.3 Collaborative Solution Generation: Passive Learning

In this section, we propose a multi-round technique for collaborative decision making, based on reinforced learning procedure over adaptive elitist solutions as selected from an evolving population pool of solutions. The technique allows participating decision makers to jointly solve MDVRP instances near optimally. However, in this procedure, we avert split delivery of customer demands using vehicles from different depots. More precisely, we treat the state variable $u^t_{ip} \in \{0, d_i\}$ such that it can have only one of these two values.

if $u^t_{ip} = d_i$ for a participant p, then for any other participant $p' \in P$; $p' \neq p$, $u^t_{ip'} = 0$. The reinforced learning procedure combines learning with evolutionary boosting technique. The latter was introduced by Mayr et al. [153] in order to find near-optimal solutions based on a statistical model. We call this technique passive learning.

The optimal solution generation in combinatorial optimization often renders the solution search procedure intractable for large supply chain networks. Evolutionary learning constructs a computationally tractable mechanism using boosting mechanism over a statistical model while searching in the solution search space. In this setting, the proposed approach continuously learns solution quality from previously generated solutions while improving decision making on assessing the assignment of each customer node to an appropriate depot. The associated search procedure involves a repetitive generation of more competitive solution populations through an elitist selection. The output of each generation progressively segregates sub-optimal solutions from potential near-optimal solutions. Alongside, progressive evaluation of more and more competitive solutions raises the confidence on the assignment of customer nodes. Thus, we successively minimize the error in assigning the customer nodes to the appropriate depots through a boosting technique.

4.3.1 Evolutionary Learning and Solution Pool Handling

Passive learning stems from the boosting technique [35]. Boosting composes a series of weak rules/learners into a strong learner which is generally used for classification purposes [199]. In the aforementioned model, variable $u^t_p = \{u^t_{1p}, u^t_{2p}, \ldots, u^t_{ip}, \ldots, u^t_{np}\}$ determines expected contribution from a participant p for all customer nodes. Thus, u^t_p determines the amount of deliveries for each customer node from depot p. In classification, we often call u^t_p as feature vector. Every element $u^t_{ip} \in \{0, d_i\}$ of this vector denotes either customer node i is served fully by depot p or it is not served at all. Let there also be a set of explored solutions each of which is denoted as s_{pj}. In s_{pj}, $s_{ipj} \in \{0, 1\}$ denotes whether customer node i is served by decision maker p in an observed solution s_{pj}. Then, the boosting can be captured through an additive model:

$$H^\tau(s_{pj}) = H^{\tau-1}(s_{pj}) + \alpha^{\tau T} h^\tau(s_{pj}) = \sum_{t=1}^{\tau} \alpha^{tT} h^t(s_{pj}) \tag{4.15}$$

where H^{τ} is a boosted classifier generated from τ weak hypotheses, h^t $(t = 1, \ldots, \tau)$ as presented in Eq. (4.15). In an iteration t, hypothesis h^t is incorporated with weight vector α^{tT} (T denotes transpose) to the classifier H^{t-1}. Hypothesis h^t focuses on assessing solutions that are not well classified by H^{t-1} generated at previous iteration. In research literature, various boosting techniques determine weight α^{tT} and hypothesis h^t for different error minimization objectives.

In classification, we train a classifier with a solution pool of known classes. If there exists two main classes, near-optimal and sub-optimal, a trained classifier then tries predicting a class for an unknown solution. Thus, a decision maker is able to classify a new solution using the classifier. Let o_{pj} be a boolean variable that determines a binary class. Simply, if $o_{pj} = true$ the solution is near-optimal whereas $o_{pj} = false$ corresponds to a sub-optimal solution. Thus, the classifier performs like a black box with respect to the feature variables and the solution class.

However, in collaborative solving of multi-depot vehicle routing problem, we find two key challenges. At first, since the optimal solution is unknown during search, proper $o_{pj} = true$ label generation is difficult. At every next round, a currently marked near-optimal solution can be found sub-optimal with more exploration of newer cost effective solutions. However, at any iteration, $o_{pj} = false$ can be correctly labeled based on currently best-known solution having lowest cost and a user defined *gap*. Secondly, the solution generation procedure can evaluate only a very small subset of solutions from the solution search space of medium and large scale problems.

Evolutionary learning addresses these challenges. However, unlike traditional classification, it uses an interpretable function $f_i(u^t_{ip})$ that determines the effect of a decision making such as customer node i to be served from depot p. Mayr et al. [153] introduced a link function $\zeta(\ldots)$ to represent the relation between the expectation (ξ) of o_{pj} and observed values of the decision variables (u^t_p) over a training sample s_{pj}, as shown in Eq. (4.16):

$$\zeta(\xi(o_{pj}|u^t_p = s_{pj})) = \gamma_0 + \sum_{i=1}^{n} f_i(s_{ipj}) \tag{4.16}$$

where s_{ipj} denotes the value of variable u^t_{ip} on a solution sample s_{pj}. As such, Eq. (4.16)

is a Generalized Additive Model (GAM) where γ_0 is an intercept.

The value of $\gamma_0 + \sum_{i=1}^{n} f_i(s_{ipj})$ can be approximately computed based on various independent factors (e.g. location, connections, demand, etc.) of a node in SCN for a given problem instance. In our case, we use heuristic mechanism to locally generate the response (which may be treated as partial solution). This, in turn, helps in marking a solution sample s_{pj}, at a particular round, with label $o_{pj} = true$ or $o_{pj} = false$ and segregates sub-optimal solutions.

Now among the potentially near-optimal solutions, if a feature u_{ip}^t is highly biased to a particular value 0 or d_i then $f_i(u_{ip}^t = 0)$ and $f_i(u_{ip}^t = 1)$ contribute dominantly in determining $\zeta(E(o_{pj}|u_p^t = s_{pj}))$. For example, at round t, if all the near-optimal solutions in a pool indicate that $s_{ipj} = 0$ then it means that customer node i should not be served from depot p. On the other hand, if the potentially near-optimal solutions exhibit a mix of $s_{ipj} = 0$ and $s_{ipj'} = 1$ $(\forall j, j' \in S)$, then node i's impact on the decision making is less conclusive. As such, a simple voting procedure can be adopted, at the end of every round, to (re)assign serving responsibilities for each customer node to a depot. Thus, the solution search procedure converges as more and more customer nodes increasingly start to retain previous assignment in the next round after the visiting of newer potentially near-optimal solutions.

In the aforementioned technique, each decision variable (u_{ip}^t) is considered as independent in the assessment of near-optimal and sub-optimal solutions. The variable independence allows every solution s_{pj} to be considered as a point on a space of on orthogonal axes. In this setting, near-optimal solutions can be seen as a subset of points delimited by a series of cutting planes over the same orthogonal axes. Function f_i helps to compute the cutting planes over variables u_{ip}^t. Thus, for each sample s_{pj}, the error in boosting technique can be seen as the difference: $|H^\tau(s_{pj}) - \zeta(E(o_{pj}|u_p^t = s_{pj}))|$. This error may come from wrongly attributing a customer node to a depot due to sampling limitations during each iteration. Therefore, an implicit error mitigation strategy is needed in the design.

4.3.2 Template Generation

The most appropriate distribution of customers per depot is unknown in the beginning. However, the use of a policy such as near-neighbor approach may allow more than half of customers to be rightly assigned to their appropriate depots during an initial policy-based allocation. Thereafter, we employ a multi-round distributed learning technique that evaluates a pool of competitive solution samples in each round. The solution pool is updated using a generator template of weight vectors for each depot as represented by:

$$W_p^t = \begin{pmatrix} w_{1,1}^t & w_{1,2}^t & \cdots & w_{1,m}^t \\ w_{2,1}^t & w_{2,2}^t & \cdots & w_{2,m}^t \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1}^t & w_{n,2}^t & \cdots & w_{n,m}^t \end{pmatrix}$$

Thus, each participant $p \in P$ from m depots ($|P| = m$) maintains a weighing vector for each of n customers ($|N| = n$). These vectors determine the bias of participant p in serving customer nodes. The generator template helps determining state variable u_{ip}^t. If $u_{ip}^t = d_i$ then $w_{i,p}^t = \max_{p' \in P}(w_{i,p'}^t)$, which is maximum among the weights of all participants for customer node i. Otherwise, $u_{ip}^t = 0$. We assume that the $\max(\dots)$ function generates an unambiguous customer distribution for all depots.

Initial Distribution Policy: Initially, a fairly good distribution policy is needed to start with a favorable customer allocation. The policy should unambiguously distribute customer nodes to the depots such that at least more than 50% customer allocation is correct. To this end, we propose assigning customers to the depot that can serve with lowest cost if served directly from the depot. To calculate the weight vector we apply a *positive voting* policy. In this policy, first, the weight is determined proportionately in terms of distance to all the depots. Second, the values are normalized such that the sum of weights across each row is 1, i.e. $\sum_{p \in P} w_{i,p}^t = 1$. Finally, we apply a positive voting whereby the weight of the selected decision maker (for a customer node) reaches more than $\frac{1}{2}$. The bias of other decision makers are thereby readjusted proportionally. The following examples explain the concept.

Example 1: Let the distances between customer node i and depots P1, P2 and P3 be 3, 4 and 5 units respectively. Then, their actual normalized approximate weights are

0.43 ($\frac{1/3}{1/3+1/4+1/5}$), 0.32 and 0.25 respectively. As per the policy, P1's weight is increased to 0.51. The weights of P2 and P3 proportionally decrease to to 0.275 and 0.215 respectively.

Example 2: Let the distances between customer node i and depots P1, P2 and P3 be 2, 4 and 4 units respectively. Then, their actual normalized approximate weights are 0.5, 0.25 and 0.25 respectively. As per the policy, P1's weight increases to 0.51. The weights of P2 and P3 proportionally decrease to 0.245 and 0.245 respectively.

Example 3: Let the distances between customer node i and depots P1, P2 and P3 be 3, 3 and 4 units respectively. Then, their actual normalized approximate weights are 0.363, 0.363 and 0.273 respectively. As per the policy, weight of either P1 or P2 is increased to 0.51 while the weights of the others are proportionally reduced to 0.28 and 0.21 respectively.

After initial distribution of customers to depots, the multi-round solution generation process begins. As explained before, the distribution of a customer needs confirmation of more than 50% as implemented in max function. The proposed policy generates a unique initial distribution of weights resulting to an unambiguous decision making over customer selection. Thus, when depot p serves a customer node i then we call i as dominated by depot p. This assures no other depot is currently dominating the same node. Later, we explain how this policy helps the evolutionary learning based solution search.

In what follows, we discuss a distributed collaborative setup with an implementable evolutionary learning procedure to divide the main problem into sub-problems using the policy based on initialized generator template of weight vectors.

4.3.3 Proposed Approach

Figure 4.1 depicts a partially distributed setup where collaborative participants may join from their respective depots with individual capacity of commodity delivery. The setup performs two main distributed operations, namely: choosing customer nodes for commodity delivery and computation of transportation cost. However, aggregation of the total cost of solution and decision making for the update of weight vectors at the depots side are performed centrally. Although, it is possible to implement a fully distributed setup without any central authority to evaluate the total commodity delivery cost along with the decision

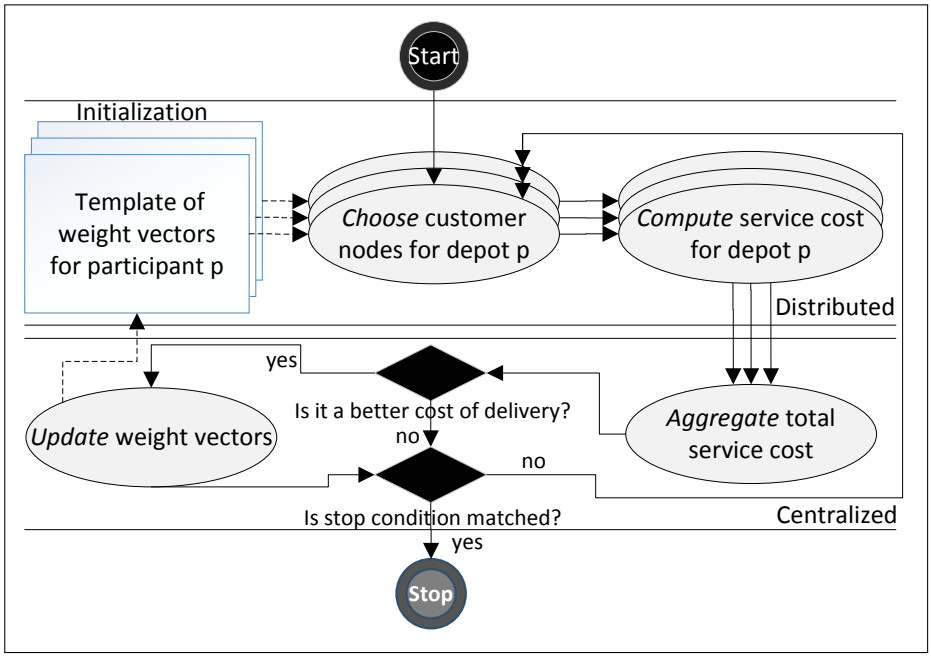


Figure 4.1: Collaborative solution generation for multi-depot vehicle routing problems

making for the update of weight vectors, it needs extensive peer-to-peer communication among depots. This, in turn, reduces the efficiency of the system. In what follows, we discuss a four-step task sharing based collaborative optimization (*see* Section 2.1.3 for details) in this setup. To simplify the discussion, we elaborate the task decomposition as the last step.

Task Allocation: In the aforementioned setup, at round t, depot p's likelihood to serve customer i is represented by its weight $w_{i,p}^t$ in the template W_p^t as maintained at participant p. A decision maker, who is associated to a depot, selects a customer node if and only if its conclusive dominance was previously established over the respective node. Then, at each round, every decision maker decides over the subset of customers under his/her dominance whether to keep them under their dominance or not. Each decision is made using a pseudo random function $f_{random}([w_{i,1}^t, w_{i,2}^t, \ldots, w_{i,m}^t])$ at participant p if node i is under p's dominance. The output of the f_{random} function is a depot that is chosen in a biased random fashion. The bias for a particular depot is generated using the input weight vector. At round t, if the output indicates that the depot p is itself, then the

customer node is required to be served by p. If the output is another depot p', then at that particular round, it will be served by p'. Depots need to unambiguously ensure the responsibility of serving a customer node to one and only one depot.

The aforementioned sub-problem design is important for four main reasons. First, it divides the original problem instance into multiple sub-problems. Second, it assures that at any round only one depot decides on the commodity delivery to a customer. Third, it also ensures that only one depot remains responsible to serve a customer. Finally but most importantly, f_{random} function offers an error mitigation strategy for near-optimal solution search. Usually, during a multi-round solution generation, dominance of a particular depot generally increases over a customer node in each round which helps the convergence of the heuristic/meta-heuristic solution search. However, it may also lead the solution search to a local optimal solution. A fairly designed random function offers a lower probability for a customer node to be served by other depot(s) than its dominating depot. It allows the solution search to reassess the potential of slightly different customer assignment possibilities which may lead the search process toward global optimal solution.

However, a task allocation does not guarantee existence of a solution since a participant is not aware of the capacity of others by design. Eq. (4.10) reflects this uncertainty in risk allocation. Therefore, multiple randomization (calling f_{random} function) may be required to reach a distribution where capacity of participants are enough to compute a solution. Thus, we handle uncertainty in risk allocation. Once the participants agree to start computing a solution, Eq. (4.8) is satisfied.

Task Accomplishment: The task of computing routes is performed in a distributed setting by individual participants at every round. The cost computation of individual SD-VRP instance is performed by applying heuristic technique followed by a meta-heuristic improvement. We use the heuristics defined in Algorithm 1 and associated meta-heuristic techniques to compute the serving cost of each individual depot for its respectively assigned customers (no commonly shared problem instance is involved among depots). Every individually computed solution must satisfy Eq.s (4.4)-(4.7) of the distributed problem model.

Result Synthesis: Depots share their computed cost by communicating to a central entity in order to calculate the total cost of service. If the total cost is found better than

the previously best found overall cost of the same MDVRP instance then it represents a new minimum solution cost for the original problem instance. Then, an *Update* request is sent to each participating depot to consider this assignment of customer nodes to be part of their decision making for the next rounds.

Task Decomposition: Each depot individually adjusts weights in its weight vectors while learning the situation with respect to the overall outcome of customer assignment. As detailed in Section 4.3.1, boosting potentially increases the likelihood of choosing an optimal allocation. During successive customer allocation round, customers are gradually allocated more appropriately toward a near-optimal solution. Thus, the interpretable function $f_i(u_{ip}^t)$ dominantly contributes in determining the response of serving a customer node by depot p. While certain allocations of customer nodes easily reflect their dominating depots, others may keep changing their dominating depots. With the progress of the multi-round allocation procedure, the undelying boosting mechanism handles these customers with increasingly less options of depots for final allocation. Thus, in the collaborative MDVRP model, we handle Eq. (4.11) on the decision making for customer nodes.

In the solution search, task decomposition is critical for the convergence and success of the proposed approach. We propose a LogitBoost based mechanism [81] to form a strong additive learner model from the elitist solutions to update the weights. This procedure is unique from two relevant aspects.

- *Elitist Solutions*: Elitist solutions are determined using a *gap* value with respect to the currently best found solution (denoted as: *CurrentBest*) as reference. Let s_{min}^t denote the current best solution then the subset of solutions having cost within a specified maximum gap (*gap*) are considered as elitist solution pool. For example, with a 10% gap, all solutions are considered elitist where the solution cost is not more than $1.1 \times s_{min}^t$. It is important to note that, an elitist solution is chosen best on overall cost instead of the contribution of a depot p in this particular solution, denoted as s_{pj}. This requires collaborative decision making since the participants work in the best interest of the overall solution search.

- *Dominance Selection*: In each round, a sorted array of elitist solutions in descending order of solution cost is used to determine the dominance of a depot. First, a binary function $b_j(i,p)$ is used to identify if customer node i is allocated to depot p in the elitist solution j. Second, a function $rank_j(p)$ uniquely determines the position of solution j at depot p. The *CurrentBest* solution always has ranking 1. Then, the weight $w_{i,p}^t$ can be calculated as $\sum_{j=1}^{r} \frac{1}{2^{rank_j(p)}} \times b_j(i,p)$. r denotes total number of elitist solutions.

As we see, the dominance selection uses a polynomial series ($\sum_{j=1}^{\infty} \frac{1}{2^j} = 1$) which assures updated weight $w_{i,p}^t$ for a customer node i will never reach 1 for a depot p, in practice. It also affirms that there is always a chance for a customer node to be served by another depot using f_{random} function even when that depot is not dominating the customer node. The update of weight is based on an estimation with respect to which node allocation produces a better solution, at the end of each round. If node i is served by a depot p in most of the competitive solutions with lower routing cost, it is most likely to be served by p. However, even if customer i is served by depot p only in the *CurrentBest* solution, still customer i is conclusively under the dominance p (assuming more than 2 depots serve customers).

Thus, we define a weight adjustment function $adjust_p(\dots)$ to update weights of customer nodes in depot p between successive rounds. Two additional implementation-specific thresholds are used, namely $maxconf$ and $minconf$. They are user-chosen and they restrict updating weight higher or lower than these chosen values to give every customer node a fair chance to be served by different depots. $adjust_p(\dots)$ can be described as follows:

$$w_{i,j}^{t+1} = \begin{cases} \rho = Max(Min(\sum_{j=1}^{r} \frac{1}{2^{rank_j(p)}} \times b_j(i,p), maxconf), minconf) & j = p \\ \frac{w_{i,j}^t \times (1-\rho)}{1-w_{i,p}^t} & \text{otherwise} \end{cases}$$

4.3.4 Algorithm Design

Algorithm 2 presents a collaborative distributed solution generation technique on problem data (depots, customers, initial allocation policy known to all depots) and input parameters. The input parameters $maxham$ and $maxcap_p$ represent the maximum number

Algorithm 2 Evolutionary Learning Procedure for depot p

Step 1: Local initialization at participant p
WeightMap $W^0_{n,m} \leftarrow \rho(N, P)$, $CurrentBest \leftarrow \infty$;$CurrAlloc_p \leftarrow \{\}$; $CostMap \leftarrow \{\}$;
Step 2: Local allocation at participant p
Initialize $cnt_p \leftarrow 0$; $CurrAlloc_p \leftarrow \{\}$; $o \leftarrow -1$; *Notify* all $(P \setminus \{p\})$ depots to start;
while $cnt_p \leq maxham$ **and** $capc(CurrAlloc_p) \leq maxcap_p$ **do**
  Randomly *choose* node i in set $\{1, \ldots, n\} \setminus CurrAlloc_p$
  if depot p *dominates* customer node i **then**
    while $(o = -1)$ **or** $\neg available(o)$ **do**
      $o \leftarrow f_{random}(w^t_{i,1}, w^t_{i,2}, \ldots, w^t_{i,m})$
    end while
    if $o \neq p$ **then**
      Send the customer node i to o; $cnt \leftarrow cnt + 1$;
    else
      Assign $\{i\}$ to $CurrAlloc_p$;
    end if
  end if
end while
Receive subset of customers N' sent to p; Assign N' to $CurrAlloc_p$;
$cnt_p \leftarrow cnt_p + |N'|$
if $(capc(CurrAlloc_p) > maxcap_p)$ **or** $(cnt_p > maxham)$ **then**
  Notify depot p cannot execute *Step 3*; go to *Step 2*;
end if
Synchronize that all depots can start *Step 3*, Otherwise, go to *Step 2*;
Step 3: Local cost calculation at participant p
if $CurrAlloc_p \notin CostMap$ **then**
  $cost_p \leftarrow solve_{heur.}(\ldots)$ for $CurrAlloc_p$;
else
  Get $cost_p$ from $CostMap$;
end if
Notify $cost_p$ to all other depots; Synchronize and *aggregate* $total \leftarrow \sum_{p \in P} cost_p$
Step 4: Local weight adjustment at participant p
if $CurrentBest = \infty$ **or** $total < getTotal(CurrentBest)$ **then**
  $CurrentBest \leftarrow total$;
  Add $\langle CurrAlloc_p, [cost_1, \ldots, cost_p, \ldots, cost_m] \rangle$ to $CostMap$
  for each customer node i in $CurrAlloc_p$ **do**
    Update weight using $adjust_p(\overrightarrow{w^t_p}, gap, CostMap, CurrentBest, maxconf, minconf)$
  end for
end if
Step 5: Central decision making
if $maxIte()$ **or** $isAcceptable(CurrentBest)$ **then** return $CurrentBest$;
else go to *Step 2*;
end if

of customer migrations allowed in a round by any depot and the maximum commodity carriage capacity of a depot respectively. The $minconf$ and $maxconf$ are confidence levels while gap helps in choosing elitist solutions. The algorithm employs the following data structures:

- $WeightMap$: An associative array holding weight vectors in each depot node;
- $CostMap$: Ordered associative array holding cost and related allocations;
- $CurrentBest$: Structure holding the current best cost and its customer allocation;
- $CurrentAlloc_p$: Structure holding current allocation of customer nodes for depot p;

$CostMap$ and $CurentBest$ may be stored at a shared location to reduce communication load among depots. The following helper functions are used:

- $capc(CurrentAlloc_p)$ computes required capacity to serve nodes in $CurrentAlloc$;
- $available(p)$ determines if a depot is available to serve more customer nodes.
- f_{random} denotes a pseudo-random function to select a node given an input weight vector.
- $solve_{heur}(\dots)$ denotes the invocation of the heuristic technique to produce a solution for a given customer allocation.
- $adjust_p(\dots)$ reassigns new weight for the customer nodes for a depot p.
- $maxIte()$ determines if the maximum round of iterations is reached.
- $isAcceptable(CurrentBest)$ determines if the current allocation is "significantly" better than previously found solutions to be accepted.

4.3.5 Case Study

In what follows, we study the modified-*E016-03m* problem by applying the proposed distributed learning technique. We assume the existence of three decision makers on three depots of the problem instance. Since the $maxham$ is 6, at most 6 nodes under dominance

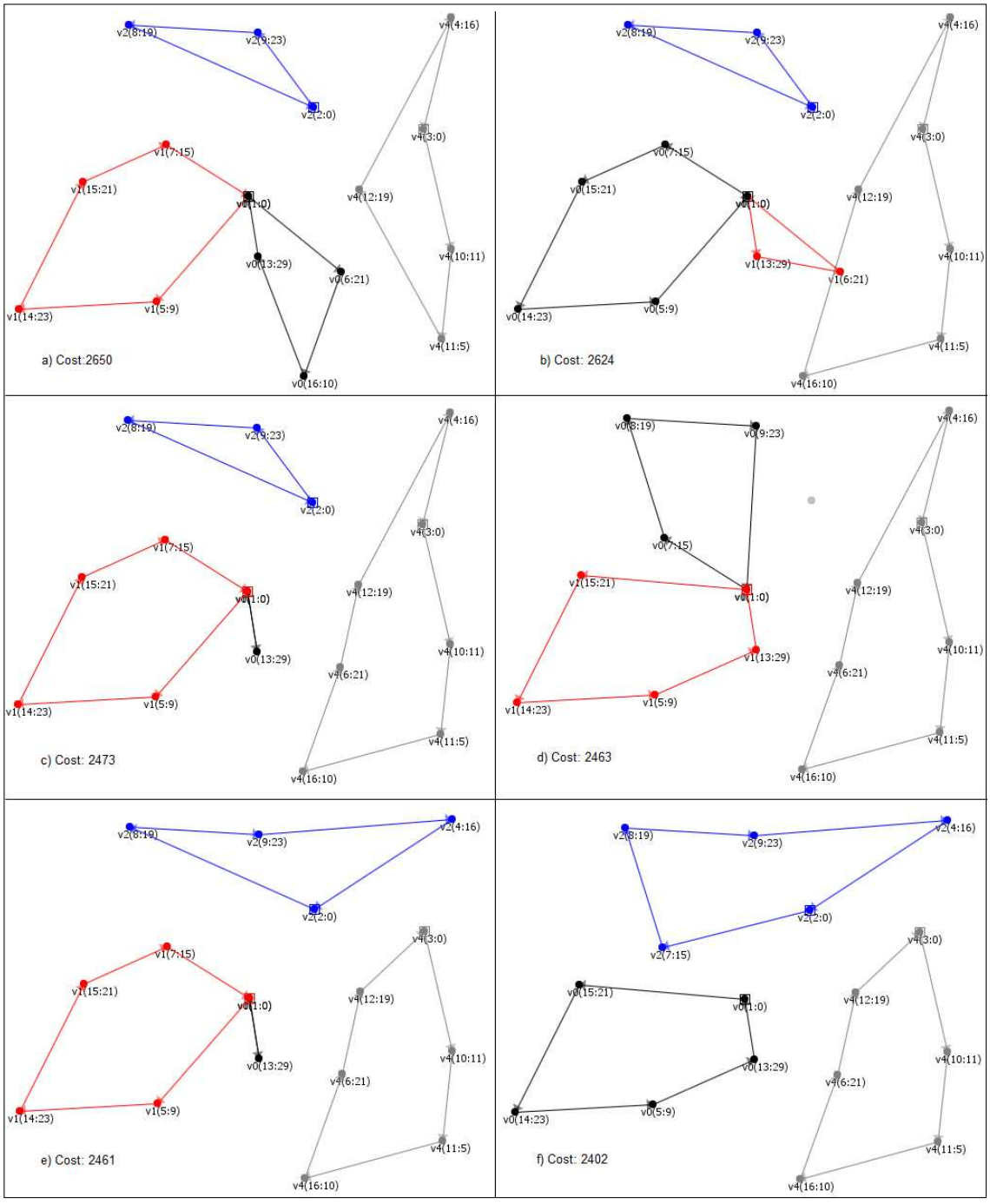


Figure 4.2: Learning-based distributed solution generation

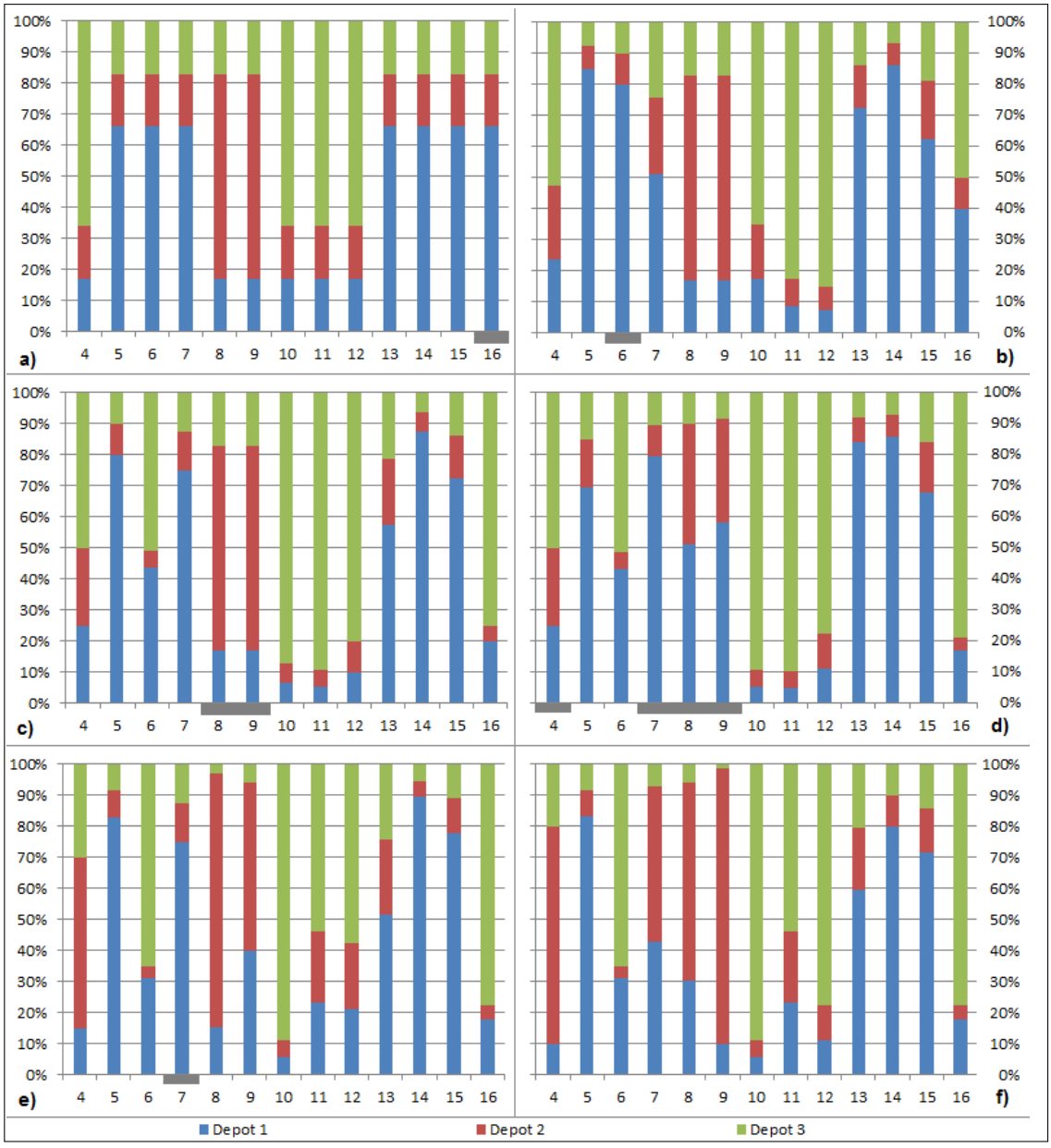


Figure 4.3: Changes in depot's influence in solution generation

of a depot can be exchanged with other depots. The minimum and the maximum confidence levels are 0.015 and 0.985. Therefore, a customer node always will have at least 0.015 chance to be served by other depot. Based on demand and vehicle capacities, the depots exchange their responsibilities of serving customers. While different executions may yield slightly different solutions due to weighted randomness, the technique observes faster converge with six intermediary steps. We start with a policy by allocating customer to the least distant depot. The actual normalized approximate weight for each depot on a customer node is then calculated as discussed in Section 4.3.2.

Figure 4.2 depicts multi-round solution generation procedure based on evolutionary learning. As the customers and their demands are progressively allocated to the appropriate depot, we can see successful cost lowering in this approach. The different cost values are presented in the six sub-figures. In this case, the initial policy-led distribution of nodes generates an overall cost of routing as 2650. The evolutionary learning procedure helps in lowering the routing cost to 2402. The iterative update of the weight vector for the majority of customer nodes retains the dominance of the same depot. However, interestingly, the reassignment of weight for the customer nodes 16 and 6 changes from their early assignment in the beginning from depots 1 to 3.

Figure 4.3 shows node allocation evolution dynamics and related weight adjustment. Three depots are shown in blue (depot 1), red (depot 2) and green (depot 3) colors. In this allocation, a total of 392 solutions with different customer allocations have been tested during the process run. Among them, there are six adjustment steps, as depicted, those involve increasingly better allocations with reduced solution cost. The gray bands at the bottom of the column show the node migration from a depot to another in successive better allocations. The node from one depot migrates to the other with change of dominance in the next round. The summation of normalized weight of all depots over a customer node is 100% with a share of the dominating depot more than 50%. At each depicted step, for every node n in x-axis, we have the weight vector depicted in y-axis such that the sum is always 100%.

4.4 Cooperative Solution Generation: Active Negotiation

Often time, vehicle fleets are controlled by self-interested rational decision makers over a shared transportation network. In such business environment, distributed platforms for solving multi-depot vehicle routing problem may find non-collaborative decision makers. In this case, a cooperative approach is more suitable than collaborative solution generation approaches. In this section, we discuss a distributed mechanism to tackle multi-depot vehicle routing problem among cooperative self-interested decision makers. We apply game theory to jointly decide customer assignments.

The cooperative solution generation starts with P participating decision makers, each of which has its own vehicle fleet. We assume that these rational decision makers are operating over a complete graph representing a common transport network similar to previous passive learning technique. The routing cost for each edge of the graph is therefore known to all decision makers. The customer demand is also a shared information. However, these self-interested decision makers do not directly disclose their vehicle information (number of vehicles and capacity of each vehicle) and the actual cost of serving a set of customers in order to enjoy business advantage.

In this context, we propose a distributed algorithm between cooperative depots and customers, where the final outcome is the (near) optimal customer assignment. The joint execution of this algorithm helps the participants to cooperate in order to assign every customer to a depot which offers to serve the customer demand at the lowest cost. The procedure generates the final outcome in multiple rounds. In each round, the interaction among the participating depots, for the negotiation of customer assignment (comparing offered service cost) can be modeled as a game. The rules of the game are devised such that the outcome minimizes the overall routing cost. In this context, the proposed approach uses a reverse *Vickrey auction* [134] in each round to design a mechanism for executing the "game" of assigning customers to appropriate depots. For each customer assignment, a payment is given for the service of the depot that performs the commodity delivery. The payment offers an incentive for fair cost offering by participating depots. Intuitively, a rational and cooperative decision maker prefers selecting a subset of customers (part of

an outcome) that maximizes its profit, i.e. the difference between its received payment and its serving cost. The payment, as designed in Vickrey auction, assures that every rational decision maker reveals its true cost of serving a customer at every round of the game. The solution generation procedure is inspired from several previous research efforts [144, 161, 162, 210]. While the approach has its own advantages and limitations, it can potentially lead to a near-optimal routing solution while serving all customers.

The approach is particularly useful in *small-world* transport networks [128, 238]. The latter is generally characterized by random connectivity with specific properties such as a short average path length, large clustering coefficient [25] and an unfavorable topology for hub formation [66]. Since every node is generally connected to every other node through a short path (in terms of average node-to-node distance), no participant can make strong assumptions about the offerings of other participants based on topological considerations. Thus, we expect the proposed planning approach to be suitable to urban logistics distribution in pursuit of timely and effective operations.

4.4.1 Game of Customer Selection

We introduce a game $\mathcal{G}$ among P participants. Each participant p possesses $\mathcal{S}_p$ strategies, based on their capacity of commodity delivery and a utility function φ_p. $\mathcal{S}_p$ is known as strategy space of participant p. Let $\mathbf{S} = \langle \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_p, \ldots, \mathcal{S}_P \rangle$ denote a tuple of all possible strategies from P participants. In our context, each participant's strategy actually represents its choices to serve customers as previously denoted using decision variables: $u_p^t = [u_{1p}^t, u_{2p}^t, \ldots, u_{np}^t]$ in the decentralized model (*see.* Section 4.2.3). Similar to evolutionary learning, we may express u_{ip}^t as binary variable. In every round of the game, each participant independently plays its strategy. Thus, we are interested in a *profile* of strategies ($\mathbf{u}^t$) that represents strategies of all participants at round t denoted by a tuple $\langle u_1^t, u_2^t, \ldots, u_m^t \rangle$. In this context, only a subset of these profiles, each of which respects all the given constraints can result into an outcome where the latter represents a valid assignment of all customers. As such, each member of this subset may lead toward a solution to our problem. A game may have several outcomes, denoted by set O. Thus, an outcome $o \in O$ indicates values of the decision variables after evaluating different strategies from

$\mathbf{u}^t$. Let us assume that there exists a *solution concept* $\mathcal{H}$ that maps $\mathcal{G}$ using $\mathcal{H}(\mathcal{G})$ to a set of valid customer assignments. Then, each member of this set denotes a particular assignment of all customers to various participating depots. Combining a set of routes heuristically calculated from the assigned customers of each depot determines a solution of the multi-depot vehicle routing problem. Now, since each member of this set represents a full assignment of all customers then it also a member of outcome O. In fact, each of them also denotes an equilibrium of the game $\mathcal{G}$.

In order to reach a solution, each player plays a particular strategy. The utility function φ_p provides a numerical measure to evaluate a strategy u_p^t against others (u_{-p}^t). To solve the game in a cooperative setting, we rely on a specific equilibrium, namely Dominant Strategy Equilibrium (DSE). In DSE, each decision maker p plays a strategy which offers him/her the maximum utility with respect to all other available strategies in $\mathcal{S}_p$. We represent this strategy as $u_p^{t,dse}$ while the rest of the strategies in $\mathcal{S}_p$ are denoted by $u_{-p}^{t,dse}$. Therefore, mathematically, for all strategies in $\mathbf{u}^t$, $\varphi_p(u_{-p}^{t,dse}, u_p^{t,dse}) \geq \varphi_p(u_{-p}^t, u_p^t)$, where $\mathcal{S}_p = u_{-p}^t \cup \{u_p^t\}$. In DSE, the utility of a player decreases if the player deviates from the equilibrium, irrespective of the strategies of other players. Thus, every rational participant is expected to stick to its strategy $u_p^{t,dse}$ if the implementation of the game is truthful. The latter condition is also known as *strategyproof*.

The implementation of a game, also termed as mechanism design, is performed by enforcing a set of rules. We apply a synthesis technique where we first specify our desired outcome (find a solution of minimum routing cost) and then start designing a set of rules accordingly. Therefore, these rules are intended to reveal the true cost of service from each player. Therefore, a mechanism is a pair $\langle \mathbf{S}, g \rangle$, where $\mathbf{S}$ is constructed over the set of strategies for all players and $g : \mathbf{S} \rightarrow O$ is a function that maps strategy profiles to outcomes.

Meanwhile, each participant implements φ_p function to devise own strategy to contribute in an outcome. Let us assume that there exists a social choice rule f that operates over a tuple $\varphi := \langle \varphi_1, \varphi_2, \ldots, \varphi_m \rangle$ to produce a set of outcomes. Then, for a finite set of utility functions Φ, function f maps Φ as $f : \Phi \rightarrow 2^O$ where $\varphi \in \Phi$.

Mechanism design implements function f in the the game. Mechanism $\langle \mathbf{S}, g \rangle$ is said

to $\mathcal{H}$ – *implement* the social choice rule f if for all utility functions, $f(\varphi) \subseteq g(\mathcal{H}(\mathcal{G}))$. More precisely, if $g(\mathcal{H}(\mathcal{G}))$ denotes our desired solutions then f is designed such that $f(\varphi)$ generates a subset of them. If $f(\varphi) = g(\mathcal{H}(\mathcal{G}))$, the mechanism is strongly implementing f. If the strategy space of $\mathbf{S}$ is same to that of Φ, the mechanism can ask each player to report its individual preference. Such a mechanism is widely known as direct revelation mechanism. However, a truthful $\mathcal{H}$-implementation of f in game environment also requires $\mathbf{u}^t$ as dominant game strategy for a given set of utilities.

From the revelation principle presented by David C. Parkes [186], if f is a DSE-implementable choice rule in the game, f is truthfully DSE-implementable. Therefore, in game $\mathcal{G}$, players play φ as their dominant strategy where $g(\varphi) \in f(\varphi)$. An important special case in this game lies where each participant's utility has a quasi-linear form.

As we have mentioned, the computation of routing cost for a participant depends on private information such as participant's vehicle capacity, number of available vehicles, heuristic route generation algorithm, etc. Let us represent all this private information through a private type θ_p. Thus, an outcome can be denoted as $o(\theta)$ where $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$. The $o(\theta)$ is computed over all outcomes from every participant. Then, using quasi-linear form, we may express $\varphi_p(o, \theta_p) = \vartheta_p(o, \theta_p) + \lambda_p$. Here $\vartheta_p(o, \theta_p)$ represents participant p's *evaluation* of a certain outcome and λ_p is a payment to p. Vickery-Clarke-Groves (VCG) mechanism provides a truthful DSE-implementation for a social choice function maximizing the summation of agent valuations where the payment for participant p has the form [152]:

$$\lambda_p = \left[\sum_{q \neq p} \vartheta_q(o(\theta), \theta_q) \right] + h_p(\theta_{-p}) \tag{4.17}$$

where $h_p(\theta_{-p})$ is an arbitrary function of $\theta_{-p} = (\theta_1, \ldots, \theta_{p-1}, \theta_{p+1}, \ldots, \theta_m)$. VCG puts two main constraints. First, there should be atleast two participants in each game. Second, the payment to a participant should be independent of its evaluation for the outcome.

4.4.2 Mechanism Implementation

In a *small-world* transport network, an allocation of each customer to an appropriate depot can be arranged as a game where each participant, in control of a depot, simultaneously presents its serving cost. The capacity of each vehicle in the fleet of vehicles under each depot is a private information. Therefore, no participant can guess the offers of another participant since they cannot accurately predict the routing cost from other depots due to small-world characteristics of the transport network. In a general setting, at any round, a number of customers receive offers from the participating depots to allocate a customer in their own route(s). The task of serving a customer is allocated to the depot that can serve the customer with lowest cost. Without a thoughtful design of payments, depots could be tempted to reveal "untrue" cost to gain personal advantage. In order to assure truthful revelation, we propose the VCG mechanism to "pay" the winning depot at each round. VCG mechanism secures a strategyproof implementation of the game at each round. Furthermore, rational participants comply with this mechanism since a larger number of assigned customers is more likely generate higher payment to a depot especially if the routes can be designed using customers close to each other.

In a multi-depot vehicle routing problem, θ_p affects the generation of the routes and evaluation of the routing cost of p for serving a chosen subset of customers. A locally executing heuristic algorithm may produce a set of routes and determine the routing cost. Given a set of all private types θ, an allocation function $a(p, \theta)$ allocates every customer node to a depot along with relevant payment per customer node. Such an allocation is designed over the submitted offer of depot p to serve a subset of customer nodes which maximizes depot's utility. For a participant p, $c_p(a(p, \theta), \theta_p)$ determines the routing cost (using heuristic algorithm) from the allocated customer nodes to p. If the payment is determined based on VCG mechanism, it ensures true revelation of the routing cost from the participating decision makers. Using quasi-linear form of the utility, for each participant p with type θ_p the payment can determined as:

$$\varphi_p(o, \theta_p) = \lambda_p - c_p(a(p, \theta), \theta_p) \tag{4.18}$$

where $c_p(a(p,\theta),\theta_p)$ replaces $\vartheta_p(o(\theta),\theta_p)$. Eq. (4.18) denotes that the utility of participant p is maximum when it performs commodity delivery to its assigned customers with minimum routing cost. This creates the premise for cooperative minimization of routing cost for MDVRP instances. An effective determination of payment λ_p may allow the game execution to reach the globally near-optimal routing cost. Thus, in order to implement a social choice function minimizing overall routing cost, we use the VCG mechanism where an depot's cost is given by $c_p(o,\theta_p) = -\vartheta_p(o,\theta_p)$. Hence, payment for the depot p is:

$$\lambda_p = \sum_{i \in P \setminus \{p\}} c_i(o^*(\theta),\theta_i) - \sum_{i \in P \setminus \{p\}} c_i(o^*(\theta_{-p}),\theta_i) \tag{4.19}$$

In equation (4.19), $o^*(.)$ is the outcome minimizing the total routing cost of all depots. The payment to a participant p reflects the difference in total routing cost of other participants in p's presence and in p's absence. Thus, the payment is independent of p's evaluation of its routing cost to serve a subset of customer nodes.

4.4.3 Proposed Approach

To solve MDVRP instances, we propose a distributed setting of the game for customer selection by participating decision makers such that the total routing cost is minimum. Such a game has three main challenges. First, it requires resolving the allocation function $a(p,\theta)$ in a decentralized setting to allocate customers without compromising private information. Second, it requires distributed determination of the appropriate payment (following VCG mechanism) to a depot for winning a subset of customers by revealing the lowest serving cost. Finally, in order to understand the dominant strategy, each participant should find out and evaluate (possibly using heuristics) various profiles of strategy while respecting his/her total capacity of commodity delivery. Each of this profile involves computing routes. This involves handling huge computation load at every round of the game.

Thus, we propose to modify the aforementioned generic concept as follows.

- *Game Setup*: We address the allocation by including customer nodes in the game execution. Let $a_i(p,\theta)$ identify the allocation of customer node i to depot p. Every

depot locally computes its cost of serving a set of unassigned customer nodes at each round. Depot p submits its offer to a customer node it wants to serve. The offer consists of the cost of serving the customer node i in particular, as denoted by $\nu(i,p)$. Depot p estimates $\nu(i,p)$ based on its total cost of serving the subset of customers $c_p(a(p,\theta),\theta_p)$ such that $\sum_{i \in N} a_i(p,\theta) \times \nu(i,p) = c_p(a(p,\theta),\theta_p)$. No strong assumptions can be made by any depot with respect to the offering of the other depots due to the *small-world* topological considerations as mentioned before. Each customer chooses a depot that offers serving the customer node with the minimum cost. Thus, the allocation takes place following the choice of the customers.

- *Payment Handling*: We address the distributed determination of payment as follows. Each customer chooses the depot p that offers the minimum cost and pays p the second minimum cost for service, i.e., $\min_{i \in P \setminus \{p\}} \nu(i,p)$. The customer pays only depot p which offers the lowest cost of service. This mechanism design is strategyproof. The second minimum cost is no less than the cost of service for the selected depot and the additional payment does not depend on the cost revealed by the selected depot. Finally, it respects Eq. (4.19), since in absence of the selected depot, the customer node would have paid the second minimum cost to be served.

- *Dominant Strategy Finding*: Finding dominant strategy is difficult in this setup of the game. Since, the routing cost of each edge over the transport network is known to all participants, it is only possible for a depot to approximately calculate the routing cost of serving a subset of customers by another depot. However, with private information, such as vehicle capacity, number of vehicles, etc. derivation of an accurate routing cost for a depot is not possible by another depot. So, we simplify the strategy finding at the game execution using a predefined policy.

The aforementioned approach has intrinsic tractability challenges in finding dominant strategy. Thus, we propose a multi-round game execution as follows.

- *Commitment Binding*: The distributed setting of active negotiation forces a depot to commit service to its assigned customers. This contrasts to a more cooperative setting

where a depot might leave a customer to other depot if there is an individual/overall cost benefit, by exchanging service to customers later in the game.

- *Customer Ordering*: It is essential to assign the right customers to the right depot. As the exact algorithm would be computationally expensive, we address this issue by auctioning the customers in a policy based ordering. We have experimented three approximation policies based on different distance/cost criteria among the customer and the depots: *a)* no ordering: random choice of customers *b)* outer edge: sorting the customers in a descending order of their largest $\frac{distance}{cost}$ ratio from any of the depots and *c)* depot bias: sorting the customers in a descending order of their evaluation of vicinity to all the depots. The evaluation of vicinity is taken as the projection of a point on the line that marks equal vicinity in a multi-dimensional space where each orthogonal dimension marks a depot in the problem. For example, a point (a, b) that holds distances a and b from two depots respectively has the vicinity $|\frac{a-b}{\sqrt{2}}|$. Therefore, in this specific implementation of the approach, we design certain ordering, that allows one customer at a time receives the offers of the depots. This contrasts to the general notion where every customer node is auctioned at every round.

  this is the general case, however it may be important to hint that when using an auction based

- *Insertion Cost Calculation*: This is challenging for medium and large VRP instances as it is often intractable to calculate the best insertion cost of serving a new customer node in bounded memory and time. Therefore, we use a near-optimal heuristic cost computation [210] (*see* Section 3.3.1) which is fast and broad in scope (offering parallelism and being free from the limitations such as triangle inequality satisfaction, angle/curvature issues). It uses multi-point stochastic insertion cost gradient descent where solutions are assembled from connecting fragments. The multi-point aspect deals with fragment construction by inserting unserved customer nodes in multiple points of selected vehicle tours. The stochastic aspect is dealt with a seed based pseudo-randomized vehicle selection for node visiting. The insertion cost gradient descent relates to exploring lower cost fragments before higher cost ones.

The aforementioned game execution may generate near-optimal distributed solutions of MDVRP instances by enforcing VCG mechanism with reasonable memory footprint and computation time. An assumption has been made that at least two depots send valid competitive offers to serve a customer demand. It also requires depots to have enough remaining capacity to serve no less than the customer demand in order to participate in an auction. Distributed setup exhibits few additional implementation challenges:

- If two or more depots submit the same amount of offer to a customer, it will require an additional decision to allocate the customer node to one of these depots.
- In every round, each participating depot has to (near) optimally solve a VRP (or SDVRP) instance with previously allocated customers along with the new customer subjected to the auction according to the policy.
- There is no guarantee that the a solution can be always found after certain steps while auctioning according to a policy.
- The setup does not allow shared service to a customer from two different depots.

Algorithm 3 Distributed Algorithm for Depot p's Offer

Select an auction policy ρ.
Initialize: $N^* \leftarrow \{\}$ as empty set.
Initialize each $l_p^k \leftarrow C_p^k$ as initial capacity.
Use ρ to compute ordered list $N[i...n]$ for all customers in N
for $i = 1, \ldots, n$ **do**
  if $\sum_{k_p \epsilon K_p} l_p^k \geq d_i$ **then**
    Assign serving $cost \leftarrow Heur(\ldots)$ with nodes $N^* \cup \{N[i]\}$
    Send offer of additional serving $\nu(i,p)$ as *request*
    Initialize: $reply \leftarrow -1$
    while $reply \leq 0$ **do**
      Wait for *utility* value update. Assign $reply \leftarrow utility$
      if $reply > 0$ **then**
        $N^* \leftarrow N^* \cup \{N[i]\}$
        Change capacity l_p^k for each assigned vehicle to $l_p^k \leftarrow l_p^k - \gamma_p^k$ such that $\sum_{k_p \epsilon K_p} \gamma_p^k = d_i$
      end if
    end while
  end if
end for

The proposed setup may handle distributed computation for every depot relative to others. However, the scope of auction makes the approach less suitable in certain contexts such as rescue missions. In these cases, customers are often not in a position to actively take part in the game and "pay" for the service. Then, same game can be executed among the depots where the depots can cooperatively decide on serving customer nodes and different forms of payment such as reputation. Secure Multi-party Communication (SMC) among the depots can be performed to select customers without involving customers in the loop [92]. The SMC respects privacy to identify better offer without revealing one depot's offer to another. Secure comparison technique such as Yao's protocol [250] can be used. Therefore, such implementation of our proposed approach can be complex.

4.4.4 Algorithm Design

In this section, we present two algorithms to be executed at the participating depots and the customers. Algorithm 3 elaborates a multi-round cooperative game execution from the perspective of each depot. Given an auction policy ρ, each depot, having enough capacity, participates in carefully assigning new customer within the set of already assigned customers and determine the routes using heuristic/meta-heuristic methods (Step 7) to keep their offered cost $\nu(i, p)$ to the customer i at minimum as per the heuristic cost calculation (Step 8). Customer may reply 0 or a non-zero integer. The latter is sent only to the winning depot (Step 13-16) and represents the utility (which is the second minimum cost of service) that covers depot p's cost of service and payment. The depot is thus committed to serve the customer (Step 14) and subsequently its total available capacity of commodity delivery reduces by the amount of customer demand (Step 15).

Algorithm 4 chooses the depot with lowest offer of service for each customer. It keeps track of the lowest and second lowest offers (Step 11). It offers the second lowest offer of serving customer node to the depot that provides the lowest offer to serve the customer (Step 17-21). Both algorithms stop after evaluating all customer demands and assigning them appropriately. In this cooperative setting, the convergence of the solution search procedure toward near-optimal solution depends partially on the chosen policy.

Algorithm 4 Distributed Algorithm for Customer i's Choice

```
1: Initialize all depots P[i...m] available to the customers
2: min ← ∞; secmin ← ∞; seldpt ← −1
3: for p = 1, ..., m do
4:   Initialize: ν(i,p) ← −1
5:   while ν(i,p) < 0 do
6:     Wait for ν(i,p) value update.
7:     if ν(i,p) > 0 then
8:       if ν(i,p) < min then
9:         secmin ← min; min ← ν(i,p); seldpt ← p
10:      end if
11:    end if
12:  end while
13: end for
14: for p = 1, ..., m do
15:   utility ← 0
16:   if p = seldpt then
17:     utility ← secmin
18:   end if
19:   Send utility to p
20: end for
```

4.4.5 Case Study

Figure 4.4 shows a solution obtained using the proposed approach on the running example with the predefined configuration of the transport network and vehicle capacity. We tested three policies of customer orderings where, in each round, the cost offered by the depots to the auctioned customer is equivalent to its heuristic insertion cost. More precisely, it is the difference of the depot's routing cost after inserting the customer to its routes with the depot's current routing cost. Each customer is assigned to the depot that offers lowest cost on that round with the payment of second lowest offer.

Table 4.1: Multi-round customer allocation and heuristic cost during problem solving

P\N	14,	15,	5,	16,	8,	11,	4,	7,	13,	10,	6,	9,	12
P1 (Offer)	584,	73,	30,	264,	299,	194,	642,	228,	19,	68,	34,	n/a,	n/a
(Utility)	838,	538,	622,	710,	456,	562,	0,	336,	418,	320,	418,	0,	0
P2 (Offer)	838,	538,	622,	720,	456,	680,	384,	336,	418,	484,	444,	234,	241
(Utility)	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	280,	0
P3 (Offer)	1002,	754,	740,	710,	698,	562,	306,	482,	478,	320,	418,	280,	204
(Utility)	0,	0,	0,	0,	0,	0,	384,	0,	0,	0,	0,	0,	241

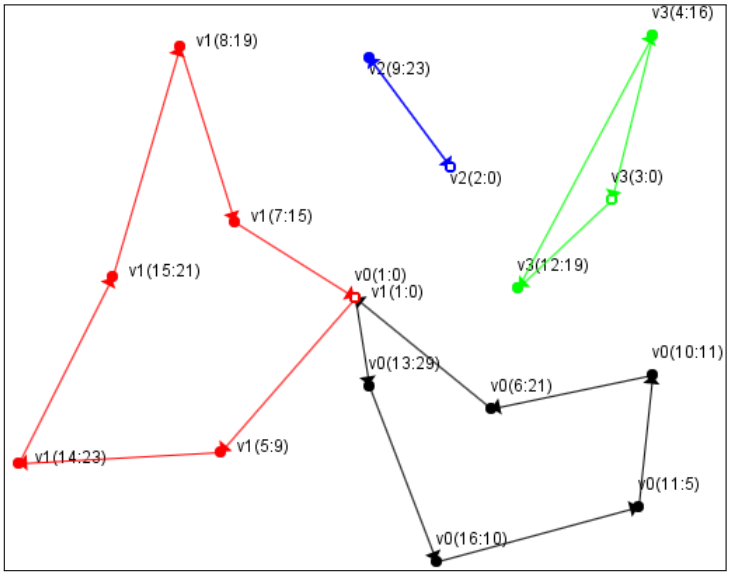


Figure 4.4: Negotiation-based solution generation on modified-*E016-03m* problem

Initially, all customers are unassigned. In the first policy, customers are chosen randomly and in each round, the depots are asked to submit their cost of serving to the customer. In the second policy, the customers are auctioned according to the descending order of the summation of their distances with respect to the depots. This assures that we start sending offers and assigning customers form the outer edge of the customer pool. The ordering in the example problem has been found as: $\langle 14, 15, 5, 16, 8, 11, 4, 7, 13, 10, 6, 9, 12 \rangle$. Finally, we also test a third policy where we order the customers according to the bias to any particular depot. The ordering is found as: $\langle 14, 5, 15, 4, 13, 7, 16, 8, 9, 6, 10, 11, 12 \rangle$. The resulting MDVRP routing costs of these orderings are between 2402 to 3005 in 200 different runs for random ordering, 2537 for outer edge ordering and 2683 for depot bias respectively. The second policy (outer edge customer ordering) provides better cost and the solution can be seen in Figure 4.4. The identified vehicle tours (4 in total) have different colors and are labeled with v0 and v1 for depot 1; v2 for depot 2 and v3 for depot 3. Nodes 4 to 16 are customers (the number alongside the node index showing the demand, e.g. v3(4:16) - node 4 has demand 16).

Table 4.1 presents a step-by-step solution search. Table 4.2 highlights the final utility and cost per depot. For customer 14, depots 1, 2 and 3 offer 584, 838 and 1002 respectively.

Table 4.2: Per depot utility and cost in active negotiation for modified-*E016-03m*

Participant (Overall Utility)	Participant (Routing Cost)	Optimal Routing Cost (Total Routing Cost)[Gap]
P1(5218) P2(280) P3(625)	P1(1793) P2(234) P3(510)	2402 (2537)[0.056]

As such customer 14 is assigned to depot 1 with utility 838. Next, customer 15 receives offers 73, 538 and 754 respectively from three depots and is assigned to depot 1. Likewise, customers 5, 16, 8 and 11 are also assigned to depot 1. For customer 4, the offers are 642, 384 and 306 respectively and it is assigned to depot 3 with utility 384. Thereafter, customers 7, 13, 10 and 6 are assigned to depot 1. For customers 9 and 12, depot 1 has no offer (n/a) as its capacity is exhausted. In contrast, depots 2 and 3 offer 234 and 280 respectively for customer 9. So, the customer 9 is assigned to depot 2 with utility 280. For customer 12, the offers are 241 and 204 from depot 2 and 3 respectively ans it is assigned to depot 3. Finally, the solution reaches game equilibrium with the routing cost of 2537.

In all possible pairwise depot node swaps possible from this solution, at least one depot decreases its utility and total routing cost increases. Table 4.3 shows the deviation from equilibrium for four nearest cost-wise solutions that can be obtained by node exchanges with respect to the game equilibrium (first column).

Table 4.3: Scenario analysis of deviation from equilibrium

(Game Sol.)\Exchg.	9<>12	10<>12	6<>12	9<>4
Total Cost(2537)	2621	2642	2734	2742
Utility P1(5218)	5218	4720	4802	5173
Utility P2(280)	121	240	182	306
Utility P3(625)	618	452	418	316

In order to evaluate the benefit of employing a predefined customer ordering, we conducted experiments using two sets (each of one hundred random orderings). We investigated three adapted problems n16_k3*; n22_k4* and n30_k3* (* marked problems are adapted multi-depot versions of known VRP instances). The values corresponding to the two sets have similar distributions and expected values indicating relevant sampling. While some random orderings may give very good solutions, such orderings have very small odds of being randomly generated. Moreover, for larger problems the number of possible

random ordering grows with the factorial of the node count, making it increasingly unlikely to find a favorable random ordering. We also found that the proposed outer edge customer ordering gives solutions (integer) that are better or in the vicinity of the corresponding expected value (floating point) of the random ordering experiments (2537 vs. 2538.17 for n16_k3*; 350 vs 353.54 for n22_k4* and 442 vs 529.31 for n30_k3*).

4.5 Benchmarks and Comparative Study

The distributed solution generation framework for collaborative evolutionary learning (passive learning) and cooperative game theory based active negotiation has been implemented in Java and tested on benchmark data of MDVRP instances provided by J.-F. Cordeau [168].

4.5.1 Benchmark Results

The routing cost of a depot is computed using the heuristics technique mentioned in Chapter 3.

Table 4.4: Passive learning based distributed solutions for MDVRP instances

Pb. [nodes/depots] (Max.Veh./Cap.)	depot node:cost (depot time mm:ss)	z^h | z^b [gap=1-z^b/z^h]
P01 [50/4] (4 per depot/80)	51:199 (02:48); 52:203 (03:53); 53: 92 (01:12); 54:114 (00:40);	608 | 577[0.05]
P02 [50/4] (2 per depot/160)	51:114 (01:40); 52:167 (05:08); 53:106 (00:51); 54:102 (02:04);	489 | 473[0.032]
P03 [75/5] (3 per depot/140)	76: 63 (01:02); 77:164 (06:41); 78:175 (00:52); 79:140 (02:20); 80:121 (01:21);	663 | 641[0.033]
P06 [100/3] (6 per depot/100)	101:283 (13:43); 102:223 (04:35); 103:421 (07:32);	927 | 876[0.055]
P07 [100/4] (4 per depot/100)	101:221 (05:38); 102:217 (04:33); 103:209 (08:37); 104:302 (03:41);	949 | 886[0.066]

Table 4.4 presents benchmark results obtained using passive evolutionary learning collaboration approach for the reference problems (with 3 or more depots and up to 100 customers) alongside best known values. We did not include problems with two depots (e.g. P04, P05) since in such case each depot can easily infer the capacity of the other. The first table column shows the problem setup, the second provides the cost and computing

time per depot while the third contrast the sum of the depot costs yielding total cost of solution (z^h) against the best known value z^b, giving the gap. The computation time of the depot represents the routing cost calculation time and excludes the communication time. Thus, the overall solution time is the maximum time value among the depots (e.g. for P07 it is 8:37). We can notice that the obtained solutions have competitive gaps relative to the best known values.

Table 4.5: Active negotiation based distributed solutions using outer edge ordering

Pb. [nodes/depots] (Max. Veh./Cap.)	[depots]{dep:veh.}	$sol^{heur.}_{game}$ | $sol^{best}_{centr.}$[gap]	MCDR/ANDR
P01 [50/4] (4 per depot/80)	[4]{51:4,52:2,53:1,54:4}	637| 577[0.094]	12/32
P02 [50/4] (2 per depot/160)	[4]{51:2,52:1,53:1,54:2}	514| 473[0.079]	12/32
P03 [75/5] (3 per depot/140)	[5]{76:2,77:3,78:2,79:3,80:1}	703| 641[0.088]	13/33
P06 [100/3] (6 per depot/100)	[3]{101:4,102:5,103:6}	1061| 876[0.174]	13/34
P07 [100/4] (4 per depot/100)	[4]{101:4,102:4,103:3,104:4}	1090| 886[0.187]	13/34

We perform distributed solution generation using game-based active negotiation on the same benchmark MDVRP instances. The experiments are conducted with predefined depot and customer locations, known demands and privately kept depot capacities (maximum allowed total vehicles capacity). The insertion cost is computed based on the heuristic technique presented in Chapter 3. Table 4.5 shows the results obtained for each problem along with the gap relative to the centralized best known value. For these problems, we have also calculated the rounded minimum clustering distance (MCDR) and the rounded average node distance (ANDR) in order to illustrate the relevance with respect to a *small-world* transport network. The values show that there are no isolated node sets since MCDR is notably smaller than the ANDR for each of the considered problems.

4.5.2 Comparisons

Table 4.6 compares solution values from Table 4.4, generated using evolutionary learning, against the same problem instances (with 3 depots or more) obtained using distributed game theoretic approach as per Table 4.5. The proposed evolutionary procedure provides

Table 4.6: Passive learning vs. Active negotiation for MDVRP solution generation

Problem [nodes/depots] (Max.Veh./Cap.)	z^h | z^{game}[gap=1-z^{game}/z^h]
P01 [50/4] (4 per depot/80)	608 | 637 [-0.047]
P02 [50/4] (2 per depot/160)	489 | 514 [-0.051]
P03 [75/5] (3 per depot/140)	663 | 703 [-0.060]
P06 [100/3] (6 per depot/100)	927 | 1061 [-0.144]
P07 [100/4] (4 per depot/100)	949 | 1090 [-0.148]

better solutions. The quality of near-optimal solutions obtained from negotiation approach is restricted by the satisfaction of the game equilibrium. Once the game equilibrium is reached it is difficult to improve such a solution in the negotiation approach.

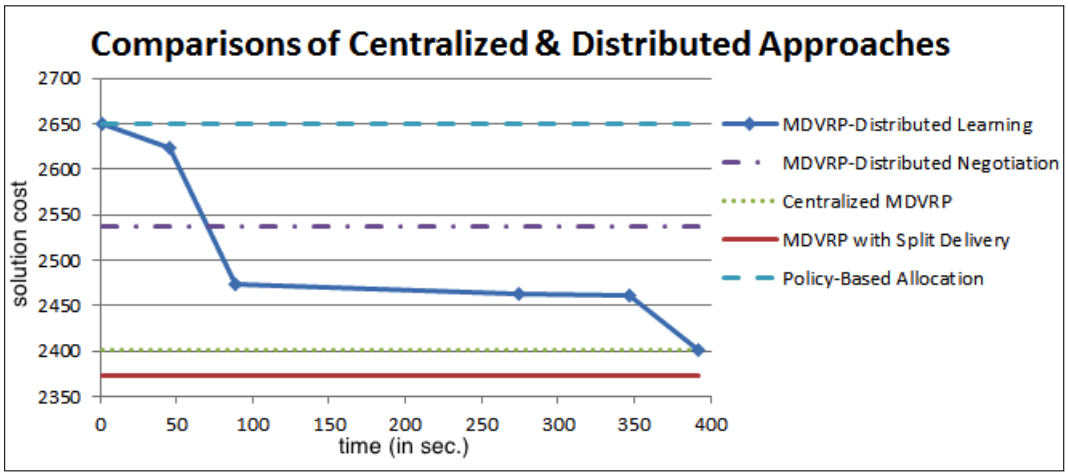


Figure 4.5: Comparative Study of Multi-Depot Vehicle Routing Solution Approaches

Figure 4.5 depicts a comparative study of various aforementioned solution finding approaches as discussed in relation to multi-depot vehicle routing problems. On the x-axis, we project time in seconds while, on the y axis, we evaluate solution cost for the modified-*E016-03m* problem of the case study. The comparison clearly shows that the distributed learning technique presents comparable solution with centralized heuristic/meta-heuristic solution search. However, in collaborative or cooperative setup, the distribution mechanism and the solution search together take longer time than centralized solution generation. As depicted in the figure, split-delivery is proven advantageous for this MDVRP instance since it lowers routing cost compared to the centralized solution approach.

4.5.3 Advantages and Limitations

We summarize next the advantages and limitations of two proposed solution generation approaches for MDVRP in distributed setting. First of all, both approaches offer task decomposition by splitting the problem of multi-depot vehicle routing in distributed setting. Such a decomposition allows distributed decision makers to handle sub-problems locally in a decentralized setup, to use their own input and search parameters (without sharing with others) and to enforce organizational policies. Thus, participants have a larger control in decision making as needed in various organizational setups. Furthermore, as the task decomposition divides the original problem instance into multiple smaller problem instances, each participant can locally produce high quality near-optimal solutions using less memory and computation time.

However, overall solution generation for the MDVRP instances takes longer time with the increase in number of participants since the result synthesis from divided sub-problems and proper task decomposition for the original problem turn more and more complex. Our benchmark results indicate that the solution quality for the MDVRP instances is less competitive for both approaches compared to the near-optimal solution found in centralized setting. This mainly relates to the difficulty of finding the most appropriate partitioning of the customer nodes among participating decision makers. Moreover, these proposed distributed approaches require further improvement to address shared delivery of commodities using vehicles from different depots.

4.6 Summary

In this chapter, we first discussed a model of a chance constraint optimization problem to address MDVRP in distributed settings. We have presented two innovative distributed approaches. First, the multi-round evolutionary learning (passive learning) approach enables collaborative decision makers to search near-optimal solutions for relevant size problem instances by locally computing sub-problems and interacting with other participants without explicitly collecting or sharing fleet/capacity information. Second, the active negotiation approach leverages a game theoretic interaction among cooperative participants where

mechanisms are designed to generate distributed solution search for MDVRP instances. The proposed approach assigns customers to participating depots over transport networks with small-world characteristics using a VCG strategyproof mechanism while aiming to minimize total routing cost. The case studies and benchmark results for both approaches show that near-optimal solutions can be found in these distributed settings despite the lack of all information at each participant's side. In this regard, the learning approach finds better quality near-optimal solution. In future, for passive learning approach, it is possible to investigate more effective initial assignment policies for quicker convergence of the solution search.

Chapter 5

Collaborative Monitor Deployment Problem

In this chapter, we investigate centralized and distributed models and approaches to determine (near) optimal deployment locations of execution monitors over a well-tracked transport network under a fixed budget. The goal of the optimization is minimizing the weighted average energy consumption for data communication between the sensors and the monitors. We illustrate a collaboration strategy of monitor deployment when the total deployment budget is unknown and split among multiple decision makers. We also determine a satisfactory (fair) sharing of monitor deployment cost for every participant.

5.1 Introduction

Typically, SCN involves the flow of products from producers/distributors to customers. Such a network consists of physical locations and traversal paths among these locations. Formally, these locations can be represented as the vertices of a graph while the directed edges between vertices may stand for the traversal paths (arcs) among locations. Monitors can be deployed on the vertices [75] or the edges [104] of a network. In SCN, monitor deployment between two locations (e.g. road, railway, etc.) is costly from security and maintenance perspectives. Thus, this research and development effort focuses on deploying

monitors over a subset of vertices. In this regard, the focus of this chapter can be stated as follows:

- A mathematical model for monitor deployment with multiple decision makers;
- Centralized and distributed approaches to minimize energy consumption;
- Conduct and analyze a case study and generate new benchmark results.

The other contribution of this chapter relates to an automated collaborative negotiation mechanism toward near-optimal monitor deployment with individual budgets. Also, a heuristic is proposed to locally compute solutions under the budget constraint. The optimal selection of monitor locations in SCN, where the total budget is split among participants, is a distributed problem derived from classical facility location [75] and p-median problems [141] which have $\mathcal{NP}$-Hard computation complexity. This requires heuristic or meta-heuristic techniques to efficiently solve large problems [67, 75, 141]. These problems are often addressed with known budget which simplifies the formulation. If the budget is split among the participants, the formulation requires coupling through a joint chance constraint [174] to limit the probability of constraint violation by the participants.

The rest of the chapter is organized as follows. Section 5.2 describes the problem, its assumptions and models. First, a base model is formulated using a known budget constraint. Second, this model is extended for the distributed case where total budget is split among participants. Section 5.3 presents the proposed approach. An exact (optimal) solution algorithm is discussed followed by a faster heuristic technique. Afterward, a distributed approach is detailed whereby participants locally run the heuristic technique and collaborate toward a near-optimal solution. A deployment cost sharing mechanism is also proposed in this regard. Section 5.4 presents a case study. Section 5.5 reports on the results obtained for some Problem Instances (PI). Useful insights are shared on the obtained results and the heuristic performance. Section 5.6 draws concluding remarks and hints on the future work.

5.2 Problem Description and Modeling

5.2.1 Problem Statement

Let $G = (V, E)$ be a complete directed graph representing an SCN. Vertices are divided into monitor nodes (or monitor) and relay nodes (or node). Appropriate equipment can be deployed on a monitor i at a deployment cost c_i, to collect task execution data. The execution information is produced by agents (e.g. vehicles) who visit a subset of vertices in sequence (also called route) through a connecting paths (e.g. roads). A relay node sends collected information to single monitor. Each edge $\langle i, j \rangle \in E$, is associated to a pair of integers: δ_{ij} and w_{ij}. The proposed problem refers to deploying monitors on a subset of vertices such that weighted average energy consumption in sending execution data is minimum. In the process of optimal solution generation, P participants (Decision Makers) will collaboratively determine monitor locations by individually allocating own budget C_p on a subset of vertices. Every solution should respect the following:

- Each vertex is either a monitor or a relay node;
- Deployment cost on each selected vertex is split among a subset of decision makers;
- Every relay node sends data to the monitor incurring least energy consumption.

5.2.2 Assumptions

Task planning (e.g. product delivery) and communication between any two vertices are considered independent of other vertices since G is a complete directed graph. Each vertex is assumed as a source of at least one execution path. Thus, w_{ij} is a positive integer. A number of factors (e.g. communication radius, obstacles, electromagnetic interference, attenuation, environmental situation) affects the energy consumption value δ_{ij} between two locations. Young et al. [253] characterized these effects on radio signal strength over a log normal shadowing model. In contrast, this effort attempts to minimize the weighted average energy consumption based on predetermined values for δ_{ij} specific to every arc that serves as an input to the problem instance. We assume that each relay node always receives execution information from all the agents moving from it to their

next destinations. However, in this context, we ignore the energy consumption in data communication for each agent to its last departing vertex. All deployed monitors are considered to have infinite capacity to receive execution data. No participant knows the exact budget of others. However, it is assumed that a feasible solution always exists and the total budget is sufficient to deploy at least one monitor.

As discussed in Section 2.1.3, in the model formulation, two types of decision variables are employed: control variables and state variables [174]. The monitor deployment problem with a known budget can be represented through a system of linear equations. The values assigned to the control variables (respecting the system of equations) represent a solution. However, monitor deployment with individual budgets is described as a dynamic system with multiple states where each state represents a system of equations with its own control variables. In this context, state variables describe the mathematical "state" of that dynamic system. A subsequent set of state variables typically depends on its previous corresponding set.

5.2.3 Centralized Setup with Single Decision Maker

Let a set of boolean *control variables* x_{ij} determine each possible communication from a relay node to a monitor. Let y_i determine location of monitors in directed graph G.

$$x_{ij} = \begin{cases} 1, & \text{if vertex } i \text{ communicates with vertex } j; \\ 0, & \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if a monitor is placed at vertex } i; \\ 0, & \text{otherwise.} \end{cases}$$

Then, the objective function of a central decision maker can be presented as a binary integer programming model:

$$min \ \frac{\sum\limits_{i \epsilon V} \Big((\sum\limits_{k \epsilon V} w_{ik}) \times (\sum\limits_{j \epsilon V} \delta_{ij} \times x_{ij}) \Big)}{\sum\limits_{i \epsilon V} (\sum\limits_{k \epsilon V,} w_{ik})}; \ i \neq j; i \neq k \tag{5.1}$$

Subject to:

Flow conservation:

$$x_{ij} \leq y_j, \quad \forall i,j \in V; i \neq j \tag{5.2}$$

$$x_{ij} + x_{jk} \leq 1, \quad \forall i,j,k \in V \tag{5.3}$$

$$\sum_{j \in V, i \neq j} x_{ij} = 1 - y_i, \quad \forall i \in V \tag{5.4}$$

$$\sum_{i \in V} \sum_{j \in V, i \neq j} x_{ij} = |V| - \sum_{i \in V} y_i \tag{5.5}$$

Budget and energy consumption restrictions:

$$0 < \sum_{i \in V} c_i \times y_i \leq C, \quad c_i > 0 \tag{5.6}$$

$$\delta_{ij} \times (2 \times x_{ij} + y_j - y_k - 1) \leq \delta_{ik} \times (1 - 2 \times x_{ik} + y_j - y_k) \quad \forall i,j,k \in V; i \neq j \neq k \tag{5.7}$$

Eq. (5.1) presents the objective function (for centralized decision making) that minimizes weighted average energy consumption for total data traffic subjected to the following constraints. Eq. (5.2) mandates that if a directed edge $\langle i,j \rangle$ is selected for a solution then its destination vertex j must be a monitor. Eqs. (5.3)-(5.5) impose additional inequalities. Eq. (5.3) restricts that the monitors only receive data while relay nodes only send data to the monitors. Eq. (5.4) states that every vertex communicates with a monitor unless it is itself a monitor. Eq. (5.5) assures that the sum of all arcs to all monitors is equal to the difference between all nodes and all monitors. Eq. (5.6) denotes the budget constraint C to deploy at least one monitor. Eq. (5.7) denotes that a relay node communicates to the monitor which incurs least energy consumption (*see* proof below). Eq.s (5.2), (5.4) and (5.6) form the minimum set of constraints to reach the optimal solution but additional valid constraints are added to reduce the solution search space.

Proposition 1. *If relay node i sends execution information to monitor j on a directed graph $G = (V,E)$, then the constraint $\delta_{ij} \times (2 \times x_{ij} + y_j - y_k - 1) \leq \delta_{ik} \times (1 - 2 \times x_{ik} + y_j - y_k)$ $\forall i,j,k \in V$ reflects that for any other monitor k, $i \neq j \neq k$, the energy consumption on arc $\langle i,k \rangle$ is higher than $\langle i,j \rangle$.*

Proof. Let binary variables y_j and y_k decide if j and/or k are the monitors. Let x_{ij} and x_{ik} be two other binary variables to decide if node i communicates to j or k. With these

four variables, there exist sixteen combinations. However, given Eq. (5.2), if $x_{ij} = 1$ then $y_j = 1$. Thus, four combinations can be excluded. In this setting, energy consumption δ_{ij} and δ_{ik} are related as: (i) If $x_{ij} = y_j = 1$ then $\delta_{ij} \leq \delta_{ik}$, (ii) If $x_{ik} = y_k = 1$ then $\delta_{ik} \leq \delta_{ij}$, or (iii) δ_{ik} and δ_{ij} are not related otherwise. These relations can be succinctly captured in: $\delta_{ij} \times (2 \times x_{ij} + y_j - y_k - 1) \leq \delta_{ik} \times (1 - 2 \times x_{ik} + y_j - y_k)$. □

5.2.4 Centralized Setup with Multiple Decision Makers

Decision makers may participate based on competition, cooperation or collaboration. In a competition or cooperation, rational participants negotiate for deploying monitors individually on a subset of selected vertices. Classical form of such negotiation is often modeled using the game theory and mechanism design [195]. In such setup, decision makers negotiate based on a quantifier, namely *price*. *Price* can be computed on a vertex using monitor deployment cost, decision maker's utility for likely collection of information in that vertex and available individual budget. However, this may lead to budget wastage since the deployment cost cannot be shared easily. In contrast, in a *collaborative* setup, decision makers may share the deployment cost of a monitor if they prefer the same vertex. Let integer *state variables* u_{ip} represent the budget contribution of decision maker p for vertex i. Then the constraints can be modified as:

$$0 \leq u_{ip} \leq c_i; \ \ \forall i \in V, p \in P \tag{5.8}$$

$$\sum_{i \in V} u_{ip} - C_p \leq 0, \ \ \forall p \in P \tag{5.9}$$

$$c_i \times y_i - \sum_{p \in P} u_{ip} \leq 0, \ \ \forall i \in V \tag{5.10}$$

$$1 + \sum_{p \in P} u_{ip} - c_i - ((|P| - 1) \times c_i + 1) \times y_i \leq 0, \ \ \forall i \in V \tag{5.11}$$

Eq. (5.8) prevents individual contribution u_{ip} from exceeding the deployment cost of i. Eq. (5.9) replaces Eq. (5.6) in the base model since the total budget is now distributed. Eq.s (5.10) and (5.11) determine whether a monitor can be deployed at vertex i (i.e. $y_i = 1$) from the contributions of participants. These constraints determine individual contributions to optimally deploy monitors. However, in this setup, all individual budgets (C_p) are known to a central authority that performs the optimization. Therefore, this setup requires further

extension to capture the optimal decision making without a central authority.

5.2.5 Decentralized Setup with Multiple Decision Makers

Let there be a set $S^* \subset V$, to optimally deploy monitors for budget $C = \sum_{p \in P} C_p$. In collaborative deployment, the individual budget C_p is best spent if participants jointly prefer deploying monitors in S^*. Thus, every participant may aim to derive S^* and contribute to every monitor in S^*. This puts the deployment procedure into risk as a monitor cannot be deployed if $\sum_{p \in P} u_{ip} < c_i$ (Eq. (5.10)). Ono and Williams [174] formalized such decentralized behavior of decision makers through joint chance-constraints for multi-agent systems. An iterative optimization of control variables in every state may lead to overall risk minimization by updating changes in state variables. Let T and R be the total iterations and global risk factor respectively. Then, a joint chance constraint can be written by substituting u_{ip}, x_{ij} and y_i with u^t_{ip}, x^t_{ijp} and y^t_{ip} $(t = 1, 2, \ldots, T)$ and combining Eqs. (5.10) and (5.11):

$$\mathbb{P}r\Bigg[\bigwedge_{p \in P}\Big[\sum_{t=0}^{T-1}[y^t_{ip} \times c_i - \sum_{p \in P} u^t_{ip} \leq 0]\Big] \bigwedge \Big[\sum_{t=0}^{T-1}[\sum_{p \in P} u^t_{ip} + 1 - c_i - \{(|P| - 1) \times c_i + 1\} \times y^t_{ip} \leq 0]\Big]\Bigg] \geq 1 - R \tag{5.12}$$

Eq. (5.12) represents a joint chance constraint that indicates a failure if any participant does not satisfy risk constraints (Eqs. (5.10) and (5.11)). Ono and Williams [174] proved that if the objective function is convex and $0 \leq R \leq 0.5$, such an optimization problem is also convex. This helps to decompose R with distributed risk π_{ip} for each participant on each vertex such that in every iteration $\sum_{p \in P} \sum_{i \in V} \pi_{ip} \leq R$ where $\pi_{ip} \geq 0$. Using Boole's inequality, $\mathbb{P}r(R) = \mathbb{P}r(\bigcup_{i \in V, p \in P} \pi_{ip}) \leq \sum_{p \in P} \sum_{i \in V} \pi_{ip}$ [174]. Thus, the bounded risk can be replaced by an unbounded penalty in distributed optimization function (s_p):

$$min \sum_{t=0}^{T-1} \frac{\sum_{i \in V}\Big((\sum_{k \in V} w_{ik}) \times (\sum_{j \in V} \delta_{ij} \times x^t_{ijp})\Big)}{\sum_{i \in V}(\sum_{k \in V,} w_{ik})} + \rho \sum_{i \in V} \pi_{ip} \tag{5.13}$$

Subject to:

$$x^t_{ijp} \le y^t_{jp}, \quad \forall i,j,k \in V \tag{5.14}$$

$$x^t_{ijp} + x^t_{jkp} \le 1, \quad \forall i,j,k \in V \tag{5.15}$$

$$\sum_{j \in V, i \neq j} x^t_{ijp} = 1 - y^t_{ip}, \quad \forall i \in V \tag{5.16}$$

$$\sum_{i \in V} \sum_{j \in V, i \neq j} x^t_{ijp} = |V| - \sum_{i \in V} y^t_{ip} \tag{5.17}$$

$$0 < \sum_{i \in V} c_i \times y^t_{ip} \le \sum_{p \in P} \sum_{i \in V} u^t_{ip}, \quad c_i > 0 \tag{5.18}$$

$$\delta_{ij}(2 \times x^t_{ijp} + y^t_{jp} - y^t_{kp} - 1) \le \delta_{ik}(1 - 2 \times x^t_{ikp} + y^t_{jp} - y^t_{kp}) \forall i,j,k \in V i \neq j; i \neq k \tag{5.19}$$

$$0 \le u^t_{ip} \le c_i; \ \forall \ i \in V, \ \forall p \in P \tag{5.20}$$

$$\sum_{i \in V} u^t_{ip} - C_p \le 0, \quad \forall p \in P \tag{5.21}$$

$$u^{t+1}_{ip} = A_p . u^t_{ip} + B_p . y^t_{ip}; \ \forall \ i \in V, p \in P \tag{5.22}$$

$$\sum_{t=0}^{T-1} (c_i \times y^t_{ip} - \sum_{p \in P} u^t_{ip}) \le -f_{ip}(\pi_{ip}), \ \forall i \in V \tag{5.23}$$

$$\sum_{t=0}^{T-1} (1 + \sum_{p \in P} u^t_{ip} - c_i - ((|P| - 1) \times c_i + 1) \times y_i) \le -f'_{ip}(\pi_{ip}), \ \forall i \in V \tag{5.24}$$

The proposed procedure includes individual optimization of $\sum_{i \in V} \pi_{ip}$ in Eq. (5.13) which reflects the influence of risk in weighted average energy consumption; ρ, also termed as "price of risk", is a penalty constant known to all participants [174]. Eqs. (5.23) and (5.24) introduce two sets of monotonically decreasing functions f_{ip} and f'_{ip} to progressively reduce the gap between the centralized and the decentralized decision making. At every iteration, decision of monitor deployment is reflected in y^t_{ip}. Eq. (5.18) prevents the sum of the values of y^t_{ip} from exceeding total available budget as determined (through a communication process) locally. The updated contribution u^{t+1}_{ip} for each vertex will be computed after each round using Eq. (5.22). A_p, B_p are participant specific constants. The problem is convex if Eq. (5.13) is convex, Eq. (5.22) is linear and f_{ip} and f'_{ip} are single-valued and monotonically decreasing functions [174]. One can anticipate that the distributed solving will require communication among the participants to synchronize u^t_{ip} values.

5.3 Proposed Approach

Collaborative monitor deployment under a known budget involves a combination of classical *set covering* and one-dimensional *bin-packing (knapsack)* problems. In addition, a multi-round coordination of individual decisions is needed to iteratively build consensus. The set covering mandates choosing optimal set of monitors $S^* \in V$ such that overall weighted average energy consumption for relay nodes $(V \setminus S^*)$ is minimum. Bin packing restricts the optimal deployment of monitors in a budget. As the problem is $\mathcal{NP}$-Hard on a complete graph, heuristics can near-optimally solve it faster.

5.3.1 Exact Algorithm

Let integer $\gamma_{ij} = \delta_{ij} \times \sum_{k \in V \setminus \{i\}}(w_{ik})$ be a measure of weighted energy consumption for vertex i to communicate with vertex j. Then, the aforementioned Eq.s can be further simplified into rules (R1-R5) to design the search procedure:

R1: *In SCN, if the size of the optimal set of deployed monitors is $|S|$, vertex i does not communicate to vertex j if there exist at least $(|V| - |S|)$ distinct edges where the energy consumption is lower than that of $\langle i, j \rangle$.* (*see* proof below)

R2: *If the optimal set of deployed monitors is S, then relay node i communicates to monitor j iff the value of γ_{ij} is the lowest, i.e. $\forall j' \in S,\ j \neq j',\ \gamma_{ij'} > \gamma_{ij}$.* [Eq. (5.7)]

R3: *Weighted average energy consumption is higher for any proper subset of a given set of monitors* [Eq. (5.4)].

R4: *The deployment cost is higher for any superset of a set of monitors* [Eq. (5.6)].

R5: *Monitors do not communicate among them* [Eq. (5.3)].

Rule 1. *(R1) In supply chain network, if the size of the optimal set of deployed monitors is $|S|$, vertex i does not communicate to vertex j if there exist at least $(|V| - |S|)$ distinct edges where the energy consumption is lower than that of $\langle i, j \rangle$.*

Proof. Let S and $V \setminus S$ be monitors and nodes respectively. In order to find optimal weighted average energy consumption, γ_{ij} for each node $i \in V \setminus S$ and $j \in S$ can be reduced

to: $\min(\frac{\sum_{i \in V \setminus S} \sum_{j \in S} \gamma_{ij} \times x_{ij}}{Z})$ (*see* Eq.s (5.1), (5.3) and (5.4)). Z represents the total amount of communicated information. The exact members of S are unknown at the beginning. However, for any $S \subset V$ (as restricted by budget constraints), if $x_{ij} = 1$, there exist at least $(|V| - |S|)$ distinct edges where $\gamma_{ij'} < \gamma_{ij}, j' \in V$. This leads to assigning $x_{ij'} = 1$ for one of these $(|V| - |S|)$ distinct edges in the solution (by Eq. (5.2)) and contradicts Eq. (5.4) since x_{ij} is already 1. □

Rule R1 eliminates redundant communication links with high γ_{ij} values. Rule R2 forces a relay node to choose its monitor where γ_{ij} is minimum. Rules R3 and R4 set boundaries for solution search. Rule R5 preserves one role (relay node or monitor) for the vertices in a solution. Algorithm 5 applies these rules to find optimal solution for monitor

Algorithm 5 : Monitor Allocation ($MonAlloc$)

1: Initially: $\mathcal{C}$, $Q \leftarrow \langle \phi, V, \phi \rangle$, and $E_{sort} \leftarrow \langle E, <_{\gamma_{ij}} \rangle$
2: Input: C_d, Max_Energy; Output: S^*
3: $S^* \leftarrow \phi$; $\Delta \leftarrow Max\_Energy$
4: **if** $getCost(V) > C_d$ **then**
5: **while** $E_{sort} \neq \phi$ **do**
6: $e \leftarrow$ first element of E_{sort};
7: *update* Q with e;
8: *compute* $\mathcal{C}' \subseteq \mathcal{P}(V)$ such that monitor can be deployed in $S \in \mathcal{C}'$ given the edges in Q
9: **for** $S \in \mathcal{C}' \setminus \mathcal{C}$ **do**
10: **if** $getCost(S) \leq C_d$ **and** $\sum_{i \in V \setminus S} min_{j \in S}(\gamma_{ij}) < \Delta$ **then**
11: $S^* \leftarrow S$; $\Delta \leftarrow \sum_{i \in V \setminus S^*} min_{j \in S^*}(\gamma_{ij})$;
12: **end if**
13: **end for**
14: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$; $E_{sort} \leftarrow E_{sort} \setminus \{e\}$
15: **end while**
16: **end if**
17: **return** S^*;

deployment from $\mathcal{P}(V)$ (power-set of V) possibilities. E_{sort} stores a sequence of edges sorted in ascending order of γ_{ij}. Q is a bipartite graph with two disjoint sets of vertices. A set of directed edges (E') connects members of these two sets. Then, $E' \subseteq V_1 \times V_2$ reflects relay node to monitor communications. Initially, Q:=$\langle \phi, V, \phi \rangle$, whereby all vertices are assumed as monitors. The cost of such deployment exceeds the budget (Step 4). During

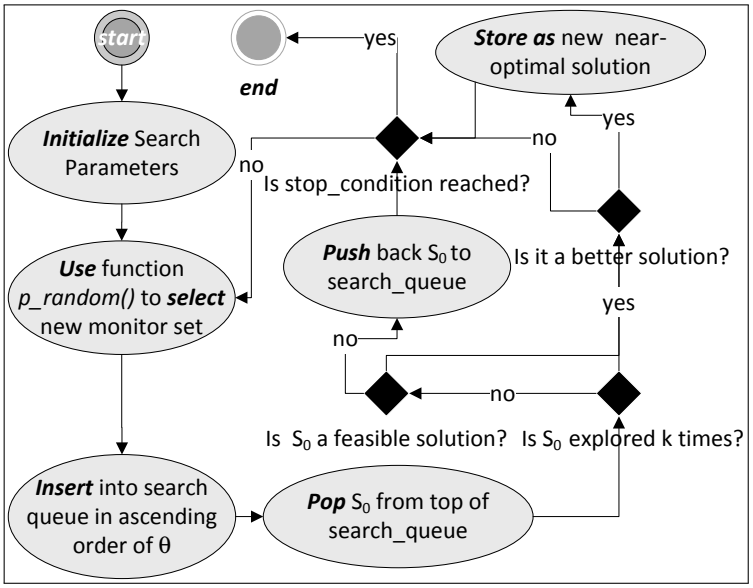


Figure 5.1: Heuristic technique of monitor deployment

the search, deployment cost gradually reduces to feasible values as edge e, having minimum γ_{ij}, is iteratively removed from E_{sort} and added to Q (Step 7). The source vertex of e is a relay node ($V_1 = V_1 \cup \{e.source\}$) and thus excluded from V_2 ($V_2 = V_2 \setminus \{e.source\}$). Rules R1-R5 are applied to produce a collection of possible sets of monitors ($\mathcal{C}'$) from Q in order to compare against $\mathcal{C}$, (a set of already generated and verified sets of monitors). Each member of $\mathcal{C}' \setminus \mathcal{C}$ is then tested for feasibility (Steps 10-12). If the new weighted average energy consumption value is lower than previously stored value in Δ, it replaces existing value in Δ. Corresponding S^* (set of monitors), $\mathcal{C}$ and E_{sort} (Step 14) are also updated for the next round. Finally, S^* is returned. Rule R3 represents a stop condition for this algorithm.

5.3.2 Heuristic Technique

A heuristic algorithm can be applied to generate $\mathcal{C}'$ in a tractable manner for larger problems. Figure 5.1 depicts the overview of the technique. It uses an efficiently generated

solution pool by comparing a quantifier θ_S, as presented in Eq. (5.25):

$$\theta_S = \Big[\sum_{i \in V \setminus S} \min_{j \in S}(\gamma_{ij})\Big] / \Big[\sum_{j \in V} c_j - \sum_{j \in S} c_j\Big] \tag{5.25}$$

where, $\sum_{j \in V} c_j$ is the total deployment cost for all vertices.

First, $\mathcal{C}_{sort}$ is defined as a sorted search queue (of maximum size sz) represented by $\langle \mathcal{S}, <_{\theta_S} \rangle$ where $S \in \mathcal{S}$ and $\mathcal{S} \subset \mathcal{P}(V) \setminus \{\phi\}$. Pseudo random function $p\_random(\dots)$ produces a subset of V as a candidate set of monitors in each iteration. Initially, $\mathcal{C}_{sort}$ is empty. Thus, $p\_random(\dots)$ selects a subset of monitors from V. This subset is inserted into $\mathcal{C}_{sort}$ where it is arranged in ascending order of θ_S. New subsets of monitors are iteratively explored later by picking the top entry (S_0) from $\mathcal{C}_{sort}$. The cardinality of explored subset is at least $|S_0| - cnv$ as determined by a convergence parameter (cnv). An entry, representing a set of candidate monitors, from $\mathcal{C}_{sort}$ is discarded once it is explored k times to generate its subsets. Function $p\_random(\dots)$ is computed based on weight ($wt(i)$) of vertex i which is defined as follows:

$$wt(i) = \frac{1}{\sum_{j \in S^{-i}} \frac{1}{\delta_{ij}}} \Big[\sum_{j \in S^{-i}} \frac{\gamma_{ij}}{\delta_{ij}}\Big] - \frac{1}{\sum_{j \in S^{-i}} \frac{1}{\delta_{ji}}} \Big[\sum_{j \in S^{-i}} \frac{\gamma_{ji}}{\delta_{ji}}\Big] \tag{5.26}$$

where, $\forall i \in S,\ S^{-i} = S \setminus \{i\}$. In Eq. (5.26), $wt(i)$ is a measure of saving that monitor i offers compared to other monitors. It determines the likelihood of a vertex to send or receive data in the optimal solution. A positive value indicates that, in general, it is more energy consuming for vertex i to send data to other vertices than receiving while a negative value indicates the opposite. $wt(i)$ is then projected as a fraction over a 0-to-1 scale and normalized. In this scale, 0 and 1 mean relay node and monitor respectively. The value 0.5 means random probability of a vertex with no bias to any role. This value is an input to the $p\_random(\dots)$ along with a seeding parameter. The latter helps to perform the selection in a manner that can be retraced for the same seed value. When $|\mathcal{S}| > sz$, entries with largest θ value in $\mathcal{C}_{sort}$ are dropped to maintain sz. If the deployment cost of selected S ($getCost(S)$) does not exceed the budget, a solution is found. The new solution is retained if it yields a lower weighted average energy consumption than the previously stored solution. S is then removed from $\mathcal{C}_{sort}$. The retained solution is returned once

one of the following stop conditions is true: (i) after ite iterations; (ii) if $\mathcal{C}_{sort}$ is empty; (iii) if there is no improvement in θ after p ($p \ll ite$) iterations. The procedure is bound in memory as a result of dropping the set of monitors with the largest value of θ when $|\mathcal{S}| > sz$. Initially deployment cost is highest (infeasible) and the weighted average energy consumption is 0. The heuristic iteratively evaluates the increase in weighted average energy consumption to decrease the deployment cost to feasibility.

5.3.3 Distributed Monitor Deployment

Algorithm 6 : Searching appropriate set of monitors

1: Initially: $\mathcal{C}_{sort} \leftarrow \phi$
2: Input: P; Output: s_p
3: $s_p \leftarrow MonAlloc(\mathcal{C}_{sort}, |P|.C_p, Max\_Energy)$
4: *communicate* s_p to other participants
5: *receive* $s_{p'}$ from all $P \setminus \{p\}$ participants
6: $rank \leftarrow getRank(s_{-p}, s_p, p)$
7: $C_{min} \leftarrow \sum_{p \in P} \lfloor \frac{getEstCost(s_p)}{|P|} \rfloor$
8: $C_{max} \leftarrow \max_{p \in P} getEstCost(s_p)$
9: **while** $C_{min} < C_{max}$ **do**
10: $s_{mid} \leftarrow MonAlloc(\mathcal{C}_{sort}, \lceil \frac{C_{min}+C_{max}}{2} \rceil, Max\_Energy)$
11: $\mathcal{U} \leftarrow AdjustContribution(s_{mid}, P, p, \tau, rank)$
12: **if** $\sum_{u_{ip} \in \mathcal{U}} u_{ip} \geq getEstCost(s_{mid})$ **then**
13: $s_p \leftarrow s_{mid}$; $temp \leftarrow getEstCost(s_{mid})$
14: **if** $C_{min} = temp$ **then** *break*; **end if**
15: $C_{min} \leftarrow getEstCost(s_{mid})$
16: **else**
17: $C_{max} \leftarrow getEstCost(s_{mid})$
18: **end if**
19: **end while**

The distributed monitor deployment requires solving two more sub-problems: (i) decision coordination to derive a common solution and (ii) computation of individual contribution on each monitor deployment. Two distributed algorithms are presented in this regard. Algorithm 6 exploits binary search and decision coordination while locally running the heuristic (Step 3). $\mathcal{C}_{sort}$ may be stored in a common memory shared by the distributed system to take computational advantage of tested solutions by other participants.

Initially, each participant determines a candidate set of monitors assuming the available budget $|P|$ times which is larger than its own. Once the choices are communicated (Step 4-5), participants are ranked (Step 6) in a strict descending order of communicated solution cost. Then, steps 7-8 determine the minimum (C_{min}) and maximum (C_{max}) bounds of available budget to run a binary search (steps 9-21) to reach a common selection. Step 10 returns a set of monitors (if exists) for half of the sum of C_{min} and C_{max}. If this solution can be deployed collaboratively, the lower bound C_{min} is replaced with the deployment cost. Otherwise, the upper bound is updated with the same. Binary search ensures a deterministic and fast convergence to a near-optimal solution through iteratively tighter budget approximation as reflected from the choices of the participants. Thus, it helps reducing the "price of risk" in collaborative deployment.

Algorithm 6 calls Algorithm 7 in step 11 for contribution adjustment among the participants. Algorithm 7 takes input s_{mid}, a temporary solution, and reorders the candidate monitors in s_{sort} based on the amount of their collected information. For contribution adjustment, each participant tries to make an average contribution (integer value) for every monitor selected in s_{sort} (steps 4-6). If the average contribution cannot be made by a subset of participants, the remaining participants cover the deficit equally (step 19). No participant can contribute to monitor(s) collecting larger amount of information unless it contributes accordingly to all the monitors collecting less amount of information (step 17). This policy helps in reaching consensus while extracting necessary contribution from each decision maker. Algorithm 7 divides individual budget (C_p) to deploy monitors as per the aforementioned policy. Step 7 communicates the individual decision ($\mathcal{U}$) to other participants. Then, an iterative contribution adjustment procedure takes place at steps 8-26 up to τ rounds. Upon receiving the other decisions, $\mathcal{U}$ is updated and a newer price adjustment is communicated. Step 24 can break the loop once a consensus is reached to deploy all monitors in s_{mid}.

Algorithm 7 : Distributed contribution adjustment

Initially: $t \leftarrow 0; \mathcal{U} \leftarrow \{u_{ip} = 0 | \forall i \in V, p \in P\}$;
Input: $s_{mid}, P, p, \tau, rank$; Output: $\mathcal{U}$
$s_{sort} \leftarrow \langle s_{mid}, < \sum_{i \in V} \frac{\gamma_{ij} \cdot x_{ij}}{\delta_{ij}} \rangle$;$\mathbf{y} \leftarrow \{y_i = 0 | \forall i \in V\}$;$est \leftarrow C_p$
for $i := 1$ *to* $|s_{sort}|$ **do**
  $y_i \leftarrow 1$; $u_{ip} \leftarrow \min(est, \lceil \frac{getCost(\{i\})}{|P|} \rceil)$; $est \leftarrow est - u_{ip}$;
end for
communicate $\mathcal{U}$ to other participants;
while $t \leq \tau$ **do**
  $est \leftarrow C_p$; $\mathbf{j} \leftarrow \{j_i = 0 | \forall i \in s_{sort}\}$
  for $i := 1$ *to* $|s_{sort}|$ **do**
    for $p \in P$ **do**
      receive $u_{ip'}$ for p' where $p' \neq p$ and insert in $\mathcal{U}$
      if $u_{ip} < \frac{y_i . c_i}{|P|}$ **then** $j_i \leftarrow j_i + 1$; **end if**
    end for
    $temp \leftarrow (\sum_{p \in P} u_{ip} - y_i . c_i)$
    if $temp \geq 0$ **then**
      $u_{ip} \leftarrow u_{ip} - \max(0, \lceil \frac{temp - rank + 1}{|P|} \rceil)$
    else
      $u_{ip} \leftarrow \min(est, u_{ip} + \lceil \frac{-temp}{|P| - j_i} \rceil)$
    end if
    $est \leftarrow est - u_{ip}$;
  end for
  communicate $\mathcal{U}$ to other participants;
  if $\sum_{j_i \in \mathbf{j}} j_i = 0$ **then** *break*; **end if**
  $t \leftarrow t + 1$;
end while

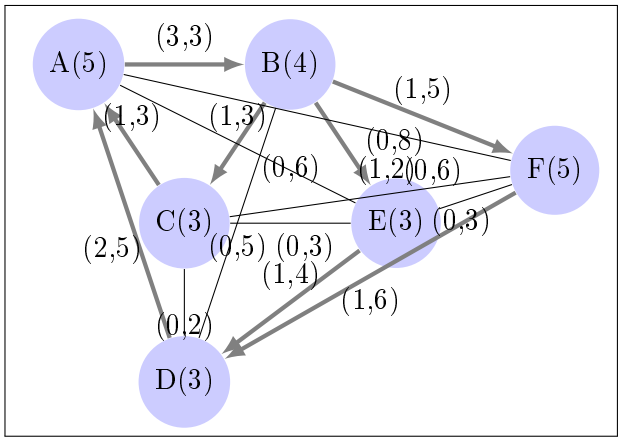


Figure 5.2: An instance of execution monitor deployment problem

5.4 Case Study

Figure 5.2 shows a plan execution scenario on a complete transport network with six vertices. The deployment cost is depicted with the vertex name (e.g. deployment cost of A is 5). The execution paths are marked by directed edges annotated with two integers for the number of agents on the execution path and associated energy consumption per transmission. For example, the execution path from F to D is used by 1 agent and the energy consumption on FD is 6. Assuming $\delta_{ij} = \delta_{ji}$, the weighted average energy consumption is minimized for (i) known total budget of 7 and (ii) two distributed decision makers (P1 and P2) with individual budgets of 3 and 4.

Table 5.1: Case Study: weighted cost of edges in ascending order

Edge	γ_{ij}	Edge	γ_{ij}	Edge	γ_{ij}	Edge	γ_{ij}	Edge	γ_{ij}	Edge	γ_{ij}
EB	2	EC	3	FB	5	FD	6	AB	9	AD	15
CD	2	FE	3	EA	6	FA	8	BC	9	BF	15
CA	3	EF	3	CF	6	DE	8	DA	10	AE	18
CB	3	DC	4	FC	6	BA	9	DB	12	BD	18
CE	3	ED	4	BE	6	AC	9	DF	12	AF	24

Table 5.1 presents E_{sort} where edges are ordered by γ_{ij}. Figure 5.3 depicts the result of each iteration (steps 5-15) in Algorithm 5. Every iteration is presented in a box with its number at the left. The candidate monitors are shown using $\{\dots\}$. The sign ":X" means that the required deployment cost exceeds the deployment budget 7. Otherwise, the deployment cost is mentioned with the corresponding set. In this example, no edges can be eliminated by Rule $R1$ since the smallest number of monitors that can be deployed using the budget 7 is 1.

Algorithm 5 starts with a bipartite graph of $V_1 = \phi$ and $V_2 = V$. Initially no edges are considered connected. Deploying monitors in every vertex of V (follow iteration 0 in Figure 5.3) is infeasible for budget value 7. Therefore, EB is added from Table 5.1. Thus, E communicates data to B (follow step 6 in Algorithm 5). Accordingly, $\mathcal{C}' = \{\{A,B,C,D,F\}\}$. With the addition of arc EB, {A,B,C,D,F} is the new set of monitors to be tested for feasibility. Similarly, at iteration 2, $\mathcal{C}' = \{\{A,B,D,F\}\}$ and CD is added to Q. The other possible candidate set {A,B,D,E,F} is not tested since $\{\{A,B,D,F\}\}$ is infeasible and {A,B,D,E,F}

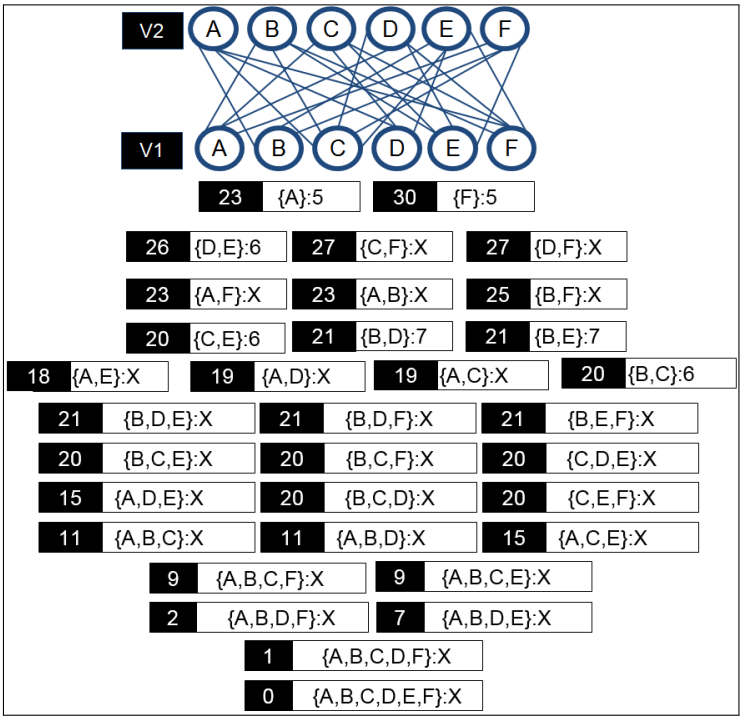


Figure 5.3: Centralized monitor allocation trace

is its superset (Rule $R4$). Thus at iteration 2, $Q := \langle\{\text{C,E}\},\{\text{A,B,D,F}\},\{\text{EB,CD}\}\rangle$. No new candidate set is found up to iteration 6 while adding edges from Table 5.1. At iteration 7, there are two new candidates: {A,B,D,E} and {A,B,C,D,E}. Since {A,B,D,E} is infeasible, {A,B,C,D,E} is not tested. Subsequently, the procedure may run at most $|V|\times(|V|-1)=30$ iterations unless a stop condition is met. The optimal solution {B,D} is found at iteration 21 with the weighted average energy consumption 1.636.

In distributed setup, decision makers P1 and P2 individually run Algorithm 6 with budget 3 and 4 respectively. Steps 3 and 10 of Algorithm 6 invoke heuristics to choose the candidate set of monitors. Decision makers also communicate their choices (steps 4-5) with increasingly tighter bounds as guided by a binary search. P1 and P2 execute step 3

Table 5.2: Case Study: Distributed contribution adjustment

Party	Iteration t0		Iteration t1		Iteration t2
P1	E[2], C[1]	communicate	D[2], B[1]	communicate	D[2,1], B[1,3]
P2	D[2], B[2]		D[1], B[3]		D[2,1], B[1,3]

(in Algorithm 6) with budget 6 and 8 respectively. The proposed outcome for P1 and P2 are: {C,E} and {B,D}. Their ranks are 2 and 1 respectively. C_{min} and C_{max} are 6 and 7. In this case, the binary search will stop after one iteration. Step 11 will be executed to determine the price contribution with the budget 7 and candidate set of monitors: {B,D}. Table 5.2 reflects the individual contribution of P1 and P2 from the execution trace of Algorithm 7.

5.5 Results and Analysis

The accuracy and performance of the proposed algorithms are analyzed on the routes of two commodity delivery problem types: Capacitated Vehicle Routing Problem (CVRP) and Multi-Depot Split-Delivery Vehicle Routing Problem (MDSDVRP) instances [100, 101]. These choices are based on solution (vehicle routes) availability [193]. Experiments are performed on few representative instances from P-Series CVRP [21] and SQ-Series MDSDVRP. The demand of each vertex is mapped to its monitor deployment cost. In CVRP, first vertex is considered as a depot with 0 demand. Vehicles with finite capacity always serve customer demands from this depot. In MDSDVRP, multiple depots serve the customer demands. Thus, depots are always selected as monitors with no deployment cost. Vehicles depart and return to the same depot. The routing solutions for each chosen problem instance are given below. **Pn16K8** (near) optimal routes:

R.1:$1,3,1$ **R.2:**$1,7,1$ **R.3:**$1,8,1$ **R.4:**$1,16,13,11,1$
R.5:$1,15,6,1$ **R.6:**$1,14,10,8,1$ **R.7:**$1,12,5,1$ **R.8:**$1,4,2,1$

Pn19K2 (near) optimal routes:

R.1:$1,5,12,15,13,4,18,17,9,7,1$; **R.2:**$1,19,6,14,16,10,8,3,11,2,1$;

Pn22K2 (near) optimal routes:

R.1:$1,7,3,14,10,8,22,18,15,6,21,1$
R.2:$1,17,2,11,9,19,20,4,13,16,12,5,1$

Pn23K8 (near) optimal routes:

R.1:$1,14,10,18,1$ **R.2:**$1,5,8,1$ **R.3:**$1,4,20,19,1$ **R.4:**$1,22,21,7,1$
R.5:$1,9,17,1$ **R.6:**$1,3,2,1$ **R.7:**$1,11,13,16,12,1$ **R.8:**$1,6,15,23,1$

Table 5.3: Benchmarks on CVRP-P-Series [21] and SQ-Series [100] problems

Bgt.	Monitors	Opt. (sec.)	Heur. (sec.)
PI	Pn16K8; *see. Appendix for (near) optimal routes*		
20	[1,14,16]	6.522 (0.005)	6.522 (0.007)
50	[1,6,11,12,13,14]	3.61 (0.058)	3.61 (0.06)
100	[1,4,6,8,10,11,12,13,14,16]	2.04 (0.16)	2.04 (0.19)
PI	Pn19K2; *see. Appendix for (near) optimal routes*		
20	[1,6,11]	9.9 (0.005)	9.9 (0.007)
50	[1,6,8,11,12,18]	5.55 (0.07)	5.55 (0.08)
100	[1,4,8,11,12,13,14,18,19]	3.25 (0.36)	3.25 (0.358)
PI	Pn22K2; *see. Appendix for (near) optimal routes*		
20	[1,11,14,20]	9.261 (0.025)	9.261 (0.032)
50	[1, 2, 12,13,14,20,22]	5.043 (0.165)	5.05 (0.180)
100	[1,2,4,6,10, 12,13,14,20,21,22]	3.174 (2.89)	3.348 (0.506)
PI	Pn23K8; *see. Appendix for (near) optimal routes*		
20	[1,11,14,20]	7.16 (0.035)	7.2 (0.045)
50	[1,2,12,13,14,20,23]	3.93 (0.29)	3.95 (0.325)
100	[1,2,4,10,11,12,14,16,20,21,22,23]	2.43 (6.35)	2.667 (0.351)
PI	Pn40K8; *see. Appendix for (near) optimal routes*		
20	[1,11,18,30,37]	11.5 (0.135)	11.5 (0.17)
50	[1,10,11,18,23,25,27,37]	7.682 (59.378)	9.341 (8.973)
100	[1,2,5,11,18,20,22,25,27,30,32,37,39]*		6.11 (5.969)
PI	Pn70K10; *see. Appendix for (near) optimal routes*		
20	[1,16,24,50]	14.095 (0.184)	14.095 (0.165)
50	[1,16,36,43,50,56,66]	9.286 (183.3)	11.57 (12.586)
100	[1,2,5,11,18,20,22,25,26,27,30,32,37,39]*		8.857 (11.783)
150	[1,4,9,15,23,43,49,56,59,69,70]*		7.012 (11.02)
200	[1,8,9,13,16,20,23,28,30,33,37,43, 50,52,66,69]*		5.512 (11.431)
PI	SQ1; *see. ref. [193] for (near) optimal routes*		
300	[1,5,6,16,21,22,34]	7.162 (2.247)	7.378 (18.313)
400	[1,5,6,16,21,22,26,34]	6.378 (17.703)	6.594 (14.705)
500	[1,2,6,8,15,17,21,22,34]	5.514 (382.70)	5.703 (9.244)
PI	SQ2; *see. ref. [193] for (near) optimal routes*		
300	[1,3,16,21,38,45,50,51]	8.0 (149.31)	8.342 (39.061)
400	[1,5,16,22,35,38,47,50,51]*		7.621 (35.866)
500	[1,5,6,17,26,40,45,47,50,51]*		7.018 (28.075)
PI	SQ3; *see. ref. [193] for (near) optimal routes*		
300	[1,6,16,25,28,66,67,68]*		8.852 (9.808)
400	[1,6,17,19,21,30,40,66,67,68]*		8.241 (9.971)
500	[1,4,5,16,38,45,63,64,66,67,68]*		7.684 (9.517)

Pn40K5 (near) optimal routes:

R.1:1, 12, 17, 30, 22, 35, 31, 10, 39, 1 **R.2:**1, 28, 9, 32, 27, 8, 24, 25, 7, 1

R.3:1, 13, 18, 38, 16, 34, 40, 11, 6, 1 **R.4:**1, 9, 5, 20, 14, 26, 15, 1

R.5:1, 2, 23, 29, 4, 37, 36, 21, 3, 33, 1

Pn70K10 (near) optimal routes:

R.1:1, 8, 36, 54, 15, 60, 20, 9, 1 **R.2:**1, 2, 44, 42, 43, 65, 23, 63, 69, 1

R.3:1, 12, 67, 66, 39, 59, 1 **R.4:**1, 4, 45, 25, 50, 57, 24, 64, 17, 52, 1

R.5:1, 11, 32, 56, 19, 51, 33, 1 **R.5:**1, 5, 32, 3, 34, 43, 65, 23, 63, 69, 7, 1

R.7:1, 18, 41, 10, 40, 13, 27, 1 **R.8:**1, 29, 62, 22, 70, 37, 48, 49, 1

R.9:1, 46, 28, 53, 35, 47, 68, 1 **R.10:**1, 30, 6, 38, 61, 21, 16, 58, 14, 55, 1

5.5.1 Accuracy

Table 5.3 compares the accuracy of the heuristics against optimal weighted average energy consumption. VRP routes of the problem instances are computed using a solver from the other research works [193]. The reference of the solution is provided for each instance. *Monitors* column depicts the heuristic solution corresponding to the budget as shown in column *Bgt.* The columns *Opt.* and *Heur.* depict the optimal and heuristically obtained minimum weighted average energy consumption respectively along with the computation time. Optimal solution generation time for few larger problem instances exceeds maximum run-time (12 hours) in an Intel *core i*7 machine, so they are omitted in Table 5.3. The parameters of the experiments are: (i) maximum sz size = 100000, (ii) $k = 5$ and (iii) $cnv = 3$. Max. iteration (ite) is 1000000. Table 5.3 demonstrates that the heuristic yields good-quality near-optimal solutions faster.

5.5.2 Performance

Table 5.3 indicates the benefits of the heuristics for large problem instances. However, no single set of parameter values can yield the best solutions for all problem instances. To analyze the trade-off between solution accuracy and computing time, Pn70K10 is further analyzed with budget 150. Figure 5.4 compares the minimum weighted average energy consumption against the elapsed computing time for various parameter combinations. Different cnv, k and sz combinations are tested. A large ite value is used to achieve convergence to near-optimality in all cases. Best solution is picked from 8 runs. Figure 5.4 depicts faster convergence for higher cnv values. In contrast, large k and sz produce better solutions but increase the time. In this problem instance,

Table 5.4: Distributed monitor selection on CVRP instances

PI	Bgt.	Avg.	Contributions
Pn23K8	(5,10,15,20)	3.93	1 [0,0,0,0]; 2 [0,0,4,3]; 12 [2,2,2,1]; 13 [1,5,4,4]; 14 [0,2,2,2]; 20 [2,2,1,1]; 23 [0,0,2,8]
	(10,20,30,40)	2.56	1 [0,0,0,0]; 2 [0,3,2,2]; 10 [2,2,2,2]; 11 [2,2,2,2]; 12 [0,1,3,3]; 14 [0,0,3,3]; 16 [0,4,4,3]; 19 [0,0,7,10]; 20 [2,2,1,1]; 21 [2,5,4,4]; 22 [2,1,1,1]; 23 [0,0,1,9]
Pn40K5	(5,10,15,20)	7.795	1[0,0,0,0]; 11[0,0,3,2]; 18[0,0,1,3]; 23[2,2,2,2]; 25[0,4,3,3]; 27[1,2,2,2]; 30[0,0,3,3]; 37[2,2,1,1]
	(10,20,30,40)	6.614	1[0,0,0,0]; 11[0,2,2,1]; 15[0,1,10,10]; 18[0,1,1,1]; 22[2,2,2,2]; 23[0,0,2,7]; 24[0,6,5,5]; 36[5,4,4,4]; 39[3,4,4,4]
Pn70K10	(5,10,15,20)	9.809	1[0,0,0,0]; 16[0,0,4,4]; 23[0,4,4,4]; 36[3,3,2,2]; 50[0,0,3,2]; 66[2,3,2,2]
	(10,20,30,40)	9.405	1[0,0,0,0]; 10[0,3,13,13]; 35[5,5,5,4]; 38[0,5,5,4]; 58[4,4,3,3]; 64[0,0,1,10]; 66[1,3,3,2]
	(20,50,80)	7.845	1[0,0,0]; 6[7,7,7]; 7[7,6,6]; 9[0,0,16]; 29[2,14,13]; 37[4,4,4]; 40[0,4,12]; 43[0,6,5]; 45[0,9,8]
	(10,20,40,80)	7.726	1[0,0,0,0]; 6[6,5,5,5]; 33[0,0,1,26]; 37[0,4,4,4]; 42[0,5,5,5]; 53[4,5,5,5]; 54[0,0,11,11]; 63[0,1,9,9]
	(20,30,40,50,60)	5.488	1[0,0,0,0,0]; 2[0,0,6,6,6]; 4[3,2,2,2,2]; 19[3,3,3,2,2]; 26[3,3,3,2]; 36[0,3,3,2,2]; 37[3,3,2,2,2]; 38[0,2,4,4,4]; 40[0,0, 1,8,8]; 43[0,3,3,3,2]; 50[1,1,1,1,1]; 52[3,3,2,2,2]; 53[0,0,0,6,12]; 64[0,3,3,3,2]; 66[0,0,3,3,3]; 69[2,2,2,2,2]; 70[2,2,2,1,1]
SQ1	(50,100,150)	7.324	1[0,0,0]; 3[20,20,20]; 5[20,20,20]; 8[10,25,25]; 21[0,30,30]; 22[0,5,55]; 34[0,0,0]
	(100,100,100)	7.351	1[0,0,0,0]; 3[20,20,20]; 17[28,28,29]; 18[30,30,30]; 24[20,20,20]; 34[0,0,0]
	(20,40,80,160)	7.13	1[0,0,0,0]; 3[0,15,22,22]; 19[0,0,3,55]; 24[0,0,30,30]; 28[20,25,25,25]; 34[0,0,0,0];
	(80,80,80,80,80)	6.594	1[0,0,0,0,0]; 3[12,12,12,12,12]; 8[12,12,12,12,12]; 14[12,12,12,12,12]; 17[17,17,17,17,17]; 21[12,12,12,12,12]; 22[12,12,12,12,12]; 34[0,0,0,0,0]
	(20,40,80,100,160)	7.108	1[0,0,0,0,0]; 5[0,0,20,20,20]; 6[0,8,18,18,18]; 16[0,0,11,21,21]; 17[17,17,17,17,17]; 21[3,15,14,14,14]; 34[0,0,0,0,0]
	(50,100,150,200)	6.081	1[0,0,0,0]; 3[13,16,16,15]; 6[0,20,20,20]; 8[0,8,26,26]; 17[22,21,21,21]; 21[0,20,20,20]; 22[15,15,15,15]; 24[0,0,30,30]; 34[0,0,0,0];

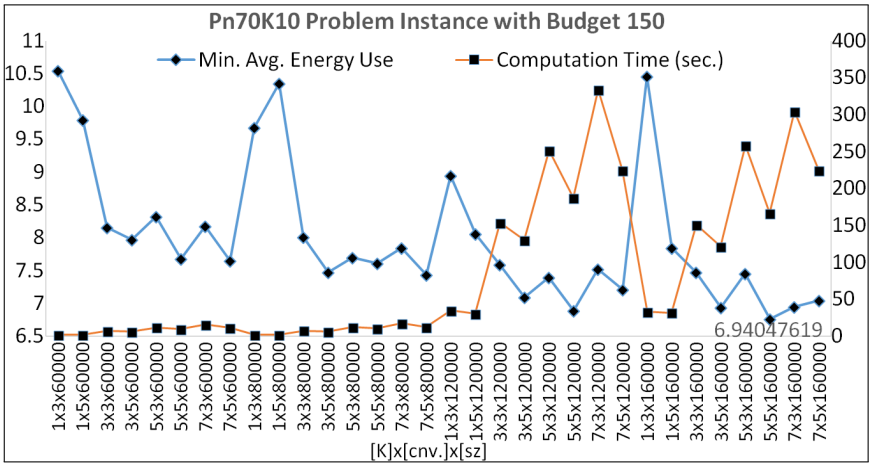


Figure 5.4: Performance comparison of heuristics with different parameters

parameters cnv=3, k=5 and sz=160000 provide the best trade-off.

5.5.3 Distributed Solutions

Table 5.4 presents experimental results from a few larger problem instances in distributed setup (with 3, 4 or 5 decision makers) where participants collaborate to deploy monitors. Individual budget of each participant and the resulting weighted average energy consumption are presented in columns *Bgt.* and *Avg.* Column *Contributions* shows the share of individual budgets. The number before bracket [] indicates a monitor while the comma separated values inside denote the contribution of each decision maker. The best solution is selected from 8 runs. Figure 5.5 shows a comparative analysis of the proposed algorithms for the first 3 problem instances in Table 5.4. The c-x (e.g. c-50) denotes a centralized setup to find optimal solution using exact algorithm with known budget 'x' over a problem instance. Likewise, h-x refers to the use of heuristics in centralized setup. In the distributed setup, the total budget is split among participants. For example, 5-10-15-20 denotes individual budgets for 4 decision makers. The column chart denotes weighted average energy consumption on primary y-axis. The secondary y-axis depicts the computation time via an area chart. The peak of the area denotes the solving time in seconds. Few experiments (e.g. c-100 for Pn40K5 PI) have been preemptively stopped after a 12 hour time limit. In general, the results indicate that the distributed technique is able to achieve similar accuracy compared to the centralized heuristic. However, its computation time is longer as the consensus generation needs multiple iterations among participants. Figure 5.5 shows that for a comparatively small total budget (of all decision makers), the negotiation takes longer to reach consensus. This indicates that

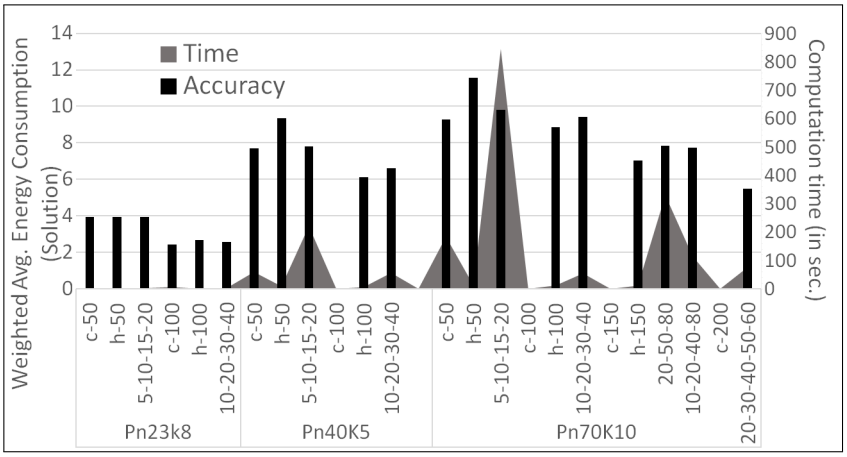


Figure 5.5: Comparative study of the proposed algorithms

the constrained distributed budget may lead to higher contribution mismatch among individually selected monitors.

5.5.4 Advantages and Limitations

In this chapter, we first elaborate a centralized heuristic technique to determine near-optimal monitor locations. Then we propose a collaborative monitor deployment technique where participating decision makers decide over the locations for monitors without explicitly sharing their available individual budgets to a central entity. Concerning the advantages and limitations, most importantly, the proposed approach offers an efficient procedure to near-optimally solve the monitor deployment problem in distributed setting. It also offers task decomposition by splitting the main problem along with the iterative use of heuristic technique locally at each participant side. In this regard, our experiments show that the technique produces good quality solutions. The approach allows each distributed participant to compute solutions of allocated sub-problems using less memory and computation time compared to centralized setting while using their own suitable parameter values to execute the heuristics.

However, overall solution generation for the MDP instances needs longer time with increasing number of participants and dissimilar available deployment budgets. Also, as the benchmark results indicate that the centralized setting produces better quality near-optimal solutions for MDP instances compared to the distributed setting.

5.6 Summary

This chapter elaborated a budget constrained monitor deployment problem. In this problem setup, distributed decision makers with individual budgets collaborate to minimize weighted average energy consumption in the data communications. A collaborative multi-round risk reduction approach was proposed along with a distributed heuristic technique to reach a near-optimal solution. As such, each participant determines monitor locations combining its own budget along with estimated deployment budgets for other participants. Over a number of iterations, each participant aims at reducing the error in the budget estimation of others. However, distributed data sharing requires addressing other aspects, such as secure communication, analysis of information gain, guarantee of honest participation, etc. All these aspects represent potential future work directions. Another relevant direction is to conduct a statistical analysis on energy consumption specific to various SCN deployments.

Chapter 6

Collaborative Plan Execution Monitoring Problem

Nowadays, numerous computing devices and sensors capture live local events and deliver pertinent information to distant data center(s) using advanced communication technologies. At the data center, event related information comes as data stream. So, it requires quick processing to extract complex interesting relationships among the events. Interesting cause-and-effect relationships among events can be reflected as association rules. In this chapter, we handle incremental mining of *interesting* association rules over the data stream. The proposed approach presents an efficient technique to update association rules through a prefix tree data structure. The latter stores a subset of simultaneously occurring events based on their frequencies of recent appearance. Furthermore, we also present a collaboration strategy whereby willing participants can jointly perform incremental generation of these association rules by partially storing the original prefix tree structure in their servers. The proposed approach is substantially less computation intensive in compare to previous efforts.

6.1 Introduction

As the next generation enterprises are transforming toward digital businesses [2], stream data analysis and mining are turning crucial for early detection of faults, performance measurement, trend analysis and other diagnostics. Frost and Sullivan, a renowned market research organization, predicts 3.5 times growth in mobile data monitoring market revenue from 2013 to 2020 [84]. A

recent Gartner report also declared that one fourth of the global firms will incorporate big data analytics by 2016 [72]. The applications of such analysis include remote monitoring of network performance, supply chain execution, oil-field and pipeline operations, health data industries, etc. [2]. Large cap companies, such as Accenture, Google, Microsoft, etc., also admit the necessity and importance of this paradigm shift to track and trace important business parameters [2, 112, 194].

Stream data requires quick processing to extract relevant events and their relationships. Let us consider an example of a monitoring system that offers alerts for possible delays on the delivery of urgent medical aid by an organization. In such a system, deployed devices and sensors continuously report traffic status, weather conditions, patient's medical condition and other information to the data center of the planning organization. This continuously reported data needs quick analysis, in an online manner, to prioritize deliveries and to avoid any possible delays. Therefore, analysis and mining of stream data may reveal hidden relations among recent events which, in turn, can help in decision making while executing business operations.

Classical data mining algorithms assume the presence of data in conventional databases where mining is performed centrally by accessing stored data multiple times and using powerful processors. There exists no strict time constraint to produce the output. In contrast, data streams are incessant and unbounded where timestamped events arrive at a high-speed. Thus, it requires an alternative methodology. Furthermore, the frequencies of various events are also expected to change over time which create a drift on the statistical properties of the events (also called *concept drift*) and their associations. Therefore, online mining of interesting and useful association rules exhibits great interest and complex challenges.

Data stream can be presented as a sequence of a timestamped set of simultaneous events which is also known as transaction. A set of events denotes an itemset where each item represents an event. So, a data stream can be treated as a temporal sequence of itemsets. The stream of distinct events is commonly treated as *moments* [190]. Moments help in understanding the distribution of frequencies for elements, analyzing stream properties and extracting stream related knowledge [190]. Figure 1.3 depicts an example of such a sequence of data stream transactions. Data streams are usually mined using landmark, damped or sliding window model. In a landmark model, data is captured from a defined point of time. In the damped model, newer transactions have higher contribution in the mining process. The sliding window model considers a transaction valid as long as it is fresh. The freshness is determined by a window size (denoted by τ) [48].

In this chapter, we investigate frequently associated events using Association Rule Mining (ARM). We propose a generic and efficient incremental stream mining approach to find selected association rules among the dependent events from a sliding window. The proposed approach

progressively extracts interesting association rules from a stream using a sliding window model. Furthermore, as depicted in Figure 1.3, we extend this approach to collaboratively extract these rules in multiple subsets using a number of servers. The servers collectively reduce the individual stream processing load and allow handling large number of events during a sliding window update. Few additional entities, namely helpers, are used to schedule the collection and mining of association rules in such distributed setting.

The existing research efforts, as reviewed in Section 2.2.3, only focus on the itemset mining which extracts frequent itemsets. In contrast, we present a hybrid technique that incrementally computes a large portion of interesting association rules by traversing a prefix tree structure. The rest of the rules are then iteratively generated from these incrementally generated rules. The incremental rule generation technique performs a selective depth-first search while pruning the prefix tree simultaneously. In distributed setting, collaborative participants partially mine sub-trees of this prefix tree (as constructed in centralized setting) to find association rules of individual interest. They locally execute the proposed hybrid technique over their respective sub-trees. In this respect, the contribution of this work can be stated as follows:

- We present a generic and efficient incremental data stream mining technique to find interesting association rules among co-occurring events.
- We present a distributed approach where multiple servers collaboratively participate in generating association rules.
- We experimentally show that the proposed techniques perform better than some existing techniques over various known datasets.

The rest of the chapter is organized as follows. In Section 6.2, we present the problem statement. Section 6.3 analyzes various properties and requirements for the generation of association rules. Section 6.4 presents the proposed approach in a centralized setting. Section 6.5 elaborates a collaboration technique to perform the association rule mining in multiple servers. The section also analyzes the scope of such distribution and associated redundancies. Section 6.6 provides benchmarks on the performance of the proposed approaches over known datasets. We summarize our findings in Section 6.7 along with future research directions.

6.2 Problem Statement

Let us assume that a finite set of all possible items (also called alphabet) is represented by $\mathcal{A} = \{e_1, e_2, \ldots, e_n\}$. Then, in a sliding window of size τ, a momentary association among two mutually

exclusive sets of items X and Y ($X, Y \subset \mathcal{A}$) is defined by two thresholds. First, $X \cup Y$ should appear at least s_{min} times within the most recent τ transactions. Second, $X \cup Y$ should appear at least c_{min} fraction of times X appears. These two thresholds, s_{min} and c_{min}, are named as minimum support and minimum confidence respectively. An association rule that satisfies both thresholds is known as strong association rule. In data mining, X and Y are called antecedent and consequent respectively. A strong association rule between X and Y is denoted by $X \rightarrow Y$. In event monitoring, we search specific association rules which satisfies another additional threshold. We interested in selected strong association rules where the product of individual occurrences of X and Y is less than the co-occurrences of $X \cup Y$ within most recent τ transactions. This threshold is known as $lift$. The $lift$ is a ratio between the confidence of the association rule over the unconditional probability of appearance for the consequent (Y). In this chapter, we investigate mining of association rules over stream data especially where $lift$ is greater than 1. We call them interesting association rules or interesting rules.

Wu et al. [243] elaborates momentary support and momentary confidence of an itemset over timestamped stream using a *lifetime* function l_j. At a timestamp t_j, l_j maps the domain of alphabet $\mathcal{A}$ to the co-domain of a set of timestamps (TS). The latter can be expressed as $\{t_i : j - \tau < i \leq j; t_i \in TS\}$. The output of function l_j is a set of timestamps which corresponds to the appearances of an element of $\mathcal{A}$ within most recent τ transactions. For example, considering $\tau = 10$, in Figure 1.3, $l_{10}(\text{a}) = \{t_3, t_5, t_7, t_9\}$ and $l_{10}(\text{b}) = \{t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}\}$. Using *lifetime* function momentary support (sup_j), momentary confidence ($conf_j$) and momentary lift ($lift_j$) of itemset X can be expressed as follows:

$$sup_j(X) = |\bigcap_{e \in X} l_j(e)|; X \subseteq \mathcal{A} \tag{6.1}$$

$$conf_j(X \rightarrow Y) = \frac{sup_j(X \cup Y)}{sup_j(X)}; X, Y \subset \mathcal{A} \tag{6.2}$$

$$lift_j(X \rightarrow Y) = \frac{\tau \times sup_j(X \cup Y)}{sup_j(X) \times sup_j(Y)}; X, Y \subset \mathcal{A} \tag{6.3}$$

From Eq. (6.1), it can be seen that $sup_j(X) \geq sup_j(X \cup Y)$. It is also called *apriori* property. Therefore, each interesting association rule must satisfy the following thresholds:

(i) $sup_j(X \cup Y) \geq s_{min}$.

(ii) $conf_j(X \rightarrow Y) \geq c_{min}$.

(iii) $lift_j(X \rightarrow Y) > 1$.

$X, Y \subset \mathcal{A}$ and $X \cap Y = \varnothing$.

We denote these interesting association rules as $X \xrightarrow{l} Y$. Subsequently, we denote invalid association rule and invalid interesting association rule among X and Y as $X \nrightarrow Y$ and $X \xrightarrow{l} Y$ respectively. If momentary support of an itemset is greater than or equal to s_{min} ($sup_j(X) \geq s_{min}$) itemset X is called *frequent*. Otherwise, itemset X is *infrequent*. If itemset X is present in all transactions of the sliding window, we call it *omnipresent*. Furthermore, if frequent itemset X has a superset $X \cup Y$ that has same support of X and there is no superset of $X \cup Y$ with the same support of $X \cup Y$, then we call X and $X \cup Y$ as *itermediate itemset* and *closed itemset* respectively. If X denotes a set of items identified as {a,b,c}, we often represent it as "abc". Similarly, $X \cup Y$ is referred as XY in short form throughout this chapter.

Association rules can be categorized based on the number of items in antecedent and consequent. A $[1-1]$ association rule refers to the relationship among two single items at the antecedent and the consequent which also respects thresholds (i) and (ii). Similarly, in an $[n-1]$ association rule, the antecedent and the consequent are a set of multiple items (up to n) and single item respectively. Accordingly, we may also categorize $[1-n]$ and $[n-n]$ association rules. Ideally $[n-n]$ association rules includes all $[1-1]$, $[n-1]$, $[1-n]$ rules as well. However, in this chapter, we depict them separately to explain the notion assuming $n > 1$.

6.3 Requirements Analysis

Table 6.1 presents an example of requirements for the minimum support of XY to generate rule $X \xrightarrow{l} Y$ over a sliding window of size 10 where the minimum support and minimum confidence are 3 and 0.7 (or 70%) respectively. The values are shown in four distinct regions of colors. The cell color gray with cell value 'F' indicates that no interesting rule can be found with the corresponding supports of X and Y in the given setting. The cell color green denotes that, in this setting, it

Table 6.1: Minimum requirements of $sup_j(XY)$ for supports of X and Y

		Y								$\tau: 10$, $s_{min}: 3$, $c_{min}: 0.7$
	sup_j	3	4	5	6	7	8	9	10	Evaluation Criteria
X	3	3	3	3	3	3	3	3	F	Threshold (i)
	4	3	3	3	3	3	4	4	F	
	5	F	4	4	4	4	5	5	F	
	6	F	F	5	5	5	5	6	F	Thresholds (i) & (ii).
	7	F	F	5	5	5	6	7	F	
	8	F	F	F	6	6	7	8	F	
	9	F	F	F	F	7	8	9	F	Thresholds (i), (ii) & (iii)
	10	F	F	F	F	F	F	F	F	

is enough to evaluate threshold (i) to form an interesting rule since threshold (ii) and (iii) are already satisfied. Similarly, cell color yellow indicates that, it requires satisfying threshold (i) and (ii) to form an interesting rule since threshold (iii) is already satisfied. Only the region marked with light-blue requires evaluating all three thresholds. In the incremental mining of interesting association rules identification of these regions is crucial for faster evaluation of a candidate rule.

6.3.1 Properties of Interesting Association Rules

To perform incremental mining of (interesting) association rules, we utilize the following properties of the rules. These properties, as observed from the aforementioned thresholds, reduce the scope of rule search and the requirements of evaluation between antecedent and consequent.

Given a set of items {a,b,c,d}, $conf_j(\{a,b,c\} \rightarrow \{d\}) \geq conf_j(\{a,b\} \rightarrow \{c,d\}) \geq conf_j(\{a\} \rightarrow \{b,c,d\})$. Thus, the confidence of rules generated likewise from the same itemset has an anti-monotonic relationship. Function $conf_j$ generates an anti-monotone with respect to the number of items at the right hand side of the rule. We may notice that {a,b,c} → {d} is an $[n-1]$ association rule. So, if $conf_j(\{a,b,c\} \rightarrow \{d\})$ does not hold, there exists no association rules among the items in {a,b,c,d} at the higher order $[n-2]$, $[n-3]$, etc. where the consequent is a superset of {d}. It significantly reduces the scope of the rule search. Property 1 mathematically captures our interest.

Property 1. *Given itemsets X, Y and Z, frequent itemsets X, XY, XZ and XYZ relate through an anti-monotonic relationship such that if $X \nrightarrow Y$ or $X \nrightarrow Z$ or $XY \nrightarrow Z$ or $XZ \nrightarrow Y$ then $X \nrightarrow YZ$.*

Proof. For an association rule $X \rightarrow YZ$, it requires to meet threshold (ii). If $\frac{sup_j(XYZ)}{sup_j(X)} \geq c_{min}$ then following constrains must be respected (using apriori property).

$\frac{sup_j^p(X \cup Y)}{sup_j^p(X)} \geq c_{min}$, $\frac{sup_j^p(X \cup Z)}{sup_j^p(X)} \geq c_{min}$, $\frac{sup_j^p(X \cup Y \cup Z)}{sup_j^p(X \cup Y)} \geq c_{min}$ and $\frac{sup_j^p(X \cup Y \cup Z)}{sup_j^p(X \cup Z)} \geq c_{min}$

The aforementioned four relations indicate four association rules $X \rightarrow Y$ or $X \rightarrow Z$ or $XY \rightarrow Z$ or $XZ \rightarrow Y$ respectively. Therefore, if one of them is invalid then $X \rightarrow YZ$ does not meet threshold (ii). □

Property 2 helps in pruning the scope of search for interesting association rules among frequent but non-omnipresent itemsets. In a short period of time, as captured by a sliding window, a number of events can be always seen in every transaction. For example, in practice, weather condition may remain hostile for a whole day. Property 2 indicates that such omnipresent events are not relevant to form interesting associations among two sets of recent events.

Property 2. *No association rule may have $lift$ greater than 1 if its antecedent and/or consequent itemset is omnipresent.*

Proof. An interesting association rule $X \xrightarrow{l} Y$ must satisfy threshold (iii). If $sup_j(X)$ or $sup_j(Y)$ is present in all transactions then its support is τ. Since, $sup_j(XY) \leq sup_j(X)$ and $sup_j(XY) \leq sup_j(Y)$. Then, $\frac{\tau \times sup_j(XY)}{sup_j(X) \times sup_j(Y)}$ cannot be greater than 1. □

An aim of measuring interest in a rule is to quantify current co-occurrence of events against random simultaneous occurrences. An intermediate itemset X which is a subset of its closed frequent itemset XY indicates such co-occurrence. Therefore, a relationship between X and Y is always considered as interesting association rule, as indicated by Property 3, unless X or Y is omnipresent. During the rule search among frequent itemsets, in a sliding window, Property 3 reduces the evaluation requirements for $X \xrightarrow{l} Y$ once we find an intermediate itemset X and its closed frequent itemset XY.

Property 3. *Given an intermediate itemset X and a closed frequent itemset XY that have same support, there exists always a rule $X \xrightarrow{l} Y$ if Y is not omnipresent.*

Proof. Since, X, Y and XY (apriori property) are frequent then it satisfies threshold (i). Since, $sup_j(X) = sup_j(XY)$ then it always satisfies threshold (ii). In this context, $lift_j(X \rightarrow Y)$ can be simplified as $\frac{\tau}{sup_j(Y)}$. Then, if Y is not present in all τ transactions, $lift_j(X \xrightarrow{l} Y) > 1$, which meets threshold (iii). □

Property 4, Property 5 and Property 6 extend the concept of Property 3. They dictate similar conditions to mine interesting association rules. Our main interest, in these contexts, is to make use of co-occurring events through these properties to improve the performance of the incremental rule search algorithms. Property 6*a.* offers a unique performance improvement for the proposed search. From Eq. 6.1, Eq. 6.2 and Eq. 6.3, it is clear that the evaluations of different thresholds require counting supports for the involved itemsets. In a large sliding window, counting support each time for various itemsets adversely impacts the search performance. In this context, Property 6*a.* significantly reduces the load of support evaluation for some itemsets.

Property 4. *Given intermediate itemsets X, XY and a closed non-omnipresent frequent itemset XYZ, all have same support, $X \xrightarrow{l} YZ$, $XY \xrightarrow{l} Z$, $XZ \xrightarrow{l} Y$, $X \xrightarrow{l} Y$ and $X \xrightarrow{l} Z$ are all valid interesting rules except the case(s) where consequent(s) are omnipresent.*

Proof. Using Property 1 and Property 3. □

Property 5. *Given an itemset X, an intermediate itemset XY and a non-omnipresent frequent closed itemset XYZ such that $sup_j(XY) = sup_j(XYZ)$, following can be found true:*

a. $X \xrightarrow{l} Y \Rightarrow X \xrightarrow{l} YZ$

b. $X \xrightarrow{l} YZ \Rightarrow X \xrightarrow{l} Y$

Proof. (*a.*) If $X \xrightarrow{l} Y$ meets thresholds (i), (ii) and (iii), Since $sup_j(XY) = sup_j(XYZ)$ then the relation $X \xrightarrow{l} YZ$ meets thresholds (i) and (ii). Now, $\frac{\tau \times sup_j(XY)}{sup_j(X) \times sup_j(Y)} \leq \frac{\tau \times sup_j(XYZ)}{sup_j(X) \times sup_j(YZ)}$ since $sup_j(Y) \geq sup_j(YZ)$ then $X \xrightarrow{l} YZ$ meets threshold (iii) also.

(*b.*) Similarly, if $X \xrightarrow{l} YZ$, it does not meet thresholds (i), (ii) or (iii). Since $sup_j(XY) = sup_j(XYZ)$, therefore, if $X \xrightarrow{l} YZ$ does not meet threshold (i) or (ii) then $X \xrightarrow{l} Y$ also does not meet the same threshold. Given, $\frac{\tau \times sup_j(XYZ)}{sup_j(X) \times sup_j(YZ)} \geq \frac{\tau \times sup_j(XY)}{sup_j(X) \times sup_j(Y)}$, if $X \xrightarrow{l} YZ$ does not satisfy threshold (iii) then $X \xrightarrow{l} Y$ also does not satisfy the same. □

Property 6. *Given an intermediate itemset X and a closed frequent itemset XY such that $sup_j(X) = sup_j(XY)$, the following can be found true with itemset XYZ:*

a. $sup_j(XZ) = sup_j(XYZ)$

b. $XY \xrightarrow{l} Z \Rightarrow X \xrightarrow{l} YZ$

c. $X \xrightarrow{l} YZ \Rightarrow XY \xrightarrow{l} Z$ *and* $X \xrightarrow{l} YZ \Rightarrow X \xrightarrow{l} Z$

Proof. (*a.*) Given $sup_j(X) = sup_j(XY)$, Y occurs in every transaction where X occurs over the sliding window t_j. Now, if X and Z co-occur only among a subset these transactions then $sup_j(XZ) = sup_j(XYZ)$.

(*b.*) $XY \xrightarrow{l} Z$ meets thresholds (i), (ii) and (iii). Since $sup_j(X) = sup_j(XY)$ then relation $X \xrightarrow{l} YZ$ meets thresholds (i) and (ii). Now, $\frac{\tau \times sup_j(XYZ)}{sup_j(XY) \times sup_j(Z)}$ is less or equal to $\frac{\tau \times sup_j(XYZ)}{sup_j(X) \times sup_j(YZ)}$ since $sup_j(Z) \geq sup_j(YZ)$. Thus, interesting rule $X \xrightarrow{l} YZ$ meets threshold (iii) also.

(*c.*) Similarly, if $X \xrightarrow{l} YZ$, it does not meet threshold (i), (ii) or (iii). Since $sup_j(X) = sup_j(XY)$, if $X \xrightarrow{l} YZ$ does not meet threshold (i) or (ii) then $X \xrightarrow{l} Y$ also does not meet the same threshold. Since $\frac{\tau \times sup_j(XYZ)}{sup_j(X) \times sup_j(YZ)} \geq \frac{\tau \times sup_j(XYZ)}{sup_j(XY) \times sup_j(Z)}$, if $X \xrightarrow{l} YZ$ does not satisfy thresholds (iii), then $XY \xrightarrow{l} Z$ also does not meet the same. Likewise, since $sup_j(XZ) = sup_j(XYZ)$ and $sup_j(Z) \geq sup_j(YZ)$, $X \xrightarrow{l} YZ$ also implies $X \xrightarrow{l} Z$. □

Now, if a set of items (Y) occurs less than $\tau \times c_{min}$ in a sliding window and yet forms association rule $X \rightarrow Y$ with itemset X, we consider such an association rule interesting. Property 7 indicates the reason behind. From the perspective of rule search, the property is important

since it reduces the evaluation requirement of threshold (iii) where the consequent is less than a particular constant value. Table 6.1 supports the property as one may notice that where the support of consequent Y is less than 7 (10×0.7), evaluation criteria does not require checking threshold (iii).

Property 7. *Given itemsets X, Y and their frequent superset XY, rule $X \xrightarrow{l} Y$ is valid if X is not omnipresent, $conf_j(X \xrightarrow{l} Y) \geq c_{min}$ and $sup_j(Y) < \tau \times c_{min}$.*

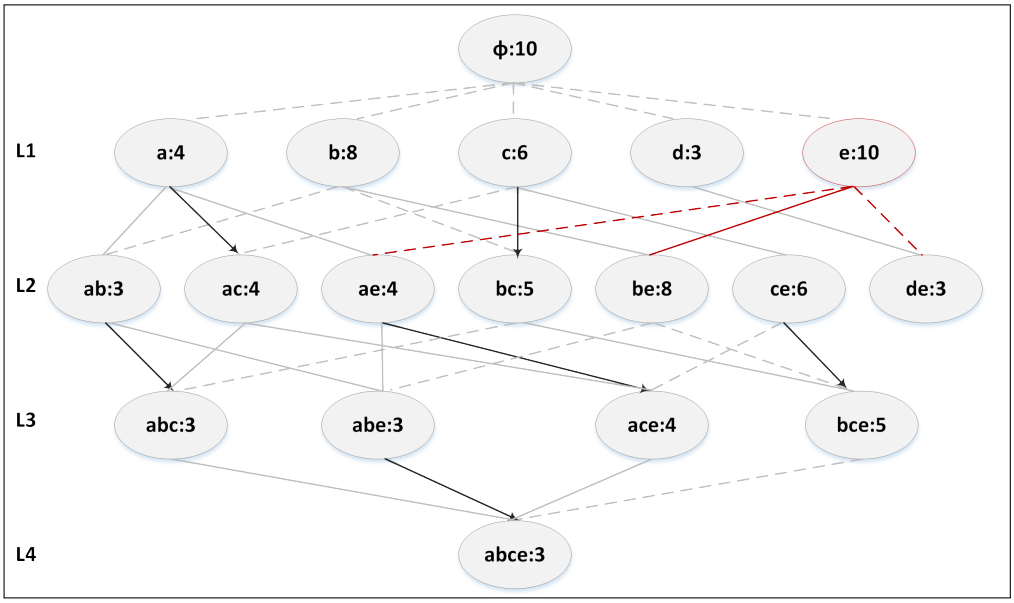
Proof. Given X, Y and XY are frequent, $sup_j(XY)$ meets threshold (i). Since $conf_j(X \xrightarrow{l} Y) \geq c_{min}$, it is a valid association rule. Finally, given, $sup_j(Y)$ is less than $\tau \times c_{min}$, $sup_j(Y)$ is also less than $\tau \times \frac{sup_j(XY)}{sup_j(X)}$ since $\frac{sup_j(XY)}{sup_j(X)} \geq c_{min}$. Then, $\frac{\tau \times sup_j(XY)}{sup_j(X) \times sup_j(Y)} > 1$ as support of X cannot exceed τ. This satisfies threshold (iii), i.e., $lift_j(X \xrightarrow{l} Y) > 1$. □

6.3.2 Identification of Interesting Association Rules

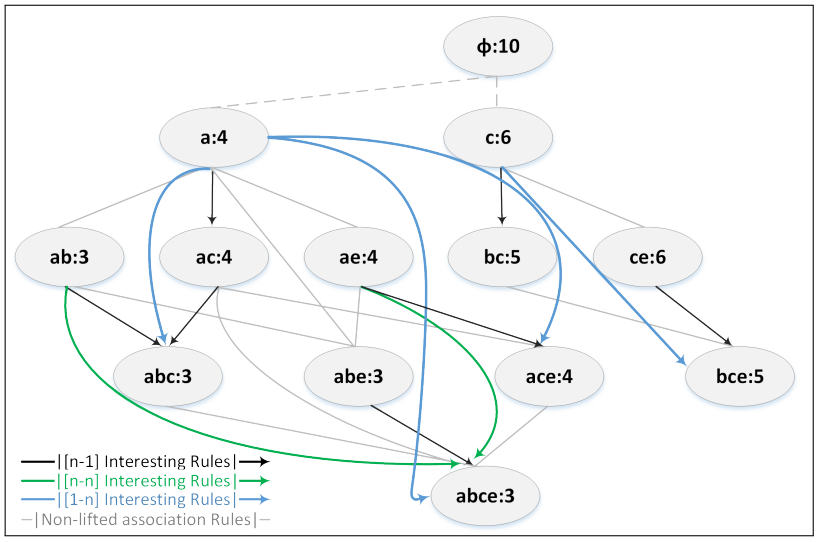
The relations between various itemsets within an alphabet can be presented using lattice [136]. A lattice is a fundamental and general algebraic structure to represent a partially ordered set which is often drawn using Hasse diagram[1]. A lattice is denoted by $\langle L, \vee, \wedge \rangle$ where L is a non-empty set that supports binary OR and binary AND operations over L. From an alphabet $\mathcal{A}$, a lattice can be derived using a partially-ordered set $(L, \preceq)$ by considering $X \wedge Y = X \cap Y$ and $X \vee Y = X \cup Y$ for any $X, Y \in L$. In lattice theory, $X \wedge Y$ is called as infimum, meet or greatest lower bound. Similarly, $X \vee Y$ is termed as supremum, join or least upper bound. So, the lattice for $\mathcal{A}$ contains all possible subsets of $\mathcal{A}$. Thus, a full lattice representation with its all feasible itemsets can be extremely large. Storing such a lattice is memory consuming and often unnecessary in our context. So, we prune the lattice using support threshold by removing itemsets that have support less than minimum support threshold. The resulting structure is a partial lattice or meet-semilattice [136]. A meet-semilattice only respects the meet (or greatest lower bound) constraint in the definition of lattice.

Figure 6.1 depicts a Hasse diagram of a meet-semilattice of itemsets and their support from the first sliding window of the example presented in Figure 1.3. The meet-semilattice is presented for minimum support 3 and minimum confidence 0.7. The root of the lattice is denoted as $\varnothing$ in similarity to the common practice. Level 1 ($L1$) at Figure 6.1(a) presents all five single items of the alphabet in a strictly alphabetical order. Then, at each level, frequent itemsets of size of its level are captured using the same alphabetical order along with their supports in the sliding window. The infrequent itemsets or their supersets are not captured since a superset of an infrequent itemset is

[1] https://en.wikipedia.org/wiki/Hasse_diagram



(a) Meet-semilattice of frequent itemsets with interesting $[n-1]$ rules



(b) Interesting $[n-1]$, $[n-n]$ and $[1-n]$ association rules with $lift > 1$

Figure 6.1: Interesting rules generated from the first sliding window of Figure 1.3

also infrequent (apriori property) and they cannot form an association rule. Itemset {e} is circled in red, since it is omnipresent and thus cannot directly be antecedent or consequent of any association rule having $lift$ greater than 1. However, supersets of {e} should be evaluated as antecedent or consequent of a rule if their supports are less than τ. It should be noted that, supports of {a}, {c} and {d} are less than 7 ($\tau \times c_{min}$). Using Property 7, every association rule having consequent {a},

{c} and {d} or any of their supersets has $lift$ greater than 1 unless they violates other properties. A dotted straight line between every two itemsets of subsequent levels indicate failure to satisfy threshold (ii). A continuous straight line is drawn otherwise. A black continuous arrow line shows the $[n-1]$ interesting association rules (containing $[1-1]$, $[2-1]$ and $[3-1]$ rules) between two successive levels. At Figure 6.1(b), we evaluate feasible [1-n] and [n-n] interesting rules similarly between every other levels among frequent itemsets that satisfy thresholds (i), (ii) and (iii). We indicate them using green continuous arrow lines. We also find a [1-n] interesting rule {a} → {b,c,e} between L1 and L4. Over a meet-semilattice, itemsets and rules can be searched using breadth-first, depth-first or a hybrid search. In this chapter, we have decided to use a hybrid depth-first search of rules since it may better take advantage of the properties during the traversal to shorten the search time. Figure 6.1 depicts two [1-1] interesting rules {a} $\xrightarrow{l}$ {c} and {c} $\xrightarrow{l}$ {b}; three [2-1] interesting rules: {a,b} $\xrightarrow{l}$ {c}, {a,e} $\xrightarrow{l}$ {c} and {c,e} $\xrightarrow{l}$ {b}, one [3-1] interesting rule: {a,b,e} $\xrightarrow{l}$ {c}, two [2-2] interesting rules: {a,b} $\xrightarrow{l}$ {c,e} and {a,e} $\xrightarrow{l}$ {b,c}, three [1-2] interesting rules: {a} $\xrightarrow{l}$ {b,c}, {a} $\xrightarrow{l}$ {c,e}, {c} $\xrightarrow{l}$ {b,e} and one [1-3] interesting rule: {a} $\xrightarrow{l}$ {b,c,e}.

6.3.3 Update of Interesting Association Rules

Table 6.2 depicts a set of general requirements of incremental update for interesting association rules. There exist 12 update settings as identified by U1-U12 based on increase and/or decrease of two mutually exclusive itemsets namely X and Y along with their joint appearance, denoted by XY, in the current sliding window. There can be one of two scenarios between X, Y and XY based on whether they have already formed a rule or not. As the sliding window is updated, support of X, Y and XY may increases by 1, decreases by 1 or remains unaffected. Assuming X, Y and XY frequent in the current and next sliding windows, each row reflects the evaluation requirements of X, Y and XY for maintaining the current condition or changing it in one of the 12 update settings. The big brackets $\big(\ldots\big)$, in Table 6.2, divide the evaluation requirements for an appropriate update setting between requirements for satisfying confidence and lift respectively. The symbol ς denotes the current ratio of support for XY over the support for X, i.e. $conf_j(X \rightarrow Y)$.

Let us consider that there exists an interesting rule between X and Y in the current sliding window τ_j. Now if the support of X decreases in the next sliding window (τ_{j+1}) while the support of Y and XY remain same, Table 6.2 shows that the existing interesting rule $X \xrightarrow{l} Y$ is still valid in the next sliding window and does not require any further evaluation. In contrast, if the support of X increases in the next sliding window while the supports of Y and XY remain same, the existing interesting rule requires two evaluations. If the current support of X is less than $\frac{c_{min}}{\varsigma - c_{min}}$ or the current support of Y is no greater to $\tau\varsigma - \frac{\tau\varsigma}{sup_j(X)+1}$ then, at the next sliding window τ_{j+1}, the

Table 6.2: Incremental update requirements of selected association rules

Scenario:	$X \overset{l}{\nrightarrow} Y$	$X \overset{l}{\rightarrow} Y$
Update	Requirement	Requirement
U1 $sup_j(X)$ decreased	$\left(\varsigma \geq c_{min} \text{ or } sup_j(X) \leq \frac{c_{min}}{c_{min}-\varsigma}\right)$ and $\left(sup_j(Y) < \tau\varsigma + \frac{\tau\varsigma}{sup_j(X)-1}\right)$	rule unchanged
U2 $sup_j(X)$ increased	rule unfeasible	$\left(sup_j(X) < \frac{c_{min}}{\varsigma-c_{min}}\right)$ or $\left(\tau\varsigma - \frac{\tau\varsigma}{sup_j(X)+1} \leq sup_j(Y)\right)$
U3 $sup_j(Y)$ decreased	$\left(\varsigma \geq c_{min}\right)$ and $\left(sup_j(Y) < \tau\varsigma + 1\right)$	rule unchanged
U4 $sup_j(Y)$ increased	rule unfeasible	$\tau\varsigma - 1 \leq sup_j(Y)$
U5 $sup_j(X)$, $sup_j(XY)$ decreased	rule unfeasible	$\left(sup_j(X) < \frac{1-c_{min}}{\varsigma-c_{min}}\right)$ or $\left(\tau\varsigma - \frac{\tau(1-\varsigma)}{sup_j(X)-1} \leq sup_j(Y)\right)$
U6 $sup_j(X)$, $sup_j(XY)$ increased	$\left(\varsigma \geq c_{min} \text{ or } sup_j(X) \leq \frac{1-c_{min}}{c_{min}-\varsigma}\right)$ and $\left(sup_j(Y) < \tau\varsigma + \frac{\tau(1-\varsigma)}{sup_j(X)+1}\right)$	rule unchanged
U7 $sup_j(Y)$, $sup_j(XY)$ decreased	rule unfeasible	$\left(sup_j(X) < \frac{1}{\varsigma-c_{min}}\right)$ or $\left(\tau\varsigma + 1 - \frac{\tau}{sup_j(X)} \leq sup_j(Y)\right)$
U8 $sup_j(Y)$, $sup_j(XY)$ increased	$\left(\varsigma \geq c_{min} \text{ or } sup_j(X) \leq \frac{1}{c_{min}-\varsigma}\right)$ and $\left(sup_j(Y) < \tau\varsigma - 1 + \frac{\tau}{sup_j(X)}\right)$	rule unchanged
U9 $sup_j(X)$ decreased, $sup_j(Y)$ increased	$\left(\varsigma \geq c_{min} \text{ or } sup_j(X) \leq \frac{c_{min}}{c_{min}-\varsigma}\right)$ and $\left(sup_j(Y) < \tau\varsigma - 1 + \frac{\tau\varsigma}{sup_j(X)-1}\right)$	$\left((sup_j(X) > sup_j(Y) + 1) \text{ and } (\tau\varsigma - 1 + \frac{\tau\varsigma}{sup_j(X)-1} \leq sup_j(Y))\right)$
U10 $sup_j(X)$ increased, $sup_j(Y)$ decreased	$\left(\varsigma \geq c_{min} \text{ and } sup_j(X) \geq \frac{c_{min}}{\varsigma-c_{min}}\right)$ and $\left(sup_j(Y) < \tau\varsigma + 1 - \frac{\tau\varsigma}{sup_j(X)+1}\right)$	$\left(sup_j(X) < \frac{c_{min}}{\varsigma-c_{min}}\right)$ or $\left((sup_j(X) < sup_j(Y) - 1) \text{ and } (\tau\varsigma + 1 - \frac{\tau\varsigma}{sup_j(X)+1} \leq sup_j(Y))\right)$
U11 $sup_j(X)$, $sup_j(Y)$ decreased	$\left(\varsigma \geq c_{min} \text{ and } sup_j(X) \geq \frac{1-c_{min}}{\varsigma-c_{min}}\right)$ and $\left(sup_j(Y) < \tau\varsigma + 1 - \frac{\tau(1-\varsigma)}{sup_j(X)-1}\right)$	$\left(sup_j(X) < \frac{1-c_{min}}{\varsigma-c_{min}}\right)$ or $\left(\tau\varsigma + 1 - \frac{\tau(1-\varsigma)}{sup_j(X)-1} \leq sup_j(Y)\right)$
U12 $sup_j(X)$, $sup_j(Y)$ increased	$\left(\varsigma \geq c_{min} \text{ or } sup_j(X) \leq \frac{1-c_{min}}{c_{min}-\varsigma}\right)$ and $\left(sup_j(Y) < \tau\varsigma - 1 + \frac{\tau(1-\varsigma)}{sup_j(X)+1}\right)$	$\left(\tau\varsigma - 1 + \frac{\tau(1-\varsigma)}{sup_j(X)+1} \leq sup_j(Y)\right)$

existing interesting rule $X \overset{l}{\rightarrow} Y$ is no longer valid. Now, τ and c_{min} are predefined values while the supports for X, Y and ς change from one sliding window to other. Table 6.2 clearly demonstrates that the proposed search procedure requires evaluating support for X, Y and ς, if needed, with every progression of the sliding window. So, next, we focus on devising a single pass traversal over a stored meet-semilattice to reveal interesting association rules.

6.4 Centralized Association Rule Mining

Rajaraman and Ullman [190] have mentioned that handling the speed of stream is a key challenge for mining rules using high-complexity mining algorithm(s). In addition, associated data structures

should also be managed prudently to quickly trace necessary information within a bounded memory. In what follows, we propose an in-memory mining procedure to capture support of relevant items using a bit matrix and a prefix tree over the sliding window.

6.4.1 Itemset Scanning

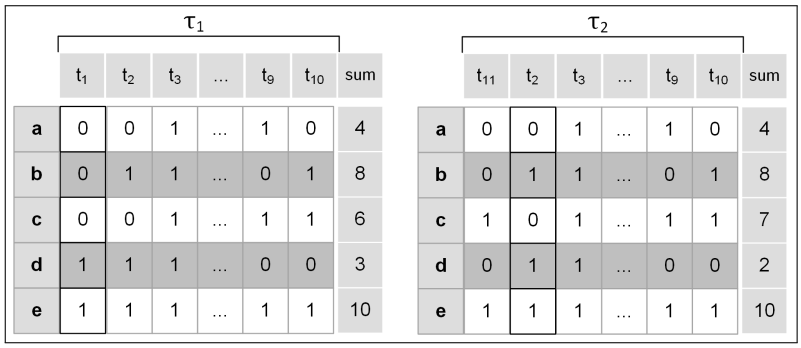


Figure 6.2: Transactions in a bit matrix over sliding window

Transactions can be stored in horizontal or vertical layout. In a horizontal layout, each row represents a transaction of items. Apriori-like algorithms often apply such layout to extract frequent itemsets [211]. In the vertical layout, each row represents all occurrences of an event in every valid transactions over the sliding window. It can be stored as a bit string. Algorithms using vertical layout generally perform faster than horizontal ones when the sliding window size is large [211]. Figure 6.2 depicts an incremental insertion of transactions from Figure 1.3 using a bit matrix where data is stored in a vertical layout. Given a strictly predefined order of items in an alphabet (e.g. a, b,⋯, e), occurrences of every item in a sliding window are stored as a separate bit array through 1s and 0s. Items from every new transaction are stored column-wise at a column indicated by a *sliding pointer*. In Figure 6.2, this designated column is marked in dark black squares. The cell value 1 in a designated column denotes the presence of the corresponding item at a transaction. Otherwise, cell value is kept 0. Once all transactions of a sliding window are filled, the sliding pointer denotes the oldest valid transaction which gets replaced by the new transaction. This matrix is used to compute momentary support of an itemset. The computation procedure is called *scanning*. To find support of an input itemset, scanning simply generates a new bit array of the same size of sliding window using bit-wise "AND" operation among the rows corresponding to every item from the input itemset. Then, it counts total number of 1s over the output bit array. Scanning can be performed sequentially or hierarchically. When the sliding window size is large, multiple bit arrays store the occurrences of an item. Thus, sequentially counting support can be

slower. Hierarchical counting saves execution time and reaches time-complexity of approximately $O(log(\tau))$ using parallel processing. In addition, the total appearance of every item is separately kept in an integer array, namely "sum". The array is updated with every new transaction.

The aforementioned scanning is efficient but it does not keep momentary support of any itemset beyond the scanning procedure. Thus, with every new transaction, it requires applying apriori or similar technique(s) to find frequent itemsets and thereafter computing association rules. For an alphabet $\mathcal{A}$ of size n, total possible itemsets and association rules are $2^n - 1$ and $3^n - 2^{n+1} + 1$ respectively. In this regard, storing momentary support of relevant itemsets offers faster computation of association rules. Therefore, we propose mining support for selective itemsets in a prefix tree variant for a set of preferred itemsets.

6.4.2 Data Structure

We introduce *Partial Association Enumeration Tree* (PAET) as a prefix tree variant that mines relevant itemsets and offers faster search for momentary (interesting) association rules.

Definition 1. *Partial Association Enumeration Tree: Over a sliding window of size τ, given a strictly ordered set of alphabet $(\mathcal{A}, \prec)$ and a root node $\langle \varnothing, \tau, - \rangle$, PAET can be defined recursively as a collection of nodes starting from the root. Each PAET node n_X is a triplet, denoted by $\langle X, sup_j(X), n_{X \setminus \{e\}} \rangle$, which consists of an itemset X ($X \in \mathcal{P}(\mathcal{A}) \setminus \{\varnothing\}$), its support $sup_j(X)$ in last τ transactions and a pointer to a node $n_{X \setminus \{e\}}$. Node n_X satisfies the following:*

- $|X| = 1$, *or* $sup_j(X) \geq s_{min} - 1$
- *If* n_X *has pointer to* $n_{X \setminus \{e\}}$ *then* $\forall e, e' \in \mathcal{A}$, $sup_j(X \setminus \{e\}) \leq \min\limits_{e' \in \mathcal{A}} sup_j(X \setminus \{e'\})$.

$\mathcal{P}(\mathcal{A})$ denotes the powerset of the alphabet $\mathcal{A}$ while $\prec$ and $\preceq$ indicate strict and partial ordering respectively. PAET keeps all single items in the first level just below the root and thus reflects the "sum" integer array (*See.* Section 6.4.1) in our implementation. Additionally, it keeps every node where its current support is more or equal to $s_{min} - 1$. The pointer associated to node n_X always tracks one of its parent nodes having the lowest support. Figure 6.3 depicts a PAET generation from the first sliding window of Figure 1.3.

6.4.3 Gateway Analysis

Incremental association rule mining over a reasonable size of sliding window needs the PAET to be constructed in memory for faster access [48] where every tree node represents an itemset. The tree size and the node selection require special attention for incremental insertion and deletion.

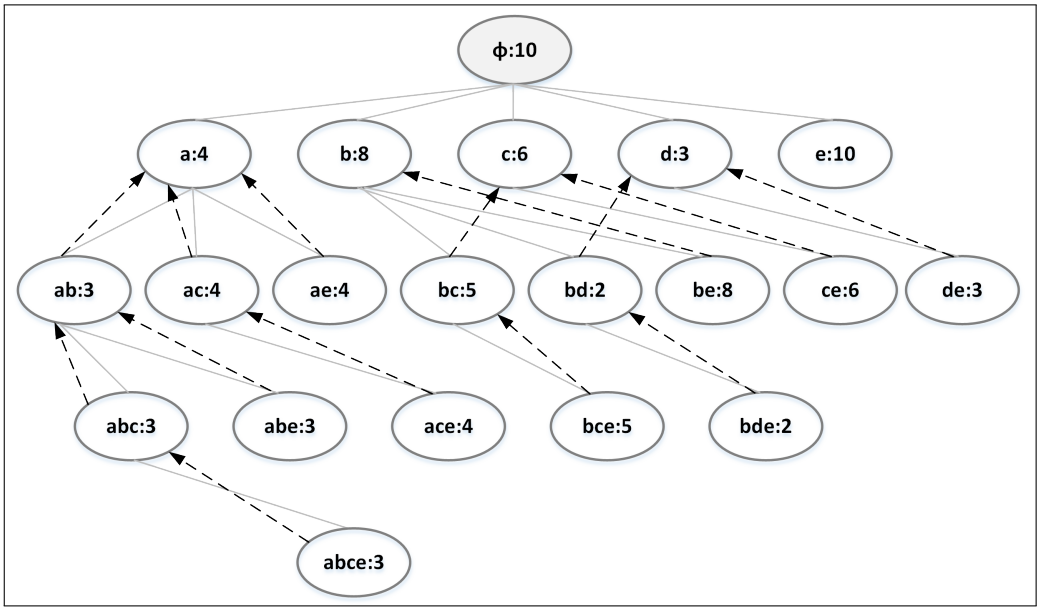


Figure 6.3: PAET generated from the first sliding window of Figure 1.3

The core challenge lies in constructing the tree to efficiently search selected association rules in every new window. To analyze an interesting association rule $X \xrightarrow{l} Y$ quickly, momentary support of node X, Y or XY should be accessed from the tree. Otherwise, it requires additional scanning to compute support of needed itemset(s) in order to validate a rule with every new transaction in the sliding window. This, in turn, makes the overall rule generation slower. So, we propose inserting tree nodes representing itemsets at least one transaction before they can possibly meet three thresholds of rule generation. Chi et al. have termed these nodes as Closed Enumeration Tree (CET) gateway [48] while describing their tree data structure. In particular, the conditions for insertion and deletion of new itemset (of size greater than 1) in a PAET are as follows:

- *Insertion*: Node n_X corresponding to Itemset X is inserted in PAET if X is a subset of current incoming transaction and it meets one of these two conditions:
  (a) *Leaf Insertion*: The momentary support of X is computed greater or equal to $s_{min} - 1$, i.e. $sup_j(X) \geq s_{min} - 1$, or
  (b) *Branch Insertion*: A superset XY of X is found such that $sup_j(XY) \geq s_{min} - 1$.
- *Deletion*: Node n_X corresponding to Itemset X can be deleted from PAET if support of X is reduced at the current sliding window and it meets one of these two conditions:
  (a) *Leaf Deletion*: X denotes a leaf node and the momentary support of X is computed

less than $s_{min} - 1$, i.e. $sup_j(X) < s_{min} - 1$, or

(b) *Branch Deletion*: There exists a subset of X, identified as $X \setminus Y$, in PAET where $sup_j(X \setminus Y) < s_{min} - 1$.

Notably, single items are never added or deleted at the tree. Only, their supports are updated with every window update, if required. In this approach, node insertion and deletion differ from the common approaches of stream mining for frequent itemsets [42, 48, 223]. The itemset is stored in PAET if and only if it can possibly be an antecedent or consequent of an association rule in the next transaction. Leaf insertion or deletion deals with one node while branch insertion or deletion handles multiple nodes. In branch insertion, supports of all new nodes are required to be evaluated. Property 6*a*. is used to reduce the total number of evaluations using the scanning procedure.

6.4.4 Maximum Confidence Analysis

The confidence of an interesting association rule changes due to the update of the supports of antecedent, consequent and their joint appearances. Therefore, in addition to mine the support of each itemset X, we propose tracking their parent nodes also. The itemset represented by these parent nodes can potentially generate $[n-1]$ association rule(s) with X. In order to track these nodes, we use the pointer of the PAET node n_X to point to one of its parent nodes that has minimum support among all its parent nodes. These pointers help in determining the maximum confidence for $[n-1]$ association rules involving two mutually exclusive subsets of X whose union is X itself. We elaborate the concept using a term Maximum Confidence Rule (MCR).

Definition 2. $[n-1]$ *Maximum Confidence Rule (MCR): An association rule* $X \rightarrow Y$ *is called* $[n-1]$ *maximum confidence rule of* XY *if and only if its antecedent* $X \subset XY$, $|XY| - |X| = 1$ *and* $conf_j(X \rightarrow Y)$ *has the maximum value of confidence in compare with any antecedent* X' $(X' \subset XY)$ *and consequent* $XY \setminus X'$.

Let us consider itemset ACE of cardinality 3 in Figure 6.3. In the meet-semilattice, its parents of cardinality 2 are {a,c}, {a,e} and {c,e} with support 4, 4 and 6 respectively. Therefore, either {a,c} → {e} or {a,e} → {c} can be considered as a $[n-1]$ MCR for itemset ace. Similarly, {c} → {b} is an $[1-1]$ MCR for itemset {b,c}. Extending the definition, {a,b} → {c,e} is a $[n-2]$ MCR for {a,b,c,e}. The following conditions reflect the importance of a $[n-1]$ MCR during a sliding window update.

C1: *If* $[n-1]$ *MCR of* XY *does not meet the minimum confidence threshold, no association rule can be constructed where the union of antecedent and consequent itemsets form* XY.

C2: *If* $sup_j(X) = sup_j(XY)$ *then* $X \rightarrow Y$ *is a* $[n-1]$ *MCR of* XY *unless there already exists another* $[n-1]$ *MCR of* XY, $X' \rightarrow Y'$, *where* $X' \cup Y' = XY$ *and* $sup_j(X') = sup_j(XY)$.

C3: *If* $X \rightarrow Y$ *is a* $[n-1]$ *MCR of* XY, *increase in support of any subsets of* XY *except* X *holds* $X \rightarrow Y$ *as MCR of* XY *in the next sliding window.*

C4: *If* $X \rightarrow Y$ *is a* $[n-1]$ *MCR of* XY, *decrease in support of* X *holds* $X \rightarrow Y$ *as MCR of* XY *in the next sliding window.*

C5: *If* $X \rightarrow Y$ *is a* $[n-1]$ *MCR of* XY, *as tracked by the pointer of PAET node* n_{XY}, *then increase, decrease or no change in support at all the parent nodes of* n_{XY} *together still holds* $X \rightarrow Y$ *as MCR of* XY *in the next sliding window.*

6.4.5 Incremental Update of Support and MCR

The incremental update of a PAET node occurs from one of the four input types: increase (+), decrease (-), no change (0) or no impact (N) for any node in PAET. We call them transition triggers denoted by a set IP. A traversal over PAET can be captured through these transition triggers among a set of states. The following example elaborates the concept.

Let ξ_- and ξ_+ denote the oldest and the newest transactions. A powerset $\mathcal{P}(\xi_-)$ indicates all itesets affected by the outgoing transaction. For example, in Figure 1.3, at sliding window τ_2, ξ_- represents the outgoing transaction {de}. Therefore, $\mathcal{P}(\xi_-)$ represents $\{\varnothing, \{d\}, \{e\}, \{d,e\}\}$. Similarly, $\mathcal{P}(\xi_+)$ is $\{\varnothing, \{c\}, \{e\}, \{c,e\}\}$. A simple analysis reveals that there is no change of support for all itemsets represented by $\mathcal{P}(\xi_- \cap \xi_+)$. In this example, it is itemset {e}. However, the momentary support increases for all itemsets denoted by $\mathcal{P}(\xi_+) \setminus \mathcal{P}(\xi_- \cap \xi_+)$ (e.g. $\{\{c\}, \{c,e\}\}$). Conversely, the momentary support decreases for all itemsets denoted by $\mathcal{P}(\xi_-) \setminus \mathcal{P}(\xi_- \cap \xi_+)$ (e.g. $\{\{d\}, \{d,e\}\}$). Also, there exists a large number of itemsets in the powerset of alphabet $\mathcal{A}$, denoted by $\mathcal{P}(\mathcal{A}) \setminus \mathcal{P}(\xi_- \cup \xi_+)$, which are not impacted by the current update of the sliding window.

As the support of an itemset is updated, depending on various changes of its items, different actions are required to evaluate current rules and form the new ones. We find 15 different states (excluding initial and end state) which require various actions in relation to track change of support and $[n-1]$ MCR using the pointers. Figure 6.4 depicts all these states through a hybrid automaton. Every state is identified uniquely by a label S_i and a set of symbols from IP. For example, state S_3 inherits symbols $(+0)0^*$. The symbol $(+0)0^*$ means that the corresponding itemset in PAET has exactly one item whose support is increasing and at least one item whose support remains same as it appears in both the outgoing and incoming transactions. The input legend of Figure 6.4 explains all the symbols in details.

These 15 states can be categorized as one of the four groups as shown in Figure 6.4:

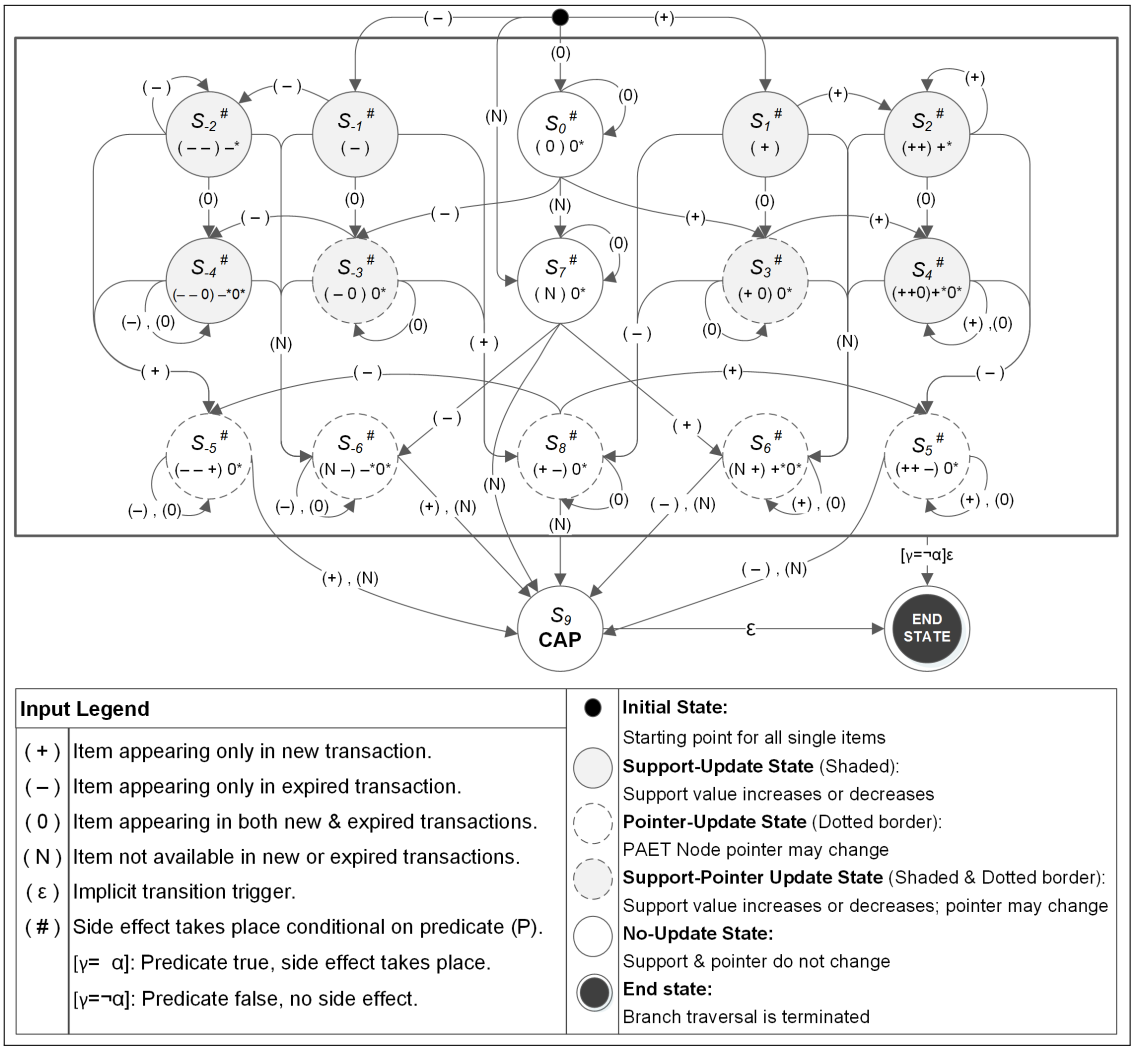


Figure 6.4: States and transitions for incremental update using *Hybrid Automaton*

- *Support-Update States*: In these states, support of X for a PAET node n_X changes but no pointer change is necessary since supports of all parent nodes of n_X either increase or decrease together (Condition C5). Representative states are S_1, S_2, S_4, S_{-1}, S_{-2} and S_{-4}.
- *Pointer-Update States*: In these states, support X for the PAET node n_X does not change but the support of its one or multiple parent nodes changes. This leads to evaluating the change for the pointer of n_X. Representative states are S_5, S_6, S_8, S_{-5} and S_{-6}. As the evaluation is costly for the memory and computation during the search, we investigate further details of these requirements for incremental update:

- In states S_5 and S_6, if the pointer of n_X points to the only parent node whose support increases it requires investigating all parent nodes of n_X to find the lowest support value. Otherwise no pointer update is required (Condition C3).
- In states S_{-5} and S_{-6}, if the pointer of n_X does not point to the only node whose support decreases, it requires comparing the support of that specific parent node with the current parent node pointed by n_X, to find the lowest support for the parent nodes. Otherwise no pointer update is required (Condition C4).
- In state S_8, the pointer of n_X may point to the single parent node whose support is increasing or the single parent node whose support is decreasing or one of the existing parent nodes whose supports have no change. In the first case, it requires evaluating all parent nodes of n_X to find the lowest support value. In the second case, no evaluation is necessary (Condition C4). Finally, in the third case, it requires comparing the support of currently pointed parent node with the support of that specific parent node whose support is decreasing in order to find out if the change of pointer is necessary.

- *No-Update States*: In these states, support of X in the PAET node n_X does not change and no update is required for the the pointer of n_X as well. All items in X are associated to (0) and/or (N) transition trigger(s). Representative states are S_0, S_7 and S_9.
- *Support-Pointer Update States*: In these states, the support of X in the PAET node n_X changes. Also, its pointer, pointing to a parent node, may also change after comparing the support from all its parent nodes.
  - In state S_3, there exists only one parent node of n_X whose support does not change over the update while the supports of other parent nodes increase. Thus, if the pointer of n_X points to a parent node whose support increases, it requires evaluating the support of that specific parent node whose support does not change in order to find the lowest support for the parent nodes. Otherwise no pointer update is required.
  - In state S_{-3}, there exists only one parent node of node n_X whose support does not change over the update while the support of others decreases. Thus, if the pointer of n_X points to that parent node, it requires investigating all parent nodes of n_X to find the lowest support value. Otherwise no pointer update is required.

  Representative states are S_3 and S_{-3}.

The changes of support and pointers inside PAET nodes can be formally treated as side-effect for the hybrid automaton described in Figure 6.4. We add an additional implicit transition trigger ε to

IP in order to describe an implicit transition from a state to end state. Thus, IP can be denoted as $\{+, -, 0, N, \varepsilon\}$ where every transition is a tuple $\langle s_i, \gamma, \alpha, e, s_j \rangle$ corresponding to a move from state s_i to state s_j based on an transition trigger $e \in IP$. If the evaluation of its underlying predicate $\gamma = \alpha$ is true, s_j represents a state where side effect takes place, if required as per program instruction. Conversely, if $\gamma = \neg\alpha$ is true, s_j denotes a state with no side effect. Therefore, in practice, we may depict s_j as one state where the evaluation of the predicate triggers the side effects inside the state. Thus we find 15 states excluding the initial and the final states as depicted in Figure 6.4. The following example presents a PAET traversal using this automaton.

Let us consider an update of the sliding window from τ_1 to τ_2 in Figure 1.3. Then, we may construct an Update Set Pairs (USP) as: $\{(\text{a}, N), (\text{b}, N), (\text{c}, +), (\text{d}, -), (\text{e}, 0)\}$. Then, in a depth first search, we start searching first branch over PAET as depicted in Figure 6.3 as {a} ⇒ {a,b} ⇒ {a,b,c} ⇒ {a,b,c,e}. The first sequence of inputs is considered as $\langle([\gamma = \alpha], N), ([\gamma = \alpha], N), ([\gamma = \alpha], +), ([\gamma = \alpha], 0), ([\gamma = \neg\alpha], \varepsilon)\rangle$, which we abbreviate as $\langle[\alpha]N[\alpha]N[\alpha] + [\alpha]0[\neg\alpha]\varepsilon\rangle$. We may notice that each transition is guarded by a predicate and takes place for an input in IP. Each input sequence finishes at a leaf node as marked by $[\neg\alpha]\varepsilon$ at the end of every input sequence. However, in this example, the corresponding state transitions finish in three steps: (step 1) from initial state to S_7 accepting $([\gamma = \alpha], N)$, (step 2) from S_7 to S_9 accepting $([\gamma = \alpha], N)$ and (step 3) from S_9 to end state since S_9 as enforced by an implicit transition to the end state irrespective of the rest of the input sequence. In all visited states, required side effects take place. The search algorithm learns from the transitions of the first input sequence and generates the second sequence of inputs as: $\langle[\neg\alpha]N[\alpha] + [\alpha]0[\neg\alpha]\varepsilon\rangle$ which corresponds to searching {a} ⇒ {a,c} ⇒ {a,c,e}. Four transitions occur here but no side effect takes place at PAET node n_a since the input N is preceded by a predicate $[\gamma = \neg\alpha]$. The depth-first search continues until node n_e is reached.

6.4.6 $[n-1]$ Association Rule Tracking

The aforementioned hybrid automaton simplifies the actions needed for rule generation during the search process. Figure 6.5 depicts the connections between the states and the update settings.

In this bipartite graph, the set of states, at the up, shows the recipient states. The set of update settings, at the bottom, presents different requirements for interesting rule evaluations as shown in Table 6.2. Each edge presents a collection of states from which a recipient state (the current state of the automaton) can be reached where the corresponding update requirements are needed to be evaluated. For example, state S_{-2} can be reached only from states S_{-1} and S_{-2} which requires evaluation for update setting $U11$ at S_{-2}. Update setting "NA" denotes that no action is required. The exact choice of update settings for few transitions depends on the received inputs

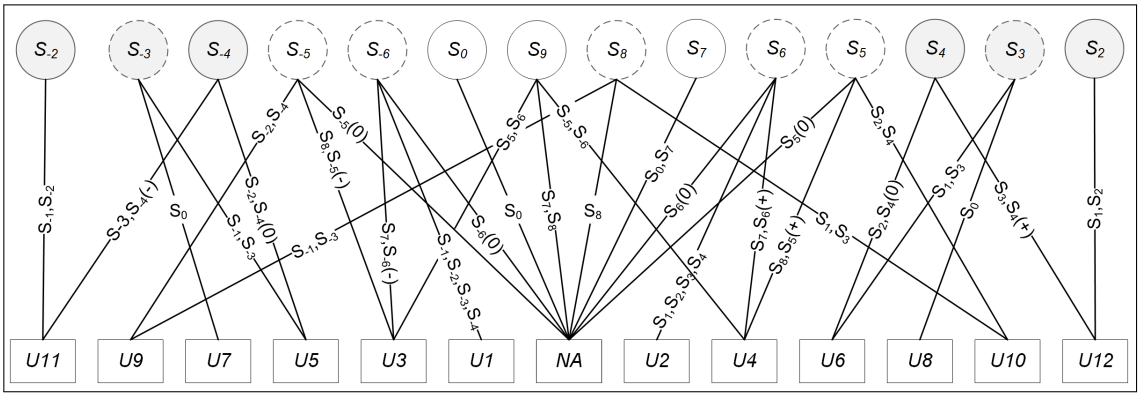


Figure 6.5: Selection of update settings for state transitions

as shown in Figure 6.5. For example, state transition from state S_{-4} to same state S_{-4} with input $[\alpha]0$ or $[\alpha]-$ leads to update setting $U5$ or $U11$ respectively.

Now, in $U11$, the supports of antecedent X, consequent Y and their joint appearance XY all are reduced by 1, as identified from Table 6.2. Further analysis of $U11$ reveals that, if first two conditions, $\left(\varsigma \geq c_{min} \text{ and } sup_j(X) \geq \frac{1-c_{min}}{\varsigma - c_{min}}\right)$, are true then new association rules can be created or an existing rule remains valid between two itemsets corresponding to the originating state (S_{-1} or S_{-2}) and the recipient state S_{-2}. However, in order to find an interesting association rule with $lift > 1$ another condition $\left(sup_j(Y) < \tau\varsigma + 1 - \frac{\tau(1-\varsigma)}{sup_j(X)-1}\right)$ has to be satisfied also. In case of failure to satisfy the first two conditions, no association rule exists between those two specific itemsets. If only the last condition does not hold, an association rule may exists but it is no longer *interesting*. Table 6.3 elaborates the exact need of evaluation actions for association rule generation, if necessary. The table is divided in two parts, namely (i) Existing Rules: when a $[n-1]$ association rule (interesting or uninteresting) exists between corresponding itemsets, (ii) Non-Existing Rules: when no $[n-1]$ association rule exists between corresponding itemsets. The column "Current State" denotes the present state of the depth-first search process. The column "Consequent Support Update" denotes update status, such as increase (+), decrease (-), no change (0) or no impact (N) for the consequent. It should be noted that, for all $[n-1]$ association rules, consequent is an itemset consisting of single item. Subsequently, two columns under "Evaluation Requirements" discusses required actions conditional to previous status of an existing relation between two itemsets. All actions can be only taken for guard $[\gamma = \alpha]$. The following example elaborates the necessary actions.

We take the same update of the sliding window from τ_1 to τ_2 in Figure 1.3. With update set

Table 6.3: Requirements for incremental evaluation of $confidence$ and $lift$

Line No.	Current State	Consequent Support Update	Evaluation Requirements $conf_j(\dots)$	$lift_j(\dots)$
	Existing Rules			
1.	S_2	Any	No	Yes
2.	S_{-2}	Any	Yes	Yes
3.	S_3	Any	No	if $(lift_{j-1}(\dots) \leq 1)$
4.	S_{-3}	Any	Yes	if $(lift_{j-1}(\dots) > 1)$
5.	S_4	(0)	No	if $(lift_{j-1}(\dots) \leq 1)$
6.	S_{-4}	(-)	Yes	Yes
7.	S_{-4}	(0)	Yes	if $(lift_{j-1}(\dots) > 1)$
8.	S_4, S_8, S_{-5}	(+)	No	Yes
9.	S_5, S_6, S_9	(+)	No	if $(lift_{j-1}(\dots) > 1)$
10.	S_5, S_8	(-)	Yes	Yes
11.	S_6	(N)	Yes	if $(lift_{j-1}(\dots) > 1)$
12.	S_{-6}	(N)	No	if $(lift_{j-1}(\dots) \leq 1)$
13.	S_9, S_{-5}	(-)	No	if $(lift_{j-1}(\dots) \leq 1)$
	Non-Existing Rules			
14.	S_{-5}, S_8	(+)	Yes	Yes
15.	S_{-6}	(N)	Yes	Yes
16.	S_2, S_3, S_4	Any	Yes	Yes

pairs $\{(a, N), (b, N), (c, +), (d, -), (e, 0)\}$, let us consider traversing the meet-semilattice in Figure 6.1(a) for {b} ⇒ {b,c} ⇒ {b,c,e}. So, we construct the input to the hybrid automaton as $\langle [\alpha]N[\alpha] + [\alpha]0[\neg\alpha]\varepsilon \rangle$. Now, when the search procedure reaches node n_{bc} in PAET, the corresponding transition in the automaton is state S_7 to state S_6. Now, S_6 here can be reached with consequent support update (+) from state S_7 (denoting relation between {b} to {b,c}) as well as consequent support update (+) from S_1 (denoting relation between {c} to {b,c}). From the meet-semilattice in Figure 6.1(a), it is clear that {b} $\nrightarrow$ {c} is not an association rule while {c} $\xrightarrow{l}$ {b} is a valid interesting association rule for the minimum support and minimum confidence of 3 and 0.7 respectively. No reevaluation of confidence or lift is needed for {b} $\nrightarrow$ {c} as understood from section of Non-Existing rules in Table 6.3. On the other hand, transition from state S_1 to S_6 invokes update setting U2. Reevaluation of both the confidence and the lift is needed for {c} $\xrightarrow{l}$ {b} as found at line no. 11 in Table 6.3. As found in Figure 6.1(a), at sliding window τ_2, {c} → {b} will be a valid association rule but its lift will be less than 1 which means it will no longer remain as an interesting rule. Similarly, when the search procedure reaches node n_{bce} in PAET, the corresponding present state of in the automaton is still S_6. This means revaluation of both the confidence and the $lift$ is needed only for {c,e} $\xrightarrow{l}$ {b}. As found in Figure 6.1(a), at sliding window τ_2, {c,e} → {b} will be only an association rule but its lift will be less than 1.

Algorithm 8 PAET update algorithm for incremental mining of $[n-1]$ association rules

Require: $\mathcal{PAET}_{t-1}$, $Stack\langle Map(itemset, parent, s_i)\rangle$

Constant: $IP : \{+, -, 0, N\}$, $S : \{s_{-6}, s_{-5}, \ldots, s_0, \ldots, s_9\}$, MIN_SUP, MIN_CONF
Known: ξ_-, ξ_+; {//Outgoing and incoming transactions respectively}
Input: $USP\langle(item, ip)\rangle \leftarrow$genUSP$(\xi_-, \xi_+, IP)$ {//$item \in \mathcal{A}$}
Initialize: Push each $(item, ip)$ of USP at $Stack$ in reverse alphabetical order along with parent $\varnothing$ and state s_{item}
Function updatePAET(USP, $Stack$) { {//Recursive update of PAET tree}
Pop top $(itemset, parent, s_i)$ from $Stack$
Search corresponding node $n_{itemset}$ in $\mathcal{PAET}_{t-1}$
$bool \leftarrow$ exists$(n_{itemset})$
if $bool$ **and** $s_i \in \{s_1, s_2, s_3, s_4\}$ **then**
    Increase support of $itemset$ at $n_{itemset}$ by 1
else if $bool$ **and** $s_i \in \{s_{-1}, s_{-2}, s_{-3}, s_{-4}\}$ **and** $sup_{t-1}(itemset) >= MIN\_SUP$ **then**
    Decrease support of $itemset$ at $n_{itemset}$ by 1
else if $bool$ **and** $s_i \in \{s_{-1}, s_{-2}, s_{-3}, s_{-4}\}$ **and** $sup_{t-1}(itemset) == MIN\_SUP - 1$ **then**
    Remove node $n_{itemset}$ and all nodes in $\mathcal{PAET}_{t-1}$ representing superset of $itemset$; $bool \leftarrow false$
else if $\neg bool$ **and** $s_i \in \{s_1, s_2, s_3, s_4\}$ **and** $sup_t(itemset) == MIN\_SUP - 1$ **then**
    Add new node $n_{itemset}$ in $\mathcal{PAET}_{t-1}$; $bool \leftarrow true$
end if
Update pointer for $n_{itemset}$ using *Hybrid Automaton* as described in Section 6.4.4
if confidence(getMCR($n_{itemset}$)) $\geq MIN\_CONF$ **then**
    Update $[n-1]$ association rules incrementally using Table 6.3
end if
if $bool$ **then**
    $children \leftarrow$ getChilds$((itemset, ip), USP)$
    Push each $child$ in $children$ at $Stack$ in reverse alphabetical order with parent $itemset$ and corresponding state s_{child}
end if
if $\neg$ empty($Stack$) **then**
    updatePAET(USP, $Stack$)
end if
}

6.4.7 Algorithm Design

In what follows, we present two algorithms to capture $[n-n]$ association rules. The first algorithm captures the incremental generation of all $[n-1]$ association rules while the second one focuses a modified apriori technique to generate $[n-n]$ association rules. Additional filters and acceleration techniques are used to quickly capture interesting rules.

In every update of the sliding window, a new transaction (ξ_+) arrives and an old transaction

(ξ_-) gets removed once the first sliding window is fully loaded. This forms a sequence of Update Set Pair (denoted as USP) using genUSP function at Step 3. The genUSP function maps every frequent item in alphabet and its IP type based on incoming and outgoing transactions. Thus, pair $(item, ip)$ in USP denotes whether a particular frequent item is increasing, decreasing, having no change or under no effect. Next, at Step 4, each item of the USP is placed in reverse alphabetical order into a *Stack* for further evaluation. The stack also keeps the parent for each item and its corresponding state in the hybrid automaton. Each state has a predefined set of update instructions for (i) node support, (ii) pointer handling, (iii) association rule evaluation, (iv) investigation of non-existing association rules and (v) further tree traversal details. Function updatePAET (Step 6-Step 27) updates the PAET through a tail-recursion to perform a selective depth first search using the stack. Inside updatePAET, traversal begins by *popping* the topmost pair out of the stack. Algorithm 8 incrementally updates all feasible $[n-1]$ association rules. At Step 10 and Step 12, it changes the support of itemsets. At Step 14 and Step 16, it adds or deletes PAET nodes, as required. Step 18 presents the update of pointers for the existing nodes while Step 20 elaborates the reevaluation of all $[n-1]$ association rules. The evaluation of association rules is only performed when the confidence of the MCR for its corresponding itemset passes the threshold of maximum confidence (Step 19). Finally, if further traversal is required for an itemset, its children are generated using function getchilds. The children nodes are selectively added based on the need for traversal using USP. Their states are also identified using the hybrid automaton. Similar to Step 4, new children nodes are reinserted into the stack in reverse alphabetical order to ensure the depth-first search. This algorithm also relates between the search procedure and automaton traversal. As the depth-first search progresses, the new input also performs the state transition in the automaton until it reaches the end state as described in Section 6.4.4. We elaborate the relation between tree traversal and state transition later in Section 6.4.8.

Algorithm 9 represents a modified apriori rule generation procedure with two main adjustments. First, it generates lifted association rules from the input of $[n-1]$ association rules instead of frequent itemsets. The apriori association rule generation is commonly performed using frequent itemsets [7, 118]. Second, it applies the aforementioned properties (*see* Section 6.3.1) to search these rules. The algorithm also applies a tail-recursion technique to subsequently generate all feasible $[n-n]$ rules from $[n-1]$ association rules. The algorithm first searches all $[n-2]$ association rules from these input $[n-1]$ association rules. Next, it finds $[n-3]$ association rules from from the $[n-2]$ rules and so on. Additional filtering is used to store only interesting association rules.

More precisely, the procedure begins with all $[n-1]$ association rules that are initially kept in an array namely *Rulelist*. Each entry of the *Rulelist* is an alphabetically ordered map of rules.

Algorithm 9 $[n-n]$ association rule mining from $[n-1]$ association rules

Require: N21Map($\langle id, rule \rangle$) {//All [n-1] association rules}

1: Constant: $WINDOW\_SZ$, MIN_CONF, MIN_LIFT;
2: Initially: $RuleList\langle Map_{ord}(\langle id, rule \rangle)\rangle \leftarrow$ sort($N21Map$), $N2NMap(\langle id, rule \rangle) \leftarrow \varnothing$;
3: **Function** N2NRuleGen($RuleList$) { {//Modified Apriori-based rule generation}
4: $newRuleList\langle Map(\langle id, rule \rangle)\rangle \leftarrow \varnothing$
5: **if** sizeof($RuleList$) >1 **then**
6: **for** level=1 **to** sizeof($RuleList$) **do**
7: $entry_{map}(\langle id, rule \rangle) \leftarrow$ elementof($RuleList, level$)
8: **if** ¬ empty($entry_{map}(\langle id, rule \rangle)$) **then**
9: $visitList\langle id \rangle \leftarrow \{\}$
10: **for all** $antecedent \in$ parentsof(antecedentof($rule$)) **do**
11: $proposedRule \leftarrow \langle antecedent,$ consequentof($rule$)$\rangle$
12: **if** ¬ contains($visitList, proposedRule$) **then**
13: Add getId($proposedRule$) to $visitList$
14: $bool \leftarrow$verify($proposedRule$) {//Use Properties 4a., 5a., 6b. and 1 to check rule}
15: **if** $bool$ **then**
16: Add to $N2NMap$ using add($proposedRule, bool$) {//new [n-n] assoc. rule found}
17: $lvl \leftarrow$ getLevel($proposedRule$)
18: Add $proposedRule$ at level lvl of $newRuleList\langle Map_{ord}(\langle id, rule \rangle)\rangle$
19: **end if**
20: **end if**
21: **end for**
22: **end if**
23: **end for**
24: **end if**
25: sort ($newRuleList$)
26: N2NRuleGen($newRuleList$) {//Perform a tail-recursion}
27: }

A map contains only those rules where the level of the antecedents of the rule in PAET matches with the position of the entry in the list. For example, an association rule {a,b} → {c} will be kept within the second map element of the *RuleList* since its antecedent belongs to Level 2. Also, it is the first element of this alphabetically ordered map given the alphabetical ordering of the rules.

In the beginning of any iteration i, a temporary new list is initialized at Step 4. The list is similar to *RuleList* in structure. Next, for each $[n-i]$ rule in *RuleList*, new candidate $[(n-1)-(i+1)]$ rules are produced at Step 11 by taking an item from the antecedent and adding it to the consequent. For example, from [2−1] association rule {a,b} → {c}, we may get two candidate $[1-2]$ rules: {a} → {b,c} and {b} → {a,c} in the next iteration. The proposed candidate rules are then verified using verify function (Step 14). This function evaluates the required properties

including the confidence and/or lift for every proposed rule, as necessary. Once the proposed rule succeeds the verification, it is considered as a new [n-n] rule. New rules are stored in $N2NMAP$ and kept in *newRuleList*. New rules are subjected to the next level of evaluation after sorting them in a predefined order (Step 25). Finally, Step 26 recursively calls the rule generation function to explore newer $[n-n]$ association rules.

In this context, we are also using *boundary*, an additional data structure, to track useful information from PAET for $[n-n]$ rule generation. A boundary is a bipartite graph among two disjoint sets of itemsets. It reduces the scope of rule search based on properties relevant to our selected association rules. Thus, it makes the search quicker but requires extra memory and periodic update in every new sliding window. Therefore, its effectiveness is an important consideration. We introduce the following two boundaries:

- *Omnipresence* (B_1): B_1 offers ignoring itemsets present in all transactions during rule search. Property 2 assures that no interesting rule can be constructed considering them antecedent or consequent.
- *Consequence* (B_2): B_2 determines if the consequent itemset (Y) has any impact over a selected association rule. Property 7 assures that if $sup_j(Y)$ is less than $c_{min} \times \tau$ then selected association rule is determined solely based on minimum support and minimum confidence requirements.

It is important to mention here that the Algorithm 9 is not incremental. It is called at every sliding window update after incrementally generating the $[n-1]$ association rules. The incremental generation of association rules is generally efficient than other existing techniques and it is completely possible to evaluate all $[n-n]$ rules incrementally by adding new rules and removing non-existing association rules. However, it requires a large amount of memory and computing resources to track a small number of rules. Therefore, we consider a design decision to generate the large number of $[n-1]$ association rules incrementally and faster followed by generating a small number of the $[n-n]$ association rules every time from the scratch after the sliding window update.

6.4.8 Example

In the following, we analyze the transactions of the first sliding window from Figure 1.3 along with incremental (interesting) association rule generation for next four sliding window update.

Table 6.4 explains the PAET traversal over Figure 6.3 during the update of the sliding window from τ_1 to τ_2 (as depicted in Figure 1.3). Parameters τ, s_{min} and c_{min} are kept at 10, 3 and 0.7 respectively. We consider that the input stack already holds all single items in reverse

Table 6.4: $[n-1]$ association rule generation for USP {(a,N),(b,N),(c,+),(d,-),(e,0)}

S_i	Pop (*Stack*)	Prev. Ptr.	Curr. Ptr.	Rule Evaluation	Push (*Stack*)
1	({a}, ∅, s_7)	∅	∅	-	({a,e},{a}, s_7), ({a,c},{a}, s_6), ({a,b},{a}, s_9)
2	({a,b},{a}, s_9)	n_a	n_a	lift({a,b}$\xrightarrow{l}${c})	-
3	({a,c},{a}, s_6)	n_a	n_a	lift({a}$\xrightarrow{l}${c})	({a,c,e},{a,c}, s_6)
4	({a,c,e},{a,c}, s_6)	n_{ac}	n_{ac}	lift({a,e}$\xrightarrow{l}${c})	-
5	({a,e},{a}, s_7)	n_a	n_a	-	-
6	({b}, ∅, s_7)	∅	∅	-	({b,e},{b}, s_7), ({b,d},{b}, s_{-6}),({b,c},{b}, s_6)
7	({b,c},{b}, s_6)	n_c	n_c*	lift({b} $\xrightarrow{l}$ {c})	({b,c,e},{b,c}, s_6)
8	({b,c,e},{b,c}, s_6)	n_{bc}	n_{bc}	lift({b,e}$\xrightarrow{l}${c}), lift({c,e} $\xrightarrow{l}$ {b})	-
9	({b,d},{b}, s_{-6})	n_d	n_d	-	({b,d,e},{b,d}, s_{-6})
10	({b,d,e},{b,d}, s_{-6})	n_{bd}	n_{bd}	-	-
11	({b,e},{b}, s_7)	n_b	n_b	-	-
12	({c}, ∅, s_1)	∅	∅	-	({c,e},{c}, s_3)
13	({c,e},{c}, s_3)	n_c	n_c*	- {e} omnipresent	-
14	({d}, ∅, s_{-1})	∅	∅	-	({d,e},{d}, s_{-3})
15	({d,e},{d}, s_{-3})	n_d	n_d	- {e} omnipresent	-
16	({e}, ∅, s_0)	∅	∅	-	-

alphabetical order. In every step (marked by column S_i), the top most entry is taken out of the stack and explored further for various update instructions. The columns *Prev. Ptr.* and *Curr. Ptr.* present the change of pointers for corresponding PAET nodes. The * mark denotes that an evaluation is necessary before assigning the pointers as discussed in Section 6.4.4. The column *Rule Update Tasks* indicates whether evaluations are necessary to create, delete or update new rules. However, at Steps 13 and 15, we may skip the rule evaluation task using $B1$ boundary since *item* "e" is omnipresent. Similarly, all the lift evaluations except lift({c,e} $\xrightarrow{l}$ {b}) can be avoided using boundary $B2$ as the support of itemset {c} is less than $\tau \times c_{min} = 7$. Column *Push (Stack)* presents new insertion of itemsets in the stack for further traversal. In this particular traversal, no node is added or removed at the tree.

Table 6.5 shows the total number of PAET nodes along with $[n-n]$ association rules and $[n-n]$ lifted interesting rules for different values of support and confidence over the transactions presented in Figure 1.3. Figure 6.6 compares our proposed data structure against other similar data structures from existing research effort. We construct Closed Enumeration Tree (CET) from Chi et al. [48] and Frequent Pattern Tree (FP-Tree) from Grahne and Zhu [96] for the first sliding window of the transactions. FP-Tree can be constructed with only 9 nodes whereas PAET and CET have 19 and 20 nodes respectively. While FP-Tree hosts minimum number of nodes, rules cannot be incrementally searched in such data structure as the whole tree may change over sliding

Table 6.5: PAET nodes and association rules for various support and confidence

Supp.	Conf.	T1			T2			T3			T4		
		Nodes	Rules	Lifted	Nodes	Rules	Lifted	Nodes	Rules	Lifted	Nodes	Rules	Lifted
	60%	19	37	21	19	33	12	19	33	19	15	33	19
30%	70%	19	27	12	19	26	9	19	23	13	15	23	13
	80%	13	18	9	19	14	6	19	14	6	15	14	6
	60%	17	19	12	15	16	3	15	16	7	15	16	7
40%	70%	17	12	6	15	12	3	15	15	7	15	15	7
	80%	17	12	6	15	9	3	15	9	3	15	9	3
	60%	11	11	6	11	11	0	11	11	4	11	11	4
50%	70%	11	7	3	11	7	0	11	10	4	11	10	4
	80%	11	7	3	11	4	0	11	4	0	11	4	0

window update depending on the support of different itemsets. PAET contains comparatively less number of nodes than CET since CET starts accumulating infrequent gateway nodes over time.

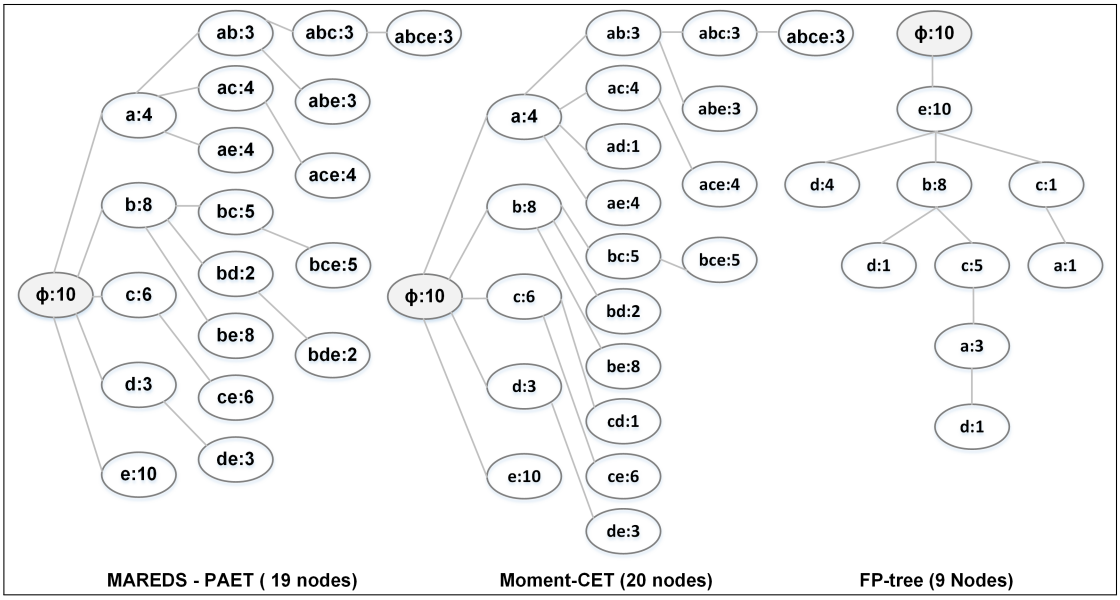


Figure 6.6: Comparison of three tree structures: PAET, CET (Moment) and FP-Tree

6.5 Collaborative Association Rule Mining

In the setting of collaborative association rule mining, there exists P decision makers, each of whom ($p \in P$) has own concern to produce alerts for a subset of events/items ($\mathcal{A}_p$) from the whole alphabet of items ($\mathcal{A}$). In our context, these alerts originate from the lifted association rules. Therefore, collaborative mining needs every participant to analyze a set of association

rules, where the antecedent and/or consequent of each rule contains item(s) from $\mathcal{A}_p$. Evaluation of an association rule $X \rightarrow Y$ involves determining support of X and the support of the joint occurrences of X and Y. In the case of lifted association rules, support of Y is also analyzed. Therefore, every participant needs to mine additional items beyond $\mathcal{A}_p$ to incrementally find lifted association rules of own concern. Furthermore, after every update of the sliding window, supports of various itemsets change which may subsequently alter the (interesting) rules as well. At time t, let $\mathcal{A'}_p^t$ ($\mathcal{A}_p \subseteq \mathcal{A'}_p^t \subseteq \mathcal{A}$) represent the subset of alphabet that should be monitored to incrementally mine all interesting association rules. Now, without any knowledge over the support of all items and itemsets at the current sliding window, it is difficult for any participating decision maker to correctly select $\mathcal{A'}_p^t$ in order to monitor all association rules of own interest. So, in this section, we discuss a collaboration technique to mine PAET locally at distributed servers of the participating decision makers in order to find all association rules. In this regard, we require a small number of *Helper* entities (as depicted in Figure 1.3) to incrementally track necessary items ($\mathcal{A'}_p^t$) for each participant at every sliding window update.

6.5.1 Incremental Tracking of Maximal Frequent Itemsets

In what follows, we analyze an incremental tracking of Maximal Frequent Itemsets (MFI) in the data center at every sliding window update in order to quickly compute $\mathcal{A'}_p^t$ for each participating decision maker. We begin with the following property.

Property 8. *If an association rule $X \rightarrow Y$ is valid at a particular update of the sliding window then frequent itemsets X, Y and XY are all subsets of at least one MFI of itemset X among all feasible MFIs at that window.*

Proof. Let us assume that there exists no single MFI that is a superset of X, Y and XY together. Then, X and Y, both being frequent, should be subsets of two different MFIs (apriori property). It also ensures that there is no MFI that is a superset of XY. However, it indicates that XY is infrequent. Therefore, $X \rightarrow Y$ cannot be valid. □

Property 8 indicates participant p should mine a set of items composed of elements from those maximal frequent itemsets which contain atleast one item of $\mathcal{A}_p$ in order to track all interesting association rules of $\mathcal{A}_p$. Assuming that every item of $\mathcal{A}$ is monitored by at least one participant, this collaborative setup will then produce all the interesting association rules of corresponding centralized setting.

Similar to centralized setting, we propose mining MFIs against the support requirement which is one less than actual minimum support threshold. This allows, in distributed setting, to

insert every relevant node to PAETs that may potentially be part of an association rule in the next sliding window update. However, with every update of sliding window, MFIs are likely to change as well. This involves formation of new MFIs and deletion of existing MFIs.

In the data center, Helpers identify these changes and deduce for every participant a new set of required items to be monitored at every sliding window update. Often time, it also requires the supports of new itemsets by scanning the bit matrix. The new information is then communicated to every participant in order to meet their concerns. Similar to the centralized setting, new nodes are locally added at the PAET of participant's side when their supports are one less than the minimum support threshold. Thus, every participant starts tracking these nodes right before one sliding window update where these nodes may potentially form association rules.

6.5.2 Algorithm Design

A quick mining of a PAET at the participant's server can be performed in three steps. First, it needs for incremental tracking of MFIs. Second, it requires determining $\mathcal{A}'^{t}_{p}$ from these MFIs. Finally, Helpers, communicate support of newly relevant itemsets to every participant from the data center. The second and the last steps involve relatively simpler standard procedures. Therefore, here, we mainly focus on a fast incremental technique to identify MFIs using previously generated MFIs. Algorithm 10 presents an overview of the technique.

Algorithm 10 illustrates three core functions of the incremental search of MFIs using previous MFIs as stored in $oMFI_{t-1}$. The main function *MFIGenerator* uses the current update set pair and previous MFIs to search MFIs in two main activities. First, it uses a *switch* statement after identifying a change of support for a previous maximum frequent itemset. If the support of itemset does not change (unrelated or no change as per USP), it retains the itemset as MFI for the current set of MFIs ($oMFI_t$) (Step 6 and Step 7). If the support of the itemset increases it calls to explore its supersets to identify new MFIs (Step 8 and Step 9). If the support decreases, the function looks into the subsets of the itemset (Step 10). Apart from this incremental update process, function *MFIGenerator* also checks for supersets of each non-decreasing item of USP selectively in order to search whether new MFIs are formed due to branch insertion (Steps 13-16). Significant computation can be saved in this last part of evaluation using already identified MFIs in the *switch* statement.

Function *exploreSuperset* identifies new MFIs that are superset of a given MFI. It is a recursive procedure that takes the advantage of a non-decreasing array generated using USP. This array contains only those members of USP where the support of the item has not been decreased. Every time, the function prepares a new itemset by increasing the input MFI with a new member

Algorithm 10 Incremental Generation of Maximal Frequent Itemsets

Require: $oMFI_{t-1}\langle id, itemset\rangle$, $USP_t\langle(item, ip)\rangle$ {//Ordered sequence of MFIs and USP}

Initially: $SUP \leftarrow MIN\_SUP - 1$, $oMFI_t \leftarrow \varnothing$; $unrelated \leftarrow -2$;
Function MFIGenerator($oMFI_{t-1}$, USP_t) { {//USP_t stores items and their incremental change}
for all $mfi \in oMFI_{t-1}$ **do**
    $change \leftarrow getChange(USP_t, mfi)$
    switch ($change$)
    case $unrelated$: add mfi to $oMFI_t$; **break**
    case 0: add mfi to $oMFI_t$; **break**
    case 1: $bool \leftarrow$ exploreSuperset(mfi, getNonDecreasingItemsets(USP_t), -1,0, $false$)
        if $\neg bool$ **then** add mfi to $oMFI_t$ if no superset exists **end if** **break**
    default: exploreSubset(mfi, getUnchangedItemsets(USP_t)); **break**
    end switch
end for
for all $item \in$ getNonDecreasingItemsets(USP_t) **do**
    $bool \leftarrow$ exploreSuperset($\{item\}$, getNonDecreasingItemsets(USP_t), -1,0, $true$)
    if $\neg bool$ **and** supportof($\{item\}$) $\geq SUP$ **then** add mfi to $oMFI_t$ if no superset exists **end if**
end for
}
Function exploreSuperset(mfi, $nonDecreArr$, $tailPos$, $sibling$, $checkMFI$) {
if $checkMFI$ **then** existsMFI($head$,$oMFI_t$) **then** return true **end if**
if sizeof($head$)=sizeof($nonDecreArr$) **then** return false **end if**
$head \leftarrow$ getHeadUTail(mfi, $nonDecreArr$, $sibling$); $bool \leftarrow true$; $temp \leftarrow$ updateTailPos($tailPos$)
if supportof($head$) $\geq SUP$ **then**
    $bool \leftarrow$ exploreSuperset($head$, $nonDecreArr$, $tailPos + 1$, 0, $checkMFI$);
    if $bool$ **and** hasMoreSibling($tailPos$,$nonDecreArr$) **then**
        $bool \leftarrow$ exploreSuperset(mfi, $nonDecreArr$, $temp$, $sibling + 1$, $checkMFI$)
    end if
else if $\neg$ hasMoreSibling($tailPos$,$nonDecreArr$) **and** supportof(mfi) $\geq SUP$ **then**
    add mfi to $oMFI_t$ if no other superset of mfi exists
else
    $bool \leftarrow$ exploreSuperset(mfi, $nonDecreArr$, $temp$, $sibling + 1$, $checkMFI$)
end if
if $\neg bool$ **and** supportof(mfi) $\geq SUP$ **then** add $head$ to $oMFI_t$ **end if**
return true
}
Function exploreSubset(mfi, $nonChngArr$) {
if supportof(mfi) $\geq SUP$ **then** add mfi to $oMFI_t$; return **end if**
for all $subset \in$ generateAllParents(mfi) **do**
    if $\neg$ superSet($subset$,$nonChngArr$) **and** supportof($subset$) $\geq SUP$ **then** add $subset$ to $oMFI_t$
    else exploreSubset($subset$, $nonChngArr$)
    end if
end for
}

from non-decreasing array and tests its support. If the support is no less than the minimum support requirement (MIN_SUP -1, where MIN_SUP is the minimum support threshold) then it keeps adding new items and evaluating. If not, then it removes the lastly added item and selects another item from the non-decreasing array to create a new itemset from the input MFI. Newly found MFIs are added to $oMFI_t$ if and only if $oMFI_t$ does not contain the MFI or its superset.

Finally, function *exploreSubset* is a small function to identify new MFIs when the support of the current MFI does not meet minimum support requirements. First it generates all parent itemsets from the current MFI where the size of each parent itemset is one less than that of the current MFI. If the support of subset meets the minimum support requirement, it is considered as a new MFI, otherwise the subset is recursively explored for its newer subsets that meets the minimum support requirement.

Once the new MFIs ($oMFI_t$) are identified, simple calculation is performed to compute $\mathcal{A}'^t_p$. Let us consider an example. Let $p1$, $p2$ and $p3$ be three collaborative decision makers of the data center analyzing the data stream depicted in Figure 1.3. The fixed concerns of $p1$, $p2$ and $p3$ are {a,b}, {d} and {c,e}. Table 6.6 presents their requirements for mining itemsets for three consecutive sliding window updates.

Table 6.6: Changing requirements of mining MFIs

Sliding window	$p1$: concern {a,b}	$p2$: concern {d}	$p3$: concern {c,e}
$\tau_1 \rightarrow \tau_2$	{a,b,c,e}	{b,d,e}	{a,b,c,d,e}
$\tau_2 \rightarrow \tau_3$	{a,b,c,e}	{a,d,e}	{a,b,c,d,e}
$\tau_3 \rightarrow \tau_4$	{a,b,c,e}	{d}	{a,b,c,e}

6.6 Benchmark Results and Comparative Study

In order to evaluate the performance, we implement the proposed algorithms in a Java application and extensively test them over seven data streams. The implemented application module is called as Mining Association Rules over Event Data Stream (MAREDS). Each data stream is generated by simulating transactions from a known dataset. These datasets are carefully chosen in similarity to previous research efforts. All our experiments are performed using 3.40 GHz Intel Core i7-2600 PC with 8 GB main memory, running 64 bit Windows 7 operating system. The main aim of these experiments is to quantitatively assess the advantages and limitations of MAREDS with respect to centralized and collaborative monitoring of events. Therefore, we mainly perform the tests to find small to medium number of association rules which can be transformed into meaningful alerts during plan execution. Table 6.7 presents characteristics of seven datasets used in our experiments.

Table 6.7: Experimental Datasets characteristics

Dataset	Data Type	Number of items	Transactions			Window Size
			Count	Avg. length	Max. length	
BMS-WebView-1	Real	497	59602	2.51	267	2K, 50K
BMS-WebView-2	Real	3340	77512	4.62	161	2K, 50K
Kosarak	Real	41270	990002	8.10	2498	5K - 120K
Accidents	Real	468	340183	33.81	51	10K
T5I4D100K	Synthetic	500	100K	4.87	17	10K - 80K
T10I4D100K	Synthetic	500	100K	9.80	29	10K - 80K
T20I5D100K	Synthetic	500	100K	19.85	47	10K - 80K

The first four datasets, namely BMS-WebView-1, BMS-WebView-2, Kosarak and Accident are generated by capturing actual events from real environment. BMS-WebView-1 and BMS-WebView-2 are two datasets of click streams of 59,601 and 77,512 transactions respectively. These two real-world datasets were used for KDDCUP 2000[2]. Kosarak[3] is another large dataset of 990,000 anonymized transactions of click streams from a large online news portal. Accidents dataset is published by Geurts et al. [89] containing information of traffic accidents from 1991 to 2000 in the region of Flanders (Belgium) as obtained from the National Institute of Statistics, Belgium. This dataset is closely related to the delay monitoring for commodity delivery plan execution as it mines large number of different events/attributes of traffic accidents. The last three datasets namely T5I4D100K, T10I4D100K and T20I5D100K are synthetically generated by the IBM Quest Synthetic Data Generator[4]. As identified in Table 6.7, The symbols T, I and D in the three synthetic datasets denote the average number of items per transaction, the average size of itemsets in potential frequent sequences and the number of transactions in the dataset respectively.

6.6.1 Performance of Incremental Association Rule Mining

We compare the MAREDS application against two closely related existing approaches. Since, there is no other suitable technique to incrementally generate association rules directly over sliding window model, we consider two approaches (i) Moment: an existing incremental frequent itemsets generation technique [48] and (ii) FP-Growth: a non-incremental frequent itemsets generation technique [96], to find all required itemsets at every sliding window update. Then, both approaches use efficient apriori technique to generate rules from the frequent itemsets. It can be noticed that all these approaches, MAREDS, Moment and FP-Growth depend on tree data structures (PAET, CET, FP-Tree respectively).

[2] http://www.kdd.org/kdd-cup/view/kdd-cup-2000
[3] http://fimi.ua.ac.be/data/
[4] https://sourceforge.net/projects/ibmquestdatagen/

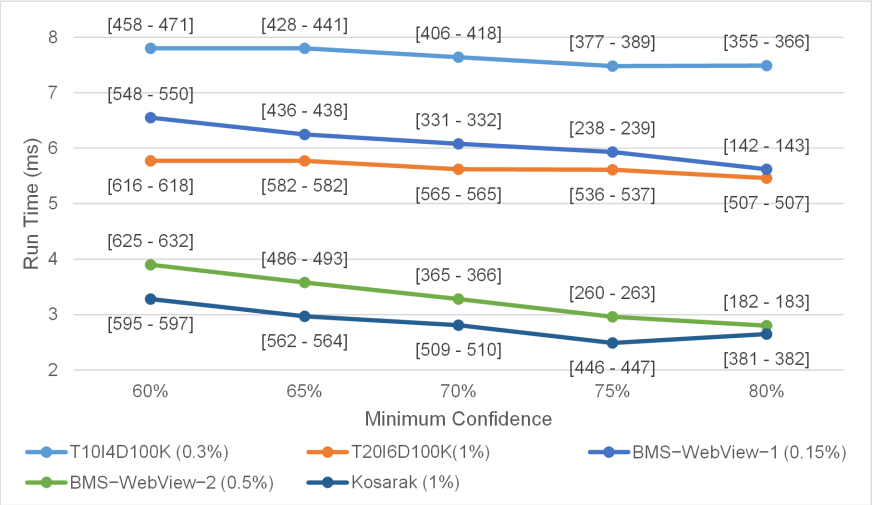


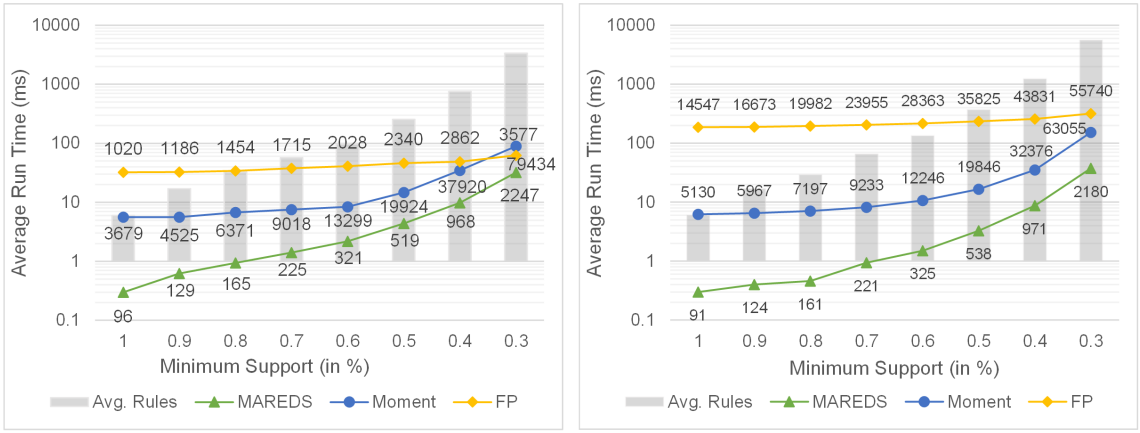
Figure 6.7: Association rules and performance evaluation for different datasets

Figure 6.7 evaluates the performance of MAREDS application in 100 consecutive updates for sliding window of size 50,000 (50K). Every dataset is tested over a range of confidence values for a fixed predefined support as marked in () within the legend. Each data point in this chart is also associated with the minimum and maximum number of rules as found during the window updates (presented in [] brackets). In all these experiments, MAREDS finds association rules in less than 10 milliseconds. It should be also noted that while decrease in confidence values increases the number of association rules, the performance of MARED remains stable for fixed support and window size.

(a) Sliding Window Size = 2,000

(b) Sliding Window Size = 50,000

Figure 6.8: Memory and execution time comparison for BMS-WebView-1 dataset



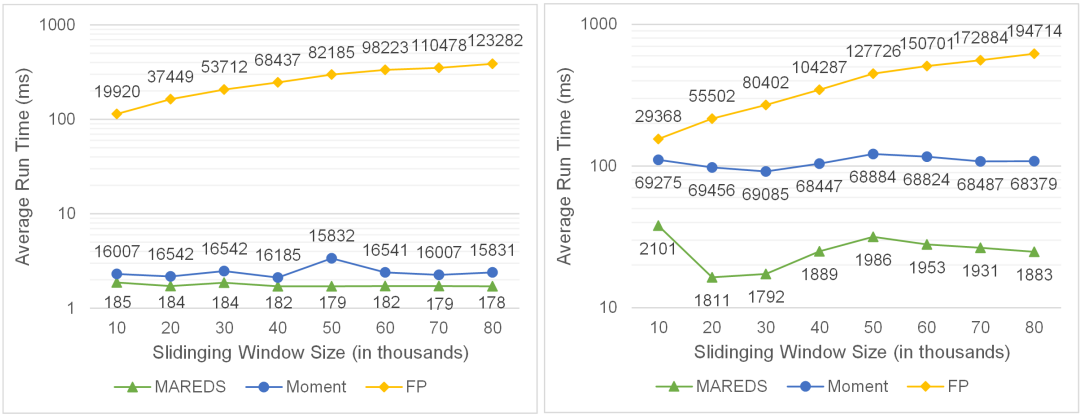
(a) Sliding Window Size = 2,000 (b) Sliding Window Size = 50,000

Figure 6.9: Memory and execution time comparison for BMS-WebView-2 dataset

Figure 6.8 and Figure 6.9 depict comparative study of handling memory and execution time for MAREDS against other two aforementioned approaches: Moment and FP-Growth. Experiments are performed over a range minimum support values by keeping minimum confidence fixed at 70%. Two sub-graphs for each figure present the performance for sliding window of size 2K and 50K in similar to previous research efforts [48]. The performance is evaluated over a logarithmic scale. The label on each data point represents the number of nodes in the corresponding trees. It is evident that, within this test range, MAREDS performs faster than other two approaches. It also stores less number of PAET nodes in compare with CET nodes in Moment and FP-Tree nodes in FP-Growth. Finally, the column graph in the background at each sub-figure projects information of the average number of association rules (over the same logarithmic scale) for each set of experiments. The average is calculated over the number of association rules as found from 100 consecutive sliding window update. We can see that the average number of association rules reaches more than 20,000 at the end of each sub-figure. Therefore, for the purpose of our plan execution monitoring, we do not intend to stretch the experiments for further lower support values.

Figure 6.10, Figure 6.11 and Figure 6.12 compare performance of MAREDS, Moment and FP-Growth for three synthetic datasets namely T5I4D100K, T10I4D100K and T20I5D100K. In these cases, we evaluate execution time and number of nodes in the core data structure of each approach for fixed percentage of minimum support and minimum confidence values over a range of sliding window sizes. In each figure, two fixed minimum support values are presented in percentage, one high and one low. The average run time is evaluated over a logarithmic scale. With the increase of sliding window size, the absolute value of minimum support linearly increases (although

the percentage value is fixed). The performance of MAREDS is very little affected whereas the execution time of FP-Growth increases continuously.



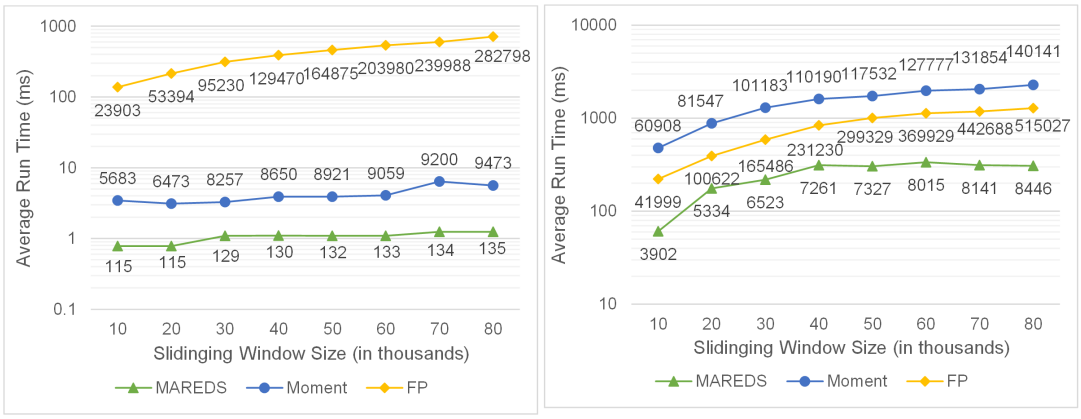
(a) Support 1.0% (b) Support 0.2%

Figure 6.10: Performance comparison for T5I4D100K dataset

This stems from the fact that FP-Growth reads all valid transactions in each sliding window update. So, with larger window size, FP-Growth takes more time to form FP-Tree and find rules. Moment and MAREDS algorithms are incremental, so they are relatively stable, since the changes of the corresponding trees are minimal after the first window. It is also observed that Moment algorithm does not perform well in lower support as a large number of infrequent CET nodes start impacting its memory management and performance negatively. In this range of experiments, among all three approaches, MAREDS stores least number of tree nodes as marked by the labels.

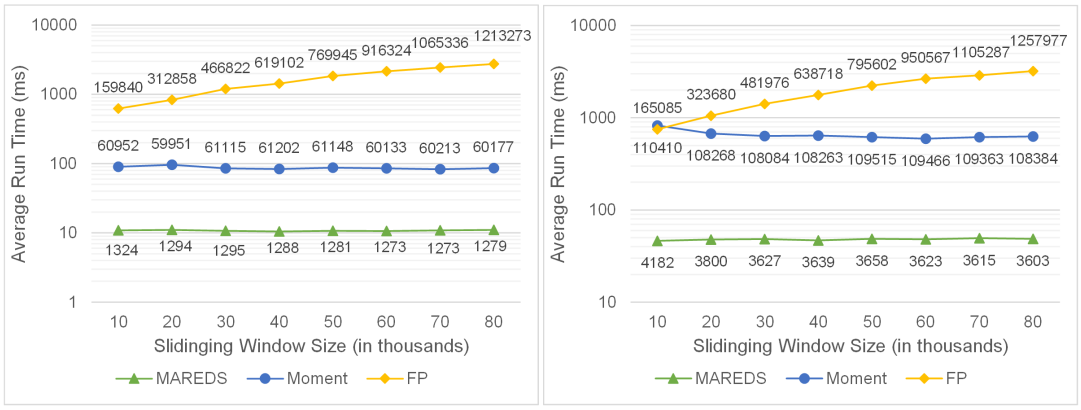
Figure 6.13 compares all three approaches over Kosarak dataset. In Kosarak, as mentioned in Table 6.7, the alphabet size is 41270 and the maximum transaction length is 2498 which are highest among all seven datasets. Such a dataset generates large number of infrequent itemsets which makes maintenance of CET difficult for Moment although infrequent itemsets cannot be the part of any association rule. Figure 6.13 shows that, in both cases, minimum support of 4% and 0.4%, Moment fails to produce results above sliding window size 10000. FP-Growth approach finds the association rules but takes longer time and mines far more tree nodes in compare to MAREDS.

Figure 6.14 depicts the performance of MAREDS over Accidents dataset. Characteristically, the average length of transaction over this dataset is 33.81 for the alphabet size of 468. Therefore, supports of itemsets are expected to be generally high in this dataset. Over a fixed sliding window size 10000 and minimum confidence value of 70%, as we decrease the minimum support from 90% to 65%, the number of association rules increases from 218 to 27020. Likewise, the number of lifted association rules also increases from 172 to 24144. The column graphs depict the association rules



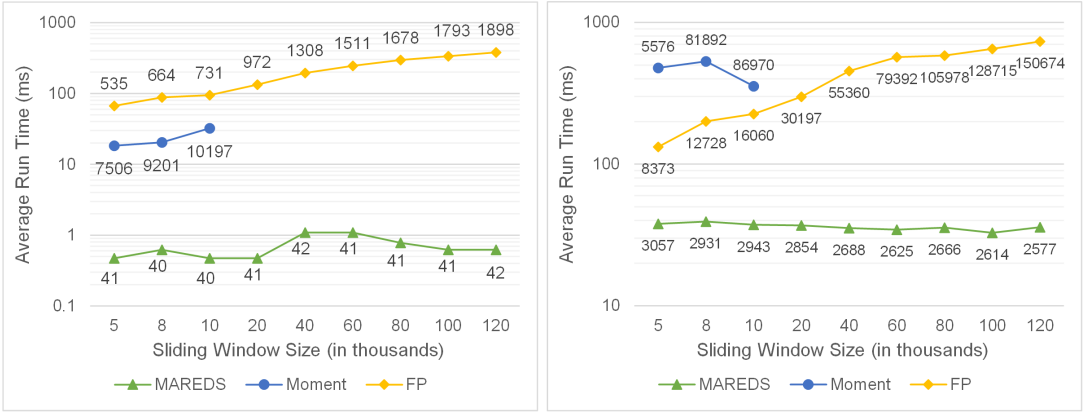
(a) Support 2.5% (b) Support 0.25%

Figure 6.11: Performance comparison for T10I4D100K dataset



(a) Support 1.6% (b) Support 1.1%

Figure 6.12: Performance comparison for T20I5D100K dataset



(a) Support 4.0% (b) Support 0.4%

Figure 6.13: Performance comparison over Kosarak dataset

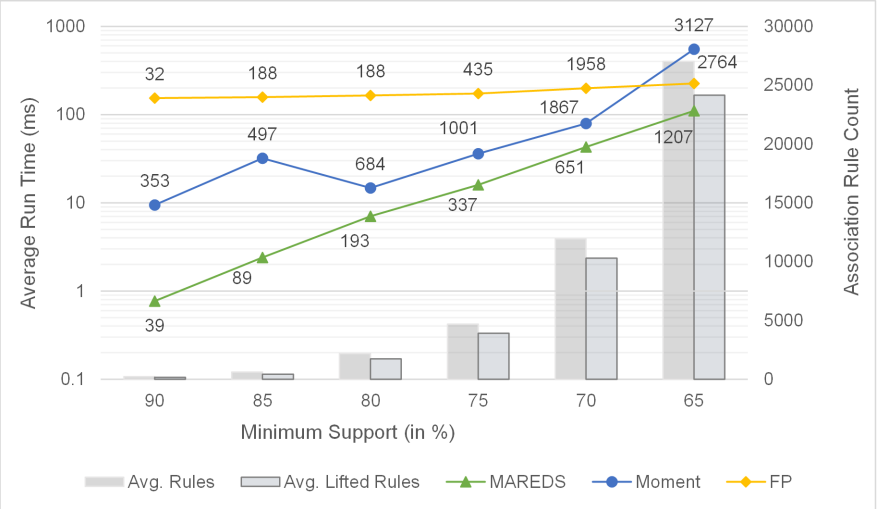


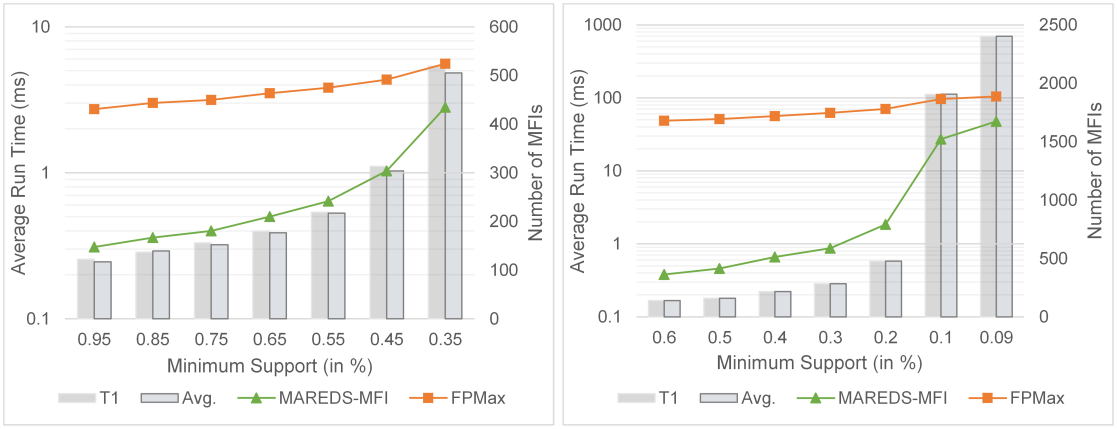
Figure 6.14: Finding lifted association rules from Accidents dataset

and lifted association rules on secondary axis. Even with such a large number of rules, MAREDS mines less number of tree nodes and computes rules faster compared to other two approaches.

6.6.2 Performance of Incremental Maximum Frequent Itemsets Mining

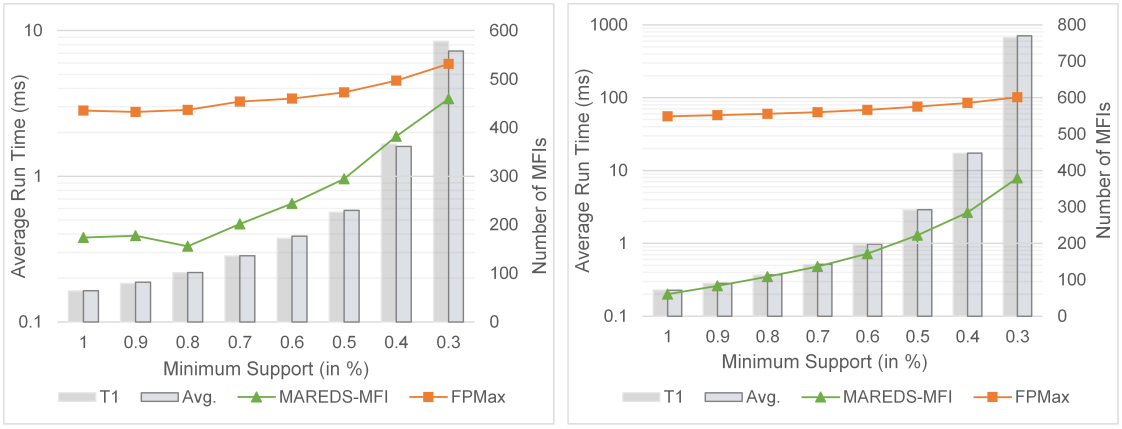
We extend the MAREDS application to incorporate the incremental generation of MFIs over data streams using various datasets in the same experimental environment. We call the implementation MAREDS-MFI in the following figures. The first two experiments have been performed on BMS-WebView-1 and BMS-WebView-2 datasets to evaluate the performance of the algorithm against the change of minimum support thresholds. For each figure, two charts have been presented for sliding window size 2000 and 50000. The last experiment has been performed on T5I4D100K dataset to evaluate the performance of the implementation against changing size of the sliding window. Two charts have been presented for fixed minimum support percentage of 1.0% and 0.2%.

Figure 6.15 depicts two charts on BMS-WebView-1 datasets elaborating performance of MAREDS-MFI implementation against different values of minimum support. The average run-time and number of MFIs are captured in each chart using primary and secondary Y-axes respectively. The average run-time of MAREDS-MFI remains almost linear with decreasing values of the minimum support thresholds. As the threshold is getting reduced, we notice that more and more MFIs are being captured by our proposed algorithm. The label T1 in the all charts represents the number of MFIs in the first sliding window while the label Avg. denotes the average number of MFIs in next 100 sliding window update. As the average run-time is presented in logarithmic scale,



(a) Sliding Window Size = 2,000 (b) Sliding Window Size = 50,000

Figure 6.15: Memory and execution time comparison for BMS-WebView-1 dataset



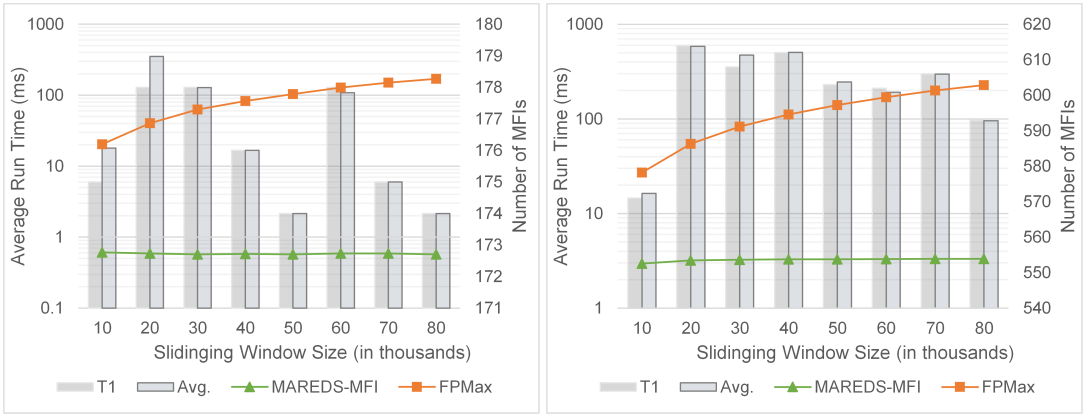
(a) Sliding Window Size = 2,000 (b) Sliding Window Size = 50,000

Figure 6.16: Memory and execution time comparison for BMS-WebView-2 dataset

our implementation clearly outperforms one of the known algorithms for MFI capturing, namely FPMax [95]. The faster searching of MFIs stems from the incremental handling of stream data.

A similar performance can be noticed in the two charts of Figure 6.16 where we compare the performance of MAREDS-MFI implementation with FPMax for different values of minimum support over BMS-WebView-2 datasets.

Figure 6.17 depicts performance of MAREDS-MFI implementation against increasing size of the sliding window. The average run-time of the implementation is presented in logarithmic scale at primary Y-axis while the number of MFIs are presented in linear scale at secondary y-axis. In both cases, number of MFIs varies little (range of 174-179 for Figure 6.17 (a) and range of 570-620 for Figure 6.17 (b)) in compare to the increase in sliding window size. The average run time of



(a) Support 1.0%

(b) Support 0.2%

Figure 6.17: Performance comparison for T5I4D100K dataset

MAREDS-MFI remains almost constant while execution time of FP-Max algorithm grows with the increasing sliding window size.

6.6.3 Performance Analysis

The MAREDS application is actually designed to monitor data streams and generate alerts. The incremental handling of a large number of transactions is beneficial in practice for low concept drift. With the update of sliding window, it demands for minimal changes of tree nodes and existing rules. Furthermore, we assume that the number of association rules (lifted or non-lifted) is reasonable to generate meaningful alerts. The aforementioned figures in Section 6.6.1 clearly present the suitability of MAREDS and the benefits of incremental association rule mining over other existing approaches. The loading time of the first window of MAREDS is time consuming in compare to FP-Growth. However, once the first window is loaded, the changes of PAET is often found minimal with the update of the sliding window. Thus, MAREDS updates the support of tree nodes faster and computes the changes in association rules efficiently.

In Section 6.6.2, we discuss our implementation's performance for the incremental maximum frequent itemsets mining approach. The charts clearly show the efficiency of our approach against other non-incremental MFI mining approach namely FPMax [95]. While comparing Figure 6.8 and Figure 6.15, it can be easily understood that incremental mining of MFI takes only a fraction of time with respect to incremental association rule generation. The same can be observed in other datasets as depicted in Figure 6.9 and Figure 6.16 along with Figure 6.10 and Figure 6.17. Therefore, we believe a significant improvement in performance can be achieved by distributing the incremental mining process using our proposed setup of collaborative mining of association rules.

6.6.4 Advantages and Limitations

Incremental mining of association rules produces up-to-date rules of users' interest and helps generating alerts during plan execution monitoring. The key advantage of our approach consists in the efficient processing of stream data over the sliding window model. This involves the use of outgoing and incoming transactions along with existing association rules from the last sliding window. This approach is faster on average compared to traditional non-incremental mining procedures as it requires updating only those association rules which are affected by the window update. Moreover, collaborative incremental mining offers additional benefits since participating decision makers can choose updating only the rules according to their interests. This leads to mining less number of items and rules which corresponds to faster update and less memory use.

On the other hand, incremental rule mining becomes slower over high concept drift particularly when the sliding window update impacts the frequency of several itemsets along with the confidence and lift of a large number of association rules. Another specific problem of incremental mining is the loading of the first sliding window. As we approach the PAET generation incrementally, it takes longer time to load the first window while periodically updating the association rules compared to non-incremental rule generation technique, such as combined FP-Growth and Apriori technique. Furthermore, PAET is designed on a prefix tree. The latter is a simple data structure which is easy to update but requires a large amount of memory. Therefore, a more sophisticated data structure is required for handling certain practical situations such as the need to obtain a trade-off between performance and memory.

6.7 Summary

In this chapter, we proposed a novel event monitoring procedure that extracts *interesting* relations among generic events from a data stream. We have demonstrated how such incremental monitoring can be performed in centralized and distributed settings. In this regard, the conducted experimental studies clearly indicate that the proposed algorithms can efficiently capture interesting relations as association rules from large stream of events. A generic and efficient monitoring technique, that can be deployed in distributed framework, can benefit a large number of monitoring applications from diverse application areas. In future, our research work can be extended by focusing on incrementally capturing patterns from data stream with the update of the sliding window and by developing forecasting mechanisms. Also we intend to investigate potential modifications on top of this study in order to address specific application domains.

Chapter 7

Conclusion

The transportation overhead cost is often considered as one of the largest spending for government, business and defense organizations across the world. With the advent of computer-assisted planning and tracking technologies, transportation related problems are expected to be handled in more and more complex environment. The potential existence of different information sources, that are currently available in widely dispersed geographical locations, offers possibilities for efficient commodity delivery planning, plan tracking and successful completion of tasks over large transport network. However, this requires developing a comprehensive framework of knowledge sharing and problem solving in line with ever increasing global reach and adoption of cyberspace. In order to mitigate the gap between current and future handling of transportation planning and monitoring, we have investigated collaborative handling of three core research problems concerning the vehicle route planning and monitoring. This thesis presents an innovative approach to solve these three problems in a specific distributed setting. Throughout this thesis, we have discussed corresponding models and solution algorithms in details.

More precisely, in Chapter 1, we have introduced three research problems and elaborated the collaboration setting for vehicle routing problem, monitor deployment problem and plan execution monitoring problem. Next, in Chapter 2, we have characterized these problems and discussed an overview of existing research and development efforts on these three aforementioned problems. In Chapter 3, we have started by proposing a heurisitc technique to near-optimally solve multi-depot vehicle routing problem in split-delivery setting. Although, this technique uses a centralized setting, the heuristic elaborates a new way of solving multi-depot VRP variants. It is used as an efficient method of route planning at individual participants in the collaborative setting. It also serves as a basis of comparison for the distributed approaches of solution generation. In Chapter

4, we have illustrated two distributed approaches to solve multi-depot VRP instances. The first one is a collaborative evolutionary learning approach while the second approach is cooperative negotiation based on game theory. These approaches address distributed solution generation for multi-depot VRP and present generic techniques for solving various commodity delivery problems. In Chapter 5, we have addressed a monitor deployment problem in a collaborative setting where the monitor deployment budget is divided among participants. We have applied a multi-round risk reduction technique to near-optimally find monitor locations for this budget constrained monitor deployment problem. Finally, Chapter 6 describes an incremental approach for plan execution monitoring based on data mining of interesting relations among generic events as association rules. Unlike traditional approaches, we search these rules incrementally from a data stream with every update of newer monitoring events using a sliding window model. We have also proposed a new approach for collaborative mining of the association rules where each participant can partially mine these rules according to its interest. Extensive experiments have been performed for each of the solution approaches discussed in this thesis. We have documented detailed benchmark results on known problem instances. These results have been contrasted against those obtained by other known techniques in order to reflect the suitability and efficiency of the proposed approaches.

A fully functional distributed platform for advanced transportation management system requires solving several research and technical issues. These problems are often complex and restricted by constraints ranging from lack of resources to prohibitive policies. In the scope of this thesis, we try to bridge a small portion of this gap in context of aforementioned three specific research problems. We have successfully proposed a number of collaboration approaches that can help using such a platform of transportation planning and monitoring. To this end, we have designed and implemented specific algorithms for distributed decision makers who can locally execute these algorithms while participating together to achieve a common global objectives. The main contributions of this thesis include these algorithms which provide practical solutions for the studied research problems. With increasing global reach and data sharing, we believe that these algorithms will promote collaborative use of individual capabilities while using common resources for transport planning and monitoring. In future, a distributed platform can be developed by leveraging the proposed algorithms and techniques for joint planning and monitoring activities in an integrated collaborative environment.

Bibliography

[1] Hernan Abeledo, Michael Bussieck, Leon Lasdon, Alex Meeraus, and Hua Ni. Global optimization and the gams branch-and cut facility. CORS/INFORMS Joint International Workshop, Banff, May 2004.

[2] Accenture. Accenture technology vision 2014 report. online http://www.accenture.com/microsites/it-technology-trends-2014/Pages/tech-vision-report.aspx, 2014.

[3] Bernardetta Addis, Antonio Capone, Giuliana Carello, Luca G. Gianoli, and Brunilde Sanso. Energy management through optimized routing and device powering for greener communication networks. *IEEE/ACM Transactions on Networking*, 22(1):313–325, Feb. 2014.

[4] Charu C. Aggarwal, editor. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.

[5] Charu C Aggarwal and Philip S Yu. Online generation of association rules. In *Proceedings of 14th International Conference on Data Engineering*, pages 402–411. IEEE, 1998.

[6] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, Jun. 1993.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[8] R. Ahuja, J. Orlin, S. Pallottino, and M. Scutella. Dynamic shortest paths minimizing travel times and costs. In *Networks*, pages 197–205, 2003.

[9] Ehab S. Al-shaer. Programmable agents for active distributed monitoring. In *10th IFIP/ IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'99)*, 1999.

[10] Mohamed Ziad Albari. A taxonomy of runtime software monitoring systems. http://www.informatik.uni-kiel.de/~wg/Lehre/Seminar-SS05/Mohamed_Ziad_Albari/vortrag.pdf, 2005. Last accessed on November 1, 2016.

[11] James P. Allen, Kevin P. Barry, John M. Mccormick, and Ross A. Paul. Plan execution monitoring with distributed intelligent agents for battle command. In *Proceedings of SPIE*, page 5441, 2004.

[12] Alan Holliday Anthony Wren. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly (1970-1977)*, 23(3):333–344, 1972.

[13] John K. Antonio, Garng M. Huang, and Wei K. Tsai. A fast distributed shortest path algorithm for a class of hierarchically clustered data networks. *IEEE Transactions on Computers*, 41(6):710–724, Jun. 1992.

[14] P. Anussornnitisarn, S. Nof, and O. Etzion. Decentralized control of cooperative and autonomous agents for solving the distributed resource allocation problem. *International Journal of Production Economics*, pages 114–128, 2005.

[15] Fujiang Ao, Jing Du, Yuejin Yan, Baohong Liu, and Kedi Huang. An efficient algorithm for mining closed frequent itemsets in data streams. In *IEEE 8th International Conference on Computer and Information Technology (CIT) Workshops*, pages 37–42. IEEE, 2008.

[16] C. Archetti and M. G. Speranza. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1-2):3–22, 2012.

[17] Claudia Archetti and Maria Grazia Speranza. An overview on the split delivery vehicle routing problem. Operations Research Proceedings, pages 123–127. Springer Berlin Heidelberg, 2007.

[18] Claudia Archetti and MariaGrazia Speranza. The split delivery vehicle routing problem: A survey. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 103–122. Springer US, 2008.

[19] A. Arsie and E. Frazzoli. Efficient routing of multiple vehicles with no explicit communications. *International Journal of Robust and Nonlinear Control*, 18(2):154–164, Jan. 2007.

[20] S. Arunapuram, K. Mathur, and D. Solow. Vehicle routing and scheduling with full truckloads. In *Transportation Science*, pages 170–182, 2003.

[21] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberan, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical report, Universite Joseph Fourier, Grenoble, France, 1995.

[22] Chen Avin and Carlos Brito. Efficient and robust query processing in dynamic environments using random walk techniques. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 277–286, New York, NY, USA, 2004. ACM.

[23] A. Awasthi, S. Chauhan, Y. Lechevallier, M. Parent, and M. Proth. A data mining approach for adaptive path planning on large road networks. *Foundations of Computational Intelligence: Data mining, Special Issue, Springer Publications*, pages 297–320, 2009.

[24] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 28–39, 2003.

[25] Dionysios Barmpoutis and Richard M. Murray. Networks with the smallest average distance and the largest average clustering, 2010.

[26] Edwin M. Bartee. A holistic view of problem solving. *Management Science*, 20(4-part-i):439–448, 1973.

[27] Abder Rezak Benaskeur, Froduald Kabanza, Eric Beaudry, and Mathieu Beaudoin. A probabilistic planner for the combat power management problem. In *ICAPS*, pages 12–19, 2008.

[28] Ygal Bendavid, Élisabeth Lefebvre, Louis A. Lefebvre, and Samuel Fosso-Wamba. Key performance indicators for the evaluation of rfid-enabled b-to-b e-commerce applications: the case of a five-layer supply chain. *Information Systems and e-Business Management*, 7(1):1–20, 2008.

[29] Zhuming Bi, Li Da Xu, and Chengen Wang. Internet of things for enterprise systems of modern manufacturing. *IEEE Transactions on Industrial Informatics*, 10(2):1537–1546, May 2014.

[30] Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 2003.

[31] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, Mar. 2004.

[32] Murray Brewster. Canadian army has no idea how it lost three artillery shells worth $500,000 on way out of Afghanistan. *National Post*, Feb. 2015. online, http://news.nationalpost.com/2015/02/08/canadian-army-has-no-idea-how-it-lost-three-artillery-shells-worth-500000-on-way-out-of-afghanistan/.

[33] Doina Bucur and Mogens Nielsen. Concurrency, Graphs and Models. chapter Secure Data Flow in a Calculus for Context Awareness, pages 439–456. Springer-Verlag, 2008.

[34] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Access control for mobile agents: The calculus of boxed ambients. *ACM Transactions on Programming Languages and Systems*, 26(1):57–124, 2004.

[35] Peter Bühlmann and Torsten Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 22(4):477–505, Nov. 2007.

[36] Y. D. Cai, D. Clutter, G. Pape, J. Han, M. Welge, and L. Auvil. MAIDS: mining alarming incidents from data streams. In *In Proceedings of the 2004 ACM Int. Conf. Management of Data (SIGMOD)*, Jun. 2004.

[37] M. C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

[38] Eugenio Cesario, Carlo Mastroianni, and Domenico Talia. A multi-domain architecture for mining frequent items and itemsets from distributed data streams. *Journal of grid computing*, 12(1):153–168, 2014.

[39] Felix T.S. Chan and Niraj Kumar. Effective allocation of customers to distribution centres: A multiple ant colony optimization approach. *Robotics and Computer-Integrated Manufacturing*, 25(1):1 – 12, 2009.

[40] Joong Hyuk Chang and Won Suk Lee. Finding recently frequent itemsets adaptively over online transactional data streams. *Information Systems*, 31(8):849–869, 2006.

[41] Archie Chapman, Rosa Anna Micillo, Ramachandra Kota, and Nick Jennings. Decentralised dynamic task allocation using overlapping potential games. *The Computer Journal*, 53(9):1462 –1477, Oct. 2010.

[42] Peng Chen, Hongye Su, Lichao Guo, and Yu Qu. Mining fuzzy association rules in data streams. In *2nd International Conference on Computer Engineering and Technology (ICCET)*, volume 4, pages V4–153. IEEE, 2010.

[43] R. Chen, K. Sivakumar, and H. Kargupta. An approach to online bayesian learning from multiple data streams. In *Proceedings of Workshop on Mobile and Distributed Data Mining, PKDD '01*, pages 31–45, 2001.

[44] Si Chen, Bruce L. Golden, and Edward A. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4):318–329, 2007.

[45] Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W. Wah, and Jianyong Wang. Multi-dimensional regression analysis of time-series data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 323–334. VLDB Endowment, 2002.

[46] James Cheng, Yiping Ke, and Wilfred Ng. Maintaining frequent closed itemsets over a sliding window. *Journal of Intelligent Information Systems*, 31(3):191–215, 2008.

[47] David W. Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu, and Yongjian Fu. A fast distributed algorithm for mining association rules. In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*, DIS '96, pages 31–43, Washington, DC, USA, 1996. IEEE Computer Society.

[48] Yun Chi, Haixun Wang, S Yu Philip, and Richard R Muntz. Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, 10(3):265–294, Oct. 2006.

[49] Yun Chi, Haixun Wang, P.S. Yu, and Richard R. Muntz. Moment: maintaining closed frequent itemsets over a stream sliding window. In *4th IEEE International Conference on Data Mining*, pages 59–66, Nov. 2004.

[50] Koen Claessen. Safety property verification of cyclic synchronous circuits. *Electronic Notes in Theoretical Computer Science*, 88:55–69, Oct. 2004.

[51] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, Aug. 1964.

[52] Samet Çokpınar and Taflan İmre G undem. Positive and negative association rule mining on xml data streams in database as a service concept. *Expert Systems with Applications*, 39(8):7503–7511, 2012.

[53] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[54] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 1076–1085, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[55] Benoit Crevier, Jean-François Cordeau, and Gilbert Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

[56] Xuan Hong Dang, Vincent CS Lee, Wee Keong Ng, and Kok Leong Ong. Incremental and adaptive clustering stream data over sliding window. In *Database and Expert Systems Applications*, pages 660–674. Springer, 2009.

[57] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959.

[58] Sanjeeb Dash. Mixed integer rounding cuts and master group polyhedra. In *Combinatorial Optimization - Methods and Applications*, pages 1–32. 2011.

[59] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 635–644, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[60] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 453–465, 1998.

[61] Rocco De Nicola, Diego Latella, Michele Loreti, and Mieke Massink. Rate-based transition systems for stochastic process calculi. In *ICALP (2)*, pages 435–446, 2009.

[62] DEIS - OR Group. Vrplib: A vehicle routing problem library. www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html.

[63] N. Delgado, A. Q. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE Transactions on Software Engineering*, 30(12):859–872, Dec. 2004.

[64] Mahmood Deypir and Mohammad Hadi Sadreddini. A dynamic layout of sliding window for frequent itemset mining over data streams. *Journal of Systems and Software*, 85(3):746–759, 2012.

[65] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[66] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275–1335, 2008.

[67] Zvi Drezner, Jack Brimberg, Nenad Mladenović, and Said Salhi. New heuristic algorithms for solving the planar p-median problem. *Computers & Operations Research*, 62:296 – 304, 2015.

[68] Moshe Dror, Gilbert Laporte, and Pierre Trudeau. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239 – 254, 1994.

[69] Moshe Dror and Pierre Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[70] Edmund H Durfee. Distributed problem solving and planning. In *Multi-agent systems and applications*, pages 118–149. Springer, 2001.

[71] T. Eiter, E. Erdem, and W. Faber. Plan reversals for recovery in execution monitoring. In *10th International Workshop on NonMonotonic Reasoning*, Whistler, Canada, 2004.

[72] Chad Eschinger and C. Dwight Klappich. Market trends: Transportation management systems worldwide; 2007-2012. Press Release G00161482, Gartner Inc., Oct. 2008.

[73] Eric Fabre, Albert Benveniste, Stefan Haar, and Claude Jard. Distributed monitoring of concurrent and asynchronous systems. *Discrete Event Dynamic Systems*, 15(1):33–84, Mar. 2005.

[74] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Principles of Database Systems (PODS)*, pages 102–113, 2001.

[75] Reza Zanjirani Farahani, Masoud Hekmatfar, Alireza Boloori Arabani, and Ehsan Nikbakhsh. Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64(4):1096 – 1109, 2013.

[76] Reza Zanjirani Farahani, Masoud Hekmatfar, Behnam Fahimnia, and Narges Kazemzadeh. Hierarchical facility location problem: Models, classifications, techniques, and applications. *Computers & Industrial Engineering*, 68:104 – 117, 2014.

[77] T. Feder, P. Hell, S. Klein, and R. Motwani. Complexity of graph partition problems. In *Symposium on the Theory of Computing*, 1999.

[78] M. Fichtner, A. Grossmann, and M. Thielscher. Intelligent execution monitoring in dynamic environments. *Fundamenta Informaticae*, 57(2-4):371–392, 2003.

[79] Christopher L. Fleming, Stanley E. Griffis, and John E. Bell. The effects of triangle inequality on the vehicle routing problem. *European Journal of Operational Research*, 224(1):1 – 7, 2013.

[80] Adrian Francalanza, Andrew Gauci, and Gordon J. Pace. Distributed system contract monitoring. In *Formal Languages and Analysis of Contract-Oriented Software (FLACOS)*, pages 23–37, 2011.

[81] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:1–45, 1998.

[82] Christian Fritz. Monitoring the execution of optimal plans. In *The 17th International Conference on Automated Planning and Scheduling (ICAPS) Doctoral Consortium*, Sep. 22 2007.

[83] Christian Fritz and Sheila McIlraith. Monitoring plan optimality during execution. In *The 17th International Conference on Automated Planning and Scheduling (ICAPS)*, 2007.

[84] Frost & Sullivan. Expansive increase of smart-phone use creates a need for mobile data monitoring solutions. online, http://www.slideshare.net/FrostandSullivan/global-mobile-data-monitoring-market, Apr. 2014.

[85] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, Jun. 2005.

[86] Michel Gendreau, Gilbert Laporte, Christophe Musaraganyi, and Éric D. Taillard. A tabu search heuristic for the heterogenous fleet vehicle routing problem. *Compututers & Operations Research*, 26:1153–1173, Oct. 1999.

[87] A. Gerevini and I. Serina. Fast plan adaptation through planning graphs: Local and systematic search techniques. In *5th International Conference on Artificial Intelligence Planning Systems*, pages 112–121, Breckenridge, CO, USA, 2000.

[88] Heiko Gerlach. Partial communication and collusion with demand uncertainty. 2005.

[89] Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board*, page 18pp, Washington DC. (USA), Jan. 2003.

[90] B L Golden, E A Wasil, J P Kelly, and I-M Chao. *Metaheuristics in vehicle routing*, chapter Fleet Management and Logistics. Kluwer, Boston, 1998.

[91] Bruce Golden, S. Raghavan, and Edward A. Wasil. *The vehicle routing problem: latest advances and new challenges*. Operations research/Computer science interfaces series, 43. Springer, 2008.

[92] Oded Goldreich. Foundations of cryptography - a primer. *Foundations and Trends in Theoretical Computer Science*, 1(1), 2005.

[93] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286 – 300, 2015.

[94] Yue-Jiao Gong, Meie Shen, Jun Zhang, O. Kaynak, Wei-Neng Chen, and Zhi-Hui Zhan. Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination. *IEEE Transactions on Industrial Informatics*, 8(4):900–912, 2012.

[95] Gösta Grahne and Jianfei Zhu. High performance mining of maximal frequent itemsets. In *6th International Workshop on High Performance Data Mining (HPDM)*, 2003.

[96] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1347–1362, 2005.

[97] M. Gronalt, R. Hartl, and M. Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, pages 520–535, 2008.

[98] Barbara J. Grosz and Sarit Kraus. The evolution of shared plans. In *Foundations and Theories of Rational Agency*, pages 227–262. Kluwer Academic Publishers, 1998.

[99] François Grünewald and Andrea Binder. Inter-agency real-time evaluation in Haiti: 3 months afterthe earthquake. online, https://ochanet.unocha.org/p/Documents/Haiti_IA_RTE_1_final_report_en.pdf, Aug. 2010.

[100] Damon Gulczynski, Bruce Golden, and Edward Wasil. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers and Industrial Engineering*, 61(3):794 – 804, 2011.

[101] Damon J. Gulczynski. *Integer Programming-based Heuristics for Vehicle Routing Problems*. PhD thesis, Robert H. Smith School of Business, University of Maryland, USA, 2010.

[102] Xu Han, Huy Bui, Suvasri Mandal, Krishna R Pattipati, and David L Kleinman. Optimization-based decision support software for a team-in-the-loop experiment: Asset package selection and planning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(2):237–251, 2013.

[103] Xu Han, Manisha Mishra, Suvasri Mandal, Huy Bui, Diego Fernando Martınez Ayala, David Sidoti, Krishna R Pattipati, and David L Kleinman. Optimization-based decision support software for a team-in-the-loop experiment: Multilevel asset allocation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(8):1098–1112, 2014.

[104] Amin Hassanzadeh, Ala Altaweel, and Radu Stoleru. Traffic-and-resource-aware intrusion detection in wireless mesh networks. *Ad Hoc Networks*, 21(0):18 – 41, 2014.

[105] Shibo He, Jiming Chen, Peng Cheng, Yu (Jason) Gu, Tian He, and Youxian Sun. Maintaining quality of sensing with actors in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(9):1657–1667, 2012.

[106] G. Hoa, C. Leea, H. Lau, and A. Ip. A hybrid intelligent system to enhance logistics workflow: An olap-based ga approach. *International Journal of Computer Integrated Manufacturing*, pages 69 – 78, 2006.

[107] John H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, USA, 1992.

[108] Jen-Wei Huang, Su-Chen Lin, and Ming-Syan Chen. Dpsp: distributed progressive sequential pattern mining on the cloud. In *Advances in Knowledge Discovery and Data Mining*, pages 27–34. Springer, 2010.

[109] Ling Huang, Minos Garofalakis, Anthony D. Joseph, and Nina Taft. Approximate decision making in large-scale distributed systems.

[110] Industry Canada. The list transportation. *Canadian Investor Magazine*, 1(3):8–9, Jun. 2012.

[111] Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41(2):253–264, May 2007.

[112] Navendu Jain, Praveen Yalagandula, Michael Dahlin, and Yin Zhang. Self-tuning, bandwidth-aware monitoring for dynamic data streams. In *IEEE 25th International Conference on Data Engineering, (ICDE'09)*, pages 114–125. IEEE, 2009.

[113] Ahmad I. Jarrah and Jonathan F. Bard. Pickup and delivery network segmentation using contiguous geographic clustering. *JORS*, 62(10):1827–1843, 2011.

[114] Yosr Jarraya, Arash Eghtesadi, Mourad Debbabi, Ying Zhang, and Makan Pourzandi. Cloud calculus: Security verification in elastic cloud computing platform. In *The 2012 International Conference on Collaboration Technologies and Systems*, pages 447–454, 2012.

[115] Nan Jiang and Le Gruenwald. CFI-Stream: mining closed frequent itemsets in data streams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 592–597. ACM, 2006.

[116] Nan Jiang and Le Gruenwald. Research issues in data stream association rule mining. *SIGMOD Rec.*, 35(1):14–19, Mar. 2006.

[117] G. Kaminka, D. Pynadath, and M. Tambe. Monitoring deployed agent teams. In *Agents*, pages 308–315, 2001.

[118] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, Sep. 2004.

[119] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 24(7):881–892, Jul. 2002.

[120] Naim Kapucu and Vener Garayev. Collaborative decision-making in emergency and disaster management. *International Journal of Public Administration*, 34(6):366–375, 2011.

[121] Hillol Kargupta, Ruchita Bhargava, Kun Liu, Michael Powers, Patrick Blair, Samuel Bushra, James Dull, Kakali Sarkar, Martin Klein, Mitesh Vasa, and David Handy. Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 300–311, 2004.

[122] Hillol Kargupta and Byung-Hoon Park. A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments. *IEEE Transactions on Knowledge and Data Engineering*, 16:216–229, 2004.

[123] C. Dwight Klappich. Hype cycle for supply chain execution technologies, 2014. Press Release G00263207, Gartner Inc., Jul. 2014.

[124] Natallia Kokash. An introduction to heuristic algorithms. online, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.8050&rep=rep1&type=pdf, 2005.

[125] S Kotsiantis and D Kanellopoulos. Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering*, 32(1):71–82, Jan. 2006.

[126] S. Kucukpetek, F. Polat, and H. Ogztuzun. Multilevel graph partitioning: An evolutionary approach. *Journal of the Operational Research Society*, pages 549–562, 2005.

[127] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama. *Location science*. Springer, Germany, Feb. 2015. ISBN: 978-3-319-13110-8.

[128] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19), 2001.

[129] Miguel A. Lejeune and François Margot. Solving chance-constrained optimization problems with stochastic quadratic inequalities. *Operations Research*, 64(4):939–957, May 2016.

[130] V.R. Lesser and D.D Corkill. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*, 4(3):15–33, 1983.

[131] Carson Kai-Sang Leung and Boyu Hao. Mining of frequent itemsets from streams of uncertain data. In *IEEE 25th International Conference on Data Engineering (ICDE'09)*, pages 1663–1670. IEEE, 2009.

[132] Carson Kai-Sang Leung and Fan Jiang. Frequent itemset mining of uncertain data streams using the damped window model. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 950–955. ACM, 2011.

[133] Carson Kai-Sang Leung and Quamrul I. Khan. Dstree: a tree structure for the mining of frequent sets from data streams. In *6th International Conference on Data Mining, (ICDM'06)*, pages 928–932. IEEE, 2006.

[134] Jonathan Levin. Auction theory. [online], http://web.stanford.edu/~jdlevin/Econ%20286/Auctions.pdf, Oct. 2004.

[135] Steven J. Levine. Monitoring the execution of temporal plans for robotic systems. Master's thesis, Massachusetts Institute of Technology, 2011.

[136] Deren Li, Shuliang Wang, and Deyi Li. *Spatial Data Mining: Theory and Application*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[137] Hua-Fu Li, Chin-Chuan Ho, and Suh-Yin Lee. Incremental updates of closed frequent itemsets over continuous data streams. *Expert Systems with Applications*, 36(2):2451–2458, 2009.

[138] Hua-Fu Li and Suh-Yin Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert Systems with Applications*, 36(2):1466–1477, 2009.

[139] Hua-Fu Li, Suh-Yin Lee, and Man-Kwan Shan. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In *Proceedings of 1st International Workshop on Knowledge Discovery in Data Streams*, volume 39, 2004.

[140] Pu Li, Harvey Arellano-Garcia, and Günter Wozny. Chance constrained programming approach to process optimization under uncertainty. *Computers & Chemical Engineering*, 32(1):25–45, 2008.

[141] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, Jan. 2013.

[142] J. Ligatti, L. Bauer, and D. Walker. Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 4(1):2–16, 2005.

[143] Jay Ligatti and Srikar Reddy. A theory of runtime enforcement, with results. In *Proceedings of the 15th European conference on Research in computer security*, ESORICS'10, pages 87–100, Berlin, Heidelberg, 2010. Springer-Verlag.

[144] A. Lim and F. Wang. The multi-depot vehicle routing problem: A one-stage approach. *IEEE Transactions on Automation Science and Engineering*, 2, 2005.

[145] Xuejun Liu, Jihong Guan, and Ping Hu. Mining frequent closed itemsets from a landmark window over online data streams. *Computers & Mathematics with Applications*, 57(6):927–936, 2009.

[146] Dimitrios Lymberopoulos, Quentin Lindsey, and Andreas Savvides. An empirical characterization of radio signal strength variability in 3-D IEEE 802.15.4 networks using monopole antennas. In *Wireless Sensor Networks*, pages 326–341. Springer, 2006.

[147] S Mandal, XU Han, KR Pattipati, and DL Kleinman. Agent-based distributed framework for collaborative planning. In *Aerospace Conference, 2010 IEEE*, pages 1–11. IEEE, 2010.

[148] Amit Manjhi, Vladislav Shkapenyuk, Kedar Dhamdhere, and Christopher Olston. Finding (recently) frequent items in distributed data streams. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 767–778, Washington, DC, USA, 2005. IEEE Computer Society.

[149] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002.

[150] Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top-k queries over web-accessible databases. *ACM Transactions on Database Systems*, 29:319–362, Jun. 2004.

[151] Alfonso Pedraza Martinez, Sameer Hasija, and Luk Van Wassenhove. An operational mechanism design for fleet management coordination in humanitarian operations. INSEAD Working Paper No. 2010/87/TOM/INSEAD Social Innovation Centre, Oct. 2010.

[152] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[153] Andreas Mayr, Harald Binder, Olaf Gefeller, and Matthias Schmid. The evolution of boosting algorithms - from machine learning to statistical modelling. *Methods of Information in Medicine*, 53(6):419–427, Dec. 2014.

[154] J. McCarthy. Epistemological problems of artificial intelligence. In *5th International Joint Conference on Artificial Intelligence (IJCAI '77)*, Cambridge, MA, USA, 1977. Invited Talk.

[155] Fiona McNeill and Alan Bundy. Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *International Journal on Semantic Web and Information Systems*, 3(3):1–35, 2007.

[156] Mona Mehrandish, Chadi M. Assi, and Mourad Debbabi. A game theoretic model to handle network intrusions over multiple packets. In *ICC*, pages 2189–2194, 2006.

[157] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems (TODS)*, 31(3):1095–1133, 2006.

[158] Nicolas Meuleau and Marco Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 8:103–121, 2002.

[159] Roberto Micalizio. A distributed control loop for autonomous recovery in a multi-agent plan. In *International Joint Conferences on Artificial Intelligence Organization (IJCAI)*, pages 1760–1765, 2009.

[160] Michael Burkett. Key issues facing the supply chain industry. Press release, Gartner Inc., Phoenix, USA, May 2014.

[161] T. Miyamoto, K. Nakatyou, and S. Kumagai. Agent based planning method for an on-demand transportation system. In *Proceedings of the 2003 IEEE International Symposium on Intellignet Control*, 2003.

[162] T. Miyamoto, N. Tsujimoto, and S. Kumagai. A cooperative algorithm for autonomous distributed vehicle systems with finite buffer capacity. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A(11):3036–3044, 2005.

[163] Abdel-Illah Mouaddib and Shlomo Zilberstein. Knowledge-based anytime computation. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, pages 775–781, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[164] Arinto Murdopo. Distributed decision tree learning for mining big data streams. Master's thesis, Universitat Politechnica De Catalunya, BarcelonaTech, Barcelona, Spain, Jul. 2013.

[165] Karen L. Myers. CPEF: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, 1999.

[166] Shankar B Naik and Jyoti D Pawar. A quick algorithm for incremental mining closed frequent itemsets over data streams. In *Proceedings of the 2nd ACM IKDD Conference on Data Sciences*, pages 126–127. ACM, 2015.

[167] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization (SIOPT)*, 17(4):969–996, Dec. 2006.

[168] NEO. Vrp library. http://neo.lcc.uma.es/radi-aeb/WebVRP/.

[169] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[170] Tho Manh Nguyen, Josef Schiefer, and A. Min Tjoa. Sense & response service architecture (saresa): an approach towards a real-time business intelligence solution and its use for a fraud detection application. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, DOLAP '05, pages 77–86. ACM, 2005.

[171] Fatemeh Nori, Mahmood Deypir, and Mohamad Hadi Sadreddini. A sliding window based algorithm for frequent closed itemset mining over data streams. *Journal of Systems and Software*, 86(3):615–623, 2013.

[172] T. Oncan, S. N. Kabadi, and K. Nair. Vlsn search algorithms for partitioning problems using matching neighborhoods. *Journal of the Operational Research Society*, pages 388–398, 2008.

[173] Masahiro Ono, Marco Pavone, Yoshiaki Kuwata, and J. Balaram. Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Autonomous Robots*, 39(4):555–571, Dec. 2015.

[174] Masahiro Ono and Brian C. Williams. Decentralized chance-constrained finite-horizon optimal control for multi-agent systems. In *Proceedings of 49th IEEE Conference on Decision and Control*, 2010.

[175] Héctor J Ortiz-Peña, Rakesh Nagi, Moises Sudit, Michael D Moskal, Michael Dawson, James Fink, Timothy Hanratty, Eric Heilman, and Daniel Tuttle. From information needs to information gathering: A system optimization perspective to isr synchronization. In *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, May 2012. Proc. SPIE 8389, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III, 838912.

[176] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3):203–228, May 2006.

[177] Stephan Otto and Gabriella Kókai. Decentralized evolutionary optimization approach to the p-median problem. In *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 659–668. Springer Berlin Heidelberg, 2008.

[178] L. Ozdamar, E. Ekinci, and B. Kucukyazici. Emergency logistics planning in natural disasters. *Annals of Operations Research*, pages 217–245, 2004.

[179] Zeynep Ozyurt and Deniz Aksen. Solving the multi-depot location-routing problem with lagrangian relaxation. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, volume 37 of *Operations Research/Computer Science Interfaces Series*, pages 125–144. Springer US, 2007.

[180] Juryon Paik, Junghyun Nam, Ung Mo Kim, and Dongho Won. Association rule extraction from xml stream data for wireless sensor networks. *Sensors*, 14(7):12937–12957, 2014.

[181] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Transactions on Industrial Informatics*, 7(3):381–388, Aug. 2011.

[182] T. Palpanas, R. Sidle, R. Cochrane, and H. Pirahesh. Incremental maintenance for non-distributive aggregate functions. In *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002.

[183] P. Pantazopoulos, M. Karaliopoulos, and I. Stavrakakis. Distributed placement of autonomic internet services. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1702–1712, Jul. 2014.

[184] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 697–708. VLDB Endowment, 2005.

[185] Byung-Hoon Park and Hillol Kargupta. Distributed data mining: Algorithms, systems and applications. pages 341–358. Citeseer, 2002.

[186] David C. Parkes. Chapter 2, iterative combinatorial auctions: Achieving economic and computational efficiency phd thesis, univesity of pennsylvania, 2001.

[187] Jossef Perl. The multi-depot routing allocation problem. *American Journal of Mathematical and Management Sciences*, 7(1-2):7–34, 1987.

[188] Dennis K. Peters and David Lorge Parnas. Requirements-based monitors for real-time systems. *IEEE Transactions on Software Engineering*, 28:146–158, Feb. 2002.

[189] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53:73–88, 2005.

[190] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[191] Tifenn Rault, Abdelmadjid Bouabdallah, and Yacine Challal. Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks*, 67:104 – 122, 2014.

[192] Sujoy Ray, Mourad Debbabi, Mohamad Khaled Allouche, Nicolas Léchevin, and Micheline Bélanger. Energy-efficient monitor deployment in collaborative distributed setting. *IEEE Transactions on Industrial Informatics*, 12(1):112–123, 2016.

[193] Sujoy Ray, Andrei Soeanu, Jean Berger, and Mourad Debbabi. The multi-depot split-delivery vehicle routing problem: model and solution algorithm. *Knowledge-Based Systems*, 71(0):238 – 265, 2014.

[194] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the 3rd ACM Symposium on Cloud Computing*, page 7. ACM, 2012.

[195] Mohammed Saleh, Andrei Soeanu, Sujoy Ray, Mourad Debbabi, Jean Berger, and Abdeslem Boukhtouta. Mechanism design for decentralized vehicle routing problem. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 749–754. ACM, 2012.

[196] Said Salhi and Graham K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[197] Aysegul Sarac, Nabil Absi, and Stéphane Dauzère-Pérès. A literature review on the impact of {RFID} technologies on supply chain management. *International Journal of Production Economics*, 128(1):77 – 95, 2010.

[198] Vinaya Sawant and Ketan Shah. A survey of distributed association rule mining algorithms. *Journal of Emerging Trends in Computing and Information Sciences*, 5(5), 2014.

[199] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[200] Kirk Schloegel, George Karypis, and Vipin Kumar. Sourcebook of parallel computing. chapter Graph Partitioning for High-performance Scientific Simulations, pages 491–541. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[201] Alexander Schrijver. *Combinatorial optimization : polyhedra and efficiency*. Algorithms and combinatorics. Springer-Verlag, Berlin, Heidelberg, New York, N.Y., et al., 2003.

[202] Timothy T Schwartz, Yves-François Pierre, and Eric Calpas. Building assessments and rubble removal in quake-affected neighborhoods in haiti. Barr report, United States Agency for International Development, May 2011.

[203] Zuo-Jun Max Shen, Roger Lezhou Zhan, and Jiawei Zhang. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS Journal on Computing*, 23(3):470–482, 2011.

[204] Se Jung Shin and Won Suk Lee. On-line generation association rules over data streams. *Information and Software Technology*, 50(6):569–578, 2008.

[205] P. Shirani, M. A. Azgomi, and S. Alrabaee. A method for intrusion detection in web services based on time series. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 836–841, May 2015.

[206] Georgios Smaragdakis, Nikolaos Laoutaris, Kleomenis Oikonomou, Ioannis Stavrakakis, and Azer Bestavros. Distributed server migration for scalable internet service deployment. *IEEE/ACM Transactions on Networking*, 22(3):917–930, Jun. 2014.

[207] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:61–70, 1981.

[208] Andrei Soeanu, Mourad Debbabi, Dima Alhadidi, Makram Makkawi, Mohamad Allouche, Micheline Belanger, and Nicholas Lechevin. Transportation risk analysis using probabilistic model checking. *Expert Systems with Applications*, 2015.

[209] Andrei Soeanu, Sujoy Ray, Mourad Debbabi, Jean Berger, and Abdeslem Boukhtouta. A learning based evolutionary algorithm for distributed multi-depot VRP. In *Advances in Knowledge-Based and Intelligent Information and Engineering Systems - 16th Annual KES Conference, San Sebastian, Spain*, pages 49–58, 2012.

[210] Andrei Soeanu, Sujoy Ray, Mourad Debbabi, Jean Berger, Abdeslem Boukhtouta, and Ahmed Ghanmi. A decentralized heuristic for multi-depot split-delivery vehicle routing problem. In *IEEE International Conference on Automation and Logistics, (ICAL), Chongqing, China, 15-16 Aug., 2011*, pages 70–75.

[211] Mingjun Song and Sanguthevar Rajasekaran. A transaction mapping algorithm for frequent itemsets mining. *IEEE transactions on Knowledge and Data Engineering*, 18(4):472–481, 2006.

[212] Ayse Durukan Sonmez and Gino J Lim. A decomposition approach for facility location and relocation problem with uncertain number of future facilities. *European Journal of Operational Research*, 218(2):327–338, 2012.

[213] Alexander Souza. Combinatorial optimization. [online] https://www2.informatik.hu-berlin.de/alcox/lehre/lvws1011/coalg/combinatorial_algorithms.pdf, Jan. 2011.

[214] Staff Report. BTS says surface trade with NAFTA partners up 11.5 percent annually in January 2012. Logistics management, Bureau of Transportation Statistics, Mar. 2012.

[215] Chunhua Su and Kouichi Sakurai. A distributed privacy-preserving association rules mining scheme using frequent-pattern tree. In Changjie Tang, CharlesX. Ling, Xiaofang Zhou, NickJ. Cercone, and Xue Li, editors, *Advanced Data Mining and Applications*, volume 5139 of *Lecture Notes in Computer Science*, pages 170–181. Springer Berlin Heidelberg, 2008.

[216] Jimeng Sun, Spiros Papadimitriou, and Christos Faloutsos. Distributed pattern discovery in multiple streams. In Wee Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *PAKDD*, volume 3918 of *Lecture Notes in Computer Science*, pages 713–718. Springer, 2006.

[217] Jimeng Sun, Spiros Papadimitriou, and Christos Faloutsos. Distributed pattern discovery in multiple streams. In *Advances in Knowledge Discovery and Data Mining*, pages 713–718. Springer, 2006.

[218] Atsuo Suzuki and Zvi Drezner. The p-center location problem in an area. *Location Science*, 4:69 – 82, 1996.

[219] Chamseddine Talhi, Nadia Tawbi, and Mourad Debbabi. Execution monitoring enforcement under memory-limitation constraints. *Information and Computation*, 206(2-4):158–184, 2008.

[220] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Sliding window-based frequent pattern mining over data streams. *Information sciences*, 179(22):3843–3865, 2009.

[221] Keming Tang, Caiyan Dai, and Ling Chen. A novel strategy for mining frequent closed itemsets in data streams. *Journal of Computers*, 7(7):1564–1573, 2012.

[222] T. Tassa. Secure mining of association rules in horizontally distributed databases. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):970–983, Apr. 2014.

[223] Hetal Thakkar, Barzan Mozafari, and Carlo Zaniolo. Continuous post-mining of association rules in a data stream management system. *Post-Mining of Association Rules: Techniques for Effective Knowledge Extraction*, pages 116–132, 2009.

[224] Michael Thielscher. *Reasoning Robots: The Art and Science of Programming Robotic Agents (Applied Logic Series)*. Springer Netherlands, Oct. 2010.

[225] Manisha Thool and Preeti Voditel. Association rule generation in streams. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(5), May 2013.

[226] Wenhong Tian, Ruini Xue, Xu Dong, and Haoyan Wang. An approach to design and implement RFID middleware system over cloud computing. *International Journal of Distributed Sensor Networks*, 2013, May 2013.

[227] Frank A. Tillman and Thomas M. Cain. An upper bounding algorithm for the single and multiple terminal delivery problem. *Management Science*, 18(11):664–682, Jun. 1972.

[228] P. Toth and D. Vigo. *An overview of vehicle routing problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[229] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics, 2002.

[230] Marc Uetz. Descrete optimization 2010: Lecture 1. [online] http://wwwhome.math.utwente.nl/~uetzm/do/DO_Lecture1.pdf, 2010.

[231] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 639–644, New York, NY, USA, 2002. ACM.

[232] R. Van der Krogt and M. de Weerdt. Plan repair as an extension of planning. In *International Conference on Automated Planning and Scheduling*, pages 161–170, Monterey, California, USA, 2005.

[233] Manuela M. Veloso, Martha E. Pollack, and Michael T. Cox. Rationale-based monitoring for planning in dynamic environments. In *AIPS*, pages 171–180, 1998.

[234] S. Vijayarani and R. Prasannalakshmi. Comparative analysis of association rule generation algorithms in data streams. *International Journal on Cybernetics & Informatics (IJCI)*, 4(1), Feb. 2015.

[235] Tricia Wachtendorf, Bethany Brown, and Jose Holguin-Veras. Catastrophe characteristics and their impact on critical supply chains: problematizing materiel convergence and management following hurricane katrina. *Journal of Homeland Security and Emergency Management*, 10(2):497–520, 2013.

[236] En Tzu Wang and Arbee LP Chen. Mining frequent itemsets over distributed data streams by continuously maintaining a global synopsis. *Data Mining and Knowledge Discovery*, 23(2):252–299, 2011.

[237] Qian Wang, Rajan Batta, Joyendu Bhadury, and Christopher M. Rump. Budget constrained location problem with opening and closing of facilities. *Computers & Operations Research*, 30(13):2047 – 2069, 2003.

[238] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, Jun. 1998.

[239] Steven Way and Yufei Yuan. Transitioning from dynamic decision support to context-aware multi-party coordination: A case for emergency response. *Group Decision and Negotiation*, 23(4):649–672, 2014.

[240] David E. Wilkins, Thomas J. Lee, and Pauline Berry. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18:217–261, 2003.

[241] David E. Wilkins, Karen L. Myers, John D. Lowrance, and Leonard P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7(1):197–227, 1995.

[242] Ho Jin Woo and Won Suk Lee. estmax: Tracing maximal frequent item sets instantly over online transactional data streams. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1418–1431, 2009.

[243] Gang Wu, Huxing Zhang, Meikang Qiu, Zhong Ming, Jiayin Li, and Xiao Qin. A decentralized approach for mining event correlations in distributed system monitoring. *Journal of parallel and Distributed Computing*, 73(3):330–340, 2013.

[244] Fetahi Wuhib, Rolf Stadler, and Mike Spreitzer. Gossip-based resource management for cloud environments. In *2010 International Conference on Network and Service Management (CNSM)*, pages 1–8, 2010.

[245] Fetahi Zebenigus Wuhib. *Distributed Monitoring and Resource Management for Large Cloud Environments*. PhD thesis, KTH- Royal Institute of Technology, Stockholm, Sweden, 2010.

[246] Kefei Xin, Peng Cheng, and Jiming Chen. Multi-target localization in wireless sensor networks: a compressive sampling-based approach. *Wireless Comm. and Mobile Computing*, 15(5):801–811, 2015.

[247] Xiaolong Xue, Jinfeng Lu, Yaowu Wang, and Qiping Shen. Towards an agent-based negotiation platform for cooperative decision-making in construction supply chain. In *KES International Symposium (KES-AMSTA)*, 2007.

[248] Po Yang, Wenyan Wu, M. Moniri, and C.C. Chibelushi. Efficient object localization using sparsely distributed passive RFID tags. *IEEE Transactions on Industrial Electronics*, 60(12):5914–5924, Dec. 2013.

[249] Qianqian Yang, Shibo He, Junkun Li, Jiming Chen, and Youxian Sun. Energy-efficient probabilistic area coverage in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 64(1):367–377, 2015.

[250] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[251] Show-Jane Yen, Cheng-Wei Wu, Yue-Shi Lee, Vincent S Tseng, and Chaur-Heh Hsieh. A fast algorithm for mining frequent closed itemsets over stream sliding window. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 996–1002. IEEE, 2011.

[252] W. Yi and A. Kumar. Ant colony optimization for disaster relief operations. In *Transportation Research Part E: Logistics and Transportation Review*, pages 660–672, 2007.

[253] W.F. Young, K.A. Remley, C.L. Holloway, G. Koepke, D. Camell, J. Ladbury, and C. Dunlap. Radiowave propagation in urban environments with application to public-safety communications. *Antennas and Propagation Magazine, IEEE*, 56(4):88–107, Aug. 2014.

[254] B. Yu, Z. Z. Yang, and J.-X. Xie. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 62(1):183–188, 2011.

[255] Jeffery Xu Yu, Zhihong Chong, Hongjun Lu, and Aoying Zhou. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 204–215. VLDB Endowment, 2004.

[256] Jeffrey Xu Yu, Zhihong Chong, Hongjun Lu, Zhenjie Zhang, and Aoying Zhou. A false negative approach to mining frequent itemsets from high speed transactional data streams. *Information Sciences*, 176(14):1986–2015, 2006.

[257] Mohammed J Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[258] Mohammed J Zaki and Karam Gouda. Fast vertical mining using diffsets. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–335. ACM, 2003.

[259] Li Zeng, Ling Li, Lian Duan, Kevin Lu, Zhongzhi Shi, Maoguang Wang, Wenjuan Wu, and Ping Luo. Distributed data mining: A survey. *Information Technology and Management*, 13(4):403–409, 2012.

[260] Meng Zhang and Guojun Mao. An approach for distributed streams mining using combination of naÃŕve bayes and decision trees. In *Proceedings of The 3rd International Conference*

on Advances in Databases, Knowledge, and Data Applications, pages 29–33, St. Maarten, The Netherlands Antilles, Jan. 2011. IARIA.

[261] Xie Zhi-Jun, Chen Hong, and Cuiping Li. An efficient algorithm for frequent itemset mining on data streams. In *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 474–491. Springer, 2006.

[262] G. Zhou, H. Min, and M. Gen. The balanced allocation of customers to multiple distribution centers in the supply chain network: a genetic algorithm approach. In *Computers and Industrial Engineering*, pages 251–261, 2002.

Appendix

The case study has been modified from the E016-03m problem that was published by Golden *et al.* [90]. Originally the problem was designed for one depot (node 1). We multiplied the x-y coordinate of each node by 10 for better visibility in graphics.

Name: E016-03m-modifed; *Comment*: Christofides, Mingozzi and Toth, 1981; *Type*: CVRP; *Dimension*: 16; *Edge_ Weight_ Type*: EUC_2D; *Capacity*: 90; *Vehicles*: 3

NODE_ COORD_ SECTION	DEMAND_ SECTION	DEPOT_ SECTION
1: 300 400	1: 0	1,2,3
2: 370 520	2: 7	
3: 490 490	3: 30	
4: 520 640	4: 16	
5: 200 260	5: 9	
6: 400 300	6: 21	
7: 210 470	7: 15	
8: 170 630	8: 19	
9: 310 620	9: 23	
10: 520 330	10: 11	
11: 510 210	11: 5	
12: 420 410	12: 19	
13: 310 320	13: 29	
14: 50 250	14: 23	
15: 120 420	15: 21	
16: 360 160	16: 10	

Table 7.1: Benchmark on CVRP A-Set instances from Augerat *et al.*[21]- Part 1

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
A-n32-k5 (31)	410 (1)	5 (100)	0.82 (no)	784 (784)	791 (785.31)	0.17 (2.31)
A-n33-k5 (32)	446 (1)	5 (100)	0.892 (no)	661 (661)	669 (663.25)	0.34 (1.50)
A-n33-k6 (32)	541 (1)	6 (100)	0.901 (no)	742 (742)	745 (742.31)	0.06 (1.38)
A-n34-k5 (33)	460 (1)	5 (100)	0.92 (no)	778 (778)	791 (784.92)	0.92 (1.42)
A-n34-k5 (33)	460 (1)	5 (100)	0.92 (yes)	778 (780)	783 (782.25)	0.60 (2.25)
A-n36-k5 (35)	442 (1)	5 (100)	0.884 (no)	799 (799)	829 (818.53)	2.41 (4.13)
A-n36-k5 (35)	442 (1)	5 (100)	0.884 (yes)	799 (820)	820 (820.00)	2.60 (3.00)
A-n37-k5 (36)	407 (1)	5 (100)	0.814 (no)	669 (670)	691 (680.19)	1.68 (5.19)
A-n37-k6 (36)	570 (1)	6 (100)	0.95 (no)	949 (955)	972 (965.00)	1.70 (2.75)
A-n37-k6 (36)	570 (1)	6 (100)	0.95 (yes)	949 (948)	968 (958.00)	0.99 (5.00)
A-n38-k5 (37)	481 (1)	5 (100)	0.962 (no)	730 (730)	739 (731.80)	0.28 (3.40)
A-n38-k5 (37)	481 (1)	5 (100)	0.962 (yes)	730 (724)	745 (730.17)	0.08 (4.50)
A-n39-k5 (38)	475 (1)	5 (100)	0.95 (no)	822 (822)	830 (826.56)	0.58 (3.89)
A-n39-k5 (38)	475 (1)	5 (100)	0.95 (yes)	822 (825)	840 (829.71)	0.99 (4.86)
A-n39-k6 (38)	526 (1)	6 (100)	0.876 (no)	831 (833)	841 (834.79)	0.50 (5.50)
A-n39-k6 (38)	526 (1)	6 (100)	0.876 (yes)	831 (834)	834 (834.00)	0.40 (4.00)
A-n44-k6 (43)	570 (1)	6 (100)	0.95 (no)	937 (937)	955 (943.69)	0.73 (7.08)
A-n44-k6 (43)	570 (1)	6 (100)	0.95 (yes)	937 (937)	938 (937.33)	0.07 (4.33)
A-n45-k6 (44)	593 (1)	6 (100)	0.988 (no)	944 (948)	966 (952.88)	0.99 (6.63)
A-n45-k6 (44)	593 (1)	6 (100)	0.988 (yes)	944 (932)	943 (938.63)	-0.54 (9.13)
A-n45-k7 (44)	634 (1)	7 (100)	0.905 (no)	1146 (1151)	1164 (1157.91)	1.07 (6.82)
A-n45-k7 (44)	634 (1)	7 (100)	0.905 (yes)	1146 (1154)	1171 (1159.80)	1.24 (11.40)
A-n46-k7 (45)	603 (1)	7 (100)	0.861 (no)	914 (915)	948 (920.36)	0.75 (9.00)
A-n46-k7 (45)	603 (1)	7 (100)	0.861 (yes)	914 (926)	935 (929.00)	1.66 (10.20)
A-n48-k7 (47)	626 (1)	7 (100)	0.894 (no)	1073 (1073)	1112 (1101.62)	2.64 (13.08)
A-n48-k7 (47)	626 (1)	7 (100)	0.894 (yes)	1073 (1078)	1085 (1082.67)	0.97 (12.00)
A-n53-k7 (52)	664 (1)	7 (100)	0.948 (no)	1010 (1014)	1036 (1025.25)	1.52 (13.88)
A-n53-k7 (52)	664 (1)	7 (100)	0.948 (yes)	1010 (1008)	1023 (1015.75)	0.60 (16.50)
A-n54-k7 (53)	669 (1)	7 (100)	0.955 (no)	1167 (1173)	1190 (1180.13)	1.19 (16.88)
A-n54-k7 (53)	669 (1)	7 (100)	0.955 (yes)	1167 (1171)	1179 (1174.50)	0.69 (29.38)
A-n55-k9 (54)	839 (1)	9 (100)	0.932 (no)	1073 (1074)	1103 (1082.78)	0.93 (10.00)
A-n55-k9 (54)	839 (1)	9 (100)	0.932 (yes)	1073 (1074)	1093 (1082.00)	0.86 (16.71)
A-n60-k9 (59)	829 (1)	9 (100)	0.921 (no)	1354 (1357)	1377 (1364.45)	0.83 (27.18)
A-n60-k9 (59)	829 (1)	9 (100)	0.921 (yes)	1354 (1357)	1375 (1363.20)	0.72 (38.00)
A-n61-k9 (60)	885 (1)	9 (100)	0.983 (no)	1035 (1038)	1052 (1043.38)	0.84 (19.13)
A-n61-k9 (60)	885 (1)	9 (100)	0.983 (yes)	1034 (1022)	1028 (1025.63)	-0.74 (36.63)
A-n62-k8 (61)	733 (1)	8 (100)	0.916 (no)	1290 (1310)	1325 (1319.46)	2.28 (37.69)
A-n62-k8 (61)	733 (1)	8 (100)	0.916 (yes)	1290 (1314)	1321 (1316.67)	2.10 (42.33)
A-n63-k9 (62)	873 (1)	9 (100)	0.97 (no)	1616 (1630)	1648 (1633.13)	1.10 (26.13)
A-n63-k9 (62)	873 (1)	9 (100)	0.97 (yes)	1616 (1625)	1633 (1627.50)	0.76 (36.13)
A-n63-k10 (62)	932 (1)	10 (100)	0.932 (no)	1315 (1321)	1330 (1325.50)	0.86 (23.63)
A-n63-k10 (62)	932 (1)	10 (100)	0.932 (yes)	1315 (1312)	1329 (1321.50)	0.54 (38.25)
A-n64-k9 (63)	848 (1)	9 (100)	0.942 (no)	1402 (1427)	1450 (1437.13)	2.51 (29.63)
A-n64-k9 (63)	848 (1)	9 (100)	0.942 (yes)	1402 (1410)	1443 (1429.00)	1.95 (42.25)

Table 7.2: Benchmark on CVRP A-Set instances from Augerat *et al.*[21]- Part 2

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
A-n69-k9 (68)	845 (1)	9 (100)	0.938 (no)	1159 (1171)	1181 (1174.89)	1.39 (35.00)
A-n69-k9 (68)	845 (1)	9 (100)	0.938 (yes)	1159 (1168)	1179 (1174.00)	1.30 (38.57)
A-n80-k10 (79)	942 (1)	10 (100)	0.942 (no)	1764 (1799)	1823 (1806.33)	2.40 (67.78)
A-n80-k10 (79)	942 (1)	10 (100)	0.942 (yes)	1764 (1785)	1816 (1799.86)	2.03 (97.00)

Table 7.3: Benchmark on CVRP B-Set instances from Augerat *et al.*[21]-Part 1

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
B-n31-k5 (30)	412 (1)	5 (100)	0.824 (no)	672 (672)	675 (672.44)	0.08 (1.38)
B-n34-k5 (33)	457 (1)	5 (100)	0.914 (no)	788 (789)	789 (789.00)	0.20 (1.88)
B-n34-k5 (33)	457 (1)	5 (100)	0.914 (yes)	788 (782)	783 (782.50)	-0.65 (3.25)
B-n35-k5 (34)	437 (1)	5 (100)	0.874 (no)	955 (956)	979 (962.60)	0.86 (3.47)
B-n35-k5 (34)	437 (1)	5 (100)	0.874 (yes)	955 (976)	976 (976.00)	2.20 (4.00)
B-n38-k6 (37)	512 (1)	6 (100)	0.853 (no)	805 (805)	809 (806.80)	0.27 (4.73)
B-n38-k6 (37)	512 (1)	6 (100)	0.853 (yes)	805 (807)	807 (807.00)	0.30 (7.00)
B-n39-k5 (38)	440 (1)	5 (100)	0.88 (no)	549 (549)	571 (560.67)	2.11 (4.11)
B-n39-k5 (38)	440 (1)	5 (100)	0.88 (yes)	549 (550)	555 (552.57)	0.70 (4.29)
B-n41-k6 (40)	567 (1)	6 (100)	0.945 (no)	829 (834)	844 (838.20)	1.12 (4.50)
B-n41-k6 (40)	567 (1)	6 (100)	0.945 (yes)	829 (827)	839 (831.67)	0.37 (8.50)
B-n43-k6 (42)	521 (1)	6 (100)	0.868 (no)	742 (742)	749 (744.69)	0.42 (9.77)
B-n43-k6 (42)	521 (1)	6 (100)	0.868 (yes)	742 (741)	746 (743.33)	0.23 (11.67)
B-n44-k7 (43)	641 (1)	7 (100)	0.915 (no)	909 (909)	932 (925.79)	1.86 (8.71)
B-n44-k7 (43)	641 (1)	7 (100)	0.915 (yes)	909 (927)	933 (930.00)	2.30 (10.00)
B-n45-k5 (44)	486 (1)	5 (100)	0.972 (no)	751 (760)	772 (765.11)	1.90 (8.89)
B-n45-k5 (44)	486 (1)	5 (100)	0.972 (yes)	751 (758)	768 (763.29)	1.67 (11.57)
B-n45-k6 (44)	592 (1)	6 (100)	0.986 (no)	678 (678)	691 (682.50)	0.68 (8.88)
B-n45-k6 (44)	592 (1)	6 (100)	0.986 (yes)	678 (674)	677 (675.38)	-0.32 (10.13)
B-n50-k7 (49)	609 (1)	7 (100)	0.87 (no)	741 (741)	744 (741.71)	0.13 (15.14)
B-n50-k7 (49)	609 (1)	7 (100)	0.87 (yes)	741 (743)	744 (743.50)	0.40 (19.50)
B-n50-k8 (49)	735 (1)	8 (100)	0.918 (no)	1312 (1319)	1332 (1327.75)	1.26 (14.88)
B-n50-k8 (49)	735 (1)	8 (100)	0.918 (yes)	1312 (1293)	1330 (1314.38)	0.22 (23.88)
B-n51-k7 (50)	684 (1)	7 (100)	0.977 (no)	1032 (1032)	1047 (1036.75)	0.47 (11.00)
B-n51-k7 (50)	684 (1)	7 (100)	0.977 (yes)	1032 (1026)	1042 (1034.75)	0.31 (19.38)
B-n52-k7 (51)	606 (1)	7 (100)	0.865 (no)	747 (748)	753 (751.54)	0.62 (15.69)
B-n52-k7 (51)	606 (1)	7 (100)	0.865 (yes)	747 (751)	753 (752.00)	0.70 (16.67)
B-n56-k7 (55)	616 (1)	7 (100)	0.88 (no)	707 (709)	716 (712.93)	0.87 (25.87)
B-n56-k7 (55)	616 (1)	7 (100)	0.88 (yes)	707 (717)	717 (717.00)	1.40 (20.00)
B-n57-k7 (56)	697 (1)	7 (100)	0.995 (no)	1153 (1158)	1192 (1173.13)	1.75 (22.00)
B-n57-k7 (56)	697 (1)	7 (100)	0.995 (yes)	1153 (1147)	1159 (1153.00)	0.05 (32.38)
B-n57-k9 (56)	803 (1)	9 (100)	0.892 (no)	1598 (1601)	1628 (1612.25)	0.93 (15.25)
B-n57-k9 (56)	803 (1)	9 (100)	0.892 (yes)	1598 (1594)	1613 (1601.75)	0.29 (28.25)
B-n63-k10 (62)	922 (1)	10 (100)	0.922 (no)	1496 (1537)	1548 (1542.38)	3.05 (29.75)
B-n63-k10 (62)	922 (1)	10 (100)	0.922 (yes)	1496 (1484)	1547 (1515.25)	1.29 (38.13)
B-n64-k9 (63)	878 (1)	9 (100)	0.975 (no)	861 (867)	881 (875.75)	1.71 (30.50)
B-n64-k9 (63)	878 (1)	9 (100)	0.975 (yes)	861 (861)	869 (865.13)	0.54 (49.00)
B-n66-k9 (65)	861 (1)	9 (100)	0.956 (no)	1316 (1318)	1332 (1323.00)	0.60 (36.63)
B-n66-k9 (65)	861 (1)	9 (100)	0.956 (yes)	1316 (1315)	1322 (1318.38)	0.24 (46.88)
B-n67-k10 (66)	907 (1)	10 (100)	0.907 (no)	1032 (1065)	1078 (1072.70)	3.83 (36.90)
B-n67-k10 (66)	907 (1)	10 (100)	0.907 (yes)	1032 (1040)	1075 (1059.33)	2.58 (53.17)
B-n68-k9 (67)	837 (1)	9 (100)	0.93 (no)	1272 (1287)	1294 (1289.88)	1.44 (40.75)

Table 7.4: Benchmark on CVRP B-Set instances from Augerat *et al.*[21]-Part 2

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
B-n68-k9 (67)	837 (1)	9 (100)	0.93 (yes)	1272 (1270)	1290 (1281.00)	0.74 (52.50)
B-n78-k10 (77)	937 (1)	10 (100)	0.937 (no)	1221 (1237)	1254 (1244.13)	1.91 (76.00)
B-n78-k10 (77)	937 (1)	10 (100)	0.937 (yes)	1221 (1222)	1247 (1232.50)	0.97 (97.38)

Table 7.5: Benchmark on known CVRP instances: P-Set from Augerat *et al.*[21]

Problem (nodes)	totalDem (depots)	vehCnt (vehCap)	tightness (split)	bestKnown (minHeurVal)	maxHeurVal (avgHeurVal)	avgGap[%] (avgTime[sec])
P-n16-k8 (15)	246 (1)	8 (35)	1.13 (no)	450 (450)	450 (450.00)	0.00 (1.00)
P-n16-k8 (15)	246 (1)	8 (35)	1.13 (yes)	450 (440)	440 (440.00)	-2.20 (1.00)
P-n19-k2 (18)	310 (1)	2 (160)	1.03 (no)	212 (212)	212 (212.00)	0.00 (1.00)
P-n19-k2 (18)	310 (1)	2 (160)	1.03 (yes)	212 (205)	205 (205.00)	-3.40 (1.00)
P-n20-k2 (19)	310 (1)	2 (160)	1.03 (no)	216 (217)	217 (217.00)	0.50 (1.00)
P-n21-k2 (20)	298 (1)	2 (160)	1.07 (no)	211 (211)	211 (211.00)	0.00 (1.00)
P-n22-k2 (21)	308 (1)	2 (160)	1.03 (no)	216 (216)	216 (216.00)	0.00 (1.00)
P-n22-k8 (21)	22500 (1)	8 (3000)	1.06 (no)	603 (603)	603 (603.00)	0.00 (1.00)
P-n22-k8 (21)	22500 (1)	8 (3000)	1.06 (yes)	603 (575)	586 (577.38)	-4.38 (1.00)
P-n23-k8 (22)	313 (1)	8 (40)	1.02 (no)	529 (529)	533 (529.50)	0.10 (1.00)
P-n23-k8 (22)	313 (1)	8 (40)	1.02 (yes)	529 (511)	519 (512.75)	-3.15 (1.00)
P-n40-k5 (39)	618 (1)	5 (140)	1.13 (no)	458 (458)	464 (459.27)	0.29 (4.87)
P-n45-k5 (44)	692 (1)	5 (150)	1.08 (no)	510 (510)	520 (516.08)	1.22 (5.92)
P-n50-k10 (49)	951 (1)	10 (100)	1.05 (no)	696 (697)	707 (702.25)	0.92 (2.63)
P-n50-k10 (49)	951 (1)	10 (100)	1.05 (yes)	696 (692)	699 (696.25)	0.08 (4.50)
P-n50-k7 (49)	951 (1)	7 (150)	1.1 (no)	554 (556)	565 (560.07)	1.12 (8.47)
P-n50-k8 (49)	951 (1)	8 (120)	1.0 (no)	631 (638)	645 (641.38)	1.65 (4.50)
P-n50-k8 (49)	951 (1)	8 (120)	1.0 (yes)	631 (618)	622 (619.50)	-1.82 (10.38)
P-n51-k10 (50)	777 (1)	10 (80)	1.02 (no)	741 (741)	756 (747.25)	0.86 (3.50)
P-n51-k10 (50)	777 (1)	10 (80)	1.02 (yes)	741 (730)	739 (733.88)	-0.94 (4.50)
P-n55-k10 (54)	1042 (1)	10 (115)	1.1 (no)	694 (696)	702 (700.00)	0.89 (5.36)
P-n55-k15 (54)	1042 (1)	15 (70)	1.0 (no)	989 (996)	1067 (1024.75)	3.49 (14.00)
P-n55-k15 (54)	1042 (1)	15 (70)	1.0 (yes)	989 (922)	937 (928.00)	-6.52 (11.38)
P-n55-k7 (54)	1042 (1)	7 (170)	1.14 (no)	568 (575)	579 (576.20)	1.48 (16.07)
P-n60-k10 (59)	1134 (1)	10 (120)	1.05 (no)	744 (750)	756 (752.50)	1.16 (8.50)
P-n60-k10 (59)	1134 (1)	10 (120)	1.05 (yes)	744 (742)	755 (749.25)	0.74 (17.50)
P-n60-k15 (59)	1134 (1)	15 (80)	1.05 (no)	968 (975)	980 (976.25)	0.93 (9.50)
P-n60-k15 (59)	1134 (1)	15 (80)	1.05 (yes)	968 (965)	971 (968.25)	0.09 (10.38)
P-n65-k10 (64)	1219 (1)	10 (130)	1.06 (no)	792 (800)	806 (802.33)	1.32 (19.08)
P-n70-k10 (69)	1313 (1)	10 (135)	1.02 (no)	827 (835)	845 (837.25)	1.26 (28.13)
P-n70-k10 (69)	1313 (1)	10 (135)	1.02 (yes)	827 (825)	837 (830.25)	0.41 (43.25)
P-n76-k4 (75)	1364 (1)	4 (350)	1.02 (no)	593 (598)	614 (606.31)	2.24 (19.23)
P-n76-k5 (75)	1364 (1)	5 (280)	1.02 (no)	627 (630)	648 (638.43)	1.83 (22.07)
P-n101-k4 (100)	1458 (1)	4 (400)	1.09 (no)	681 (696)	735 (718.31)	5.23 (40.75)

Table 7.6: Solutions from SQ-Series with better cost than best known

Problem[sol]			SQ1 (Split) [1048]					
Route	0, 1, 2,0	(3)	Route	0,11, 3,0	(3)	Route	33,31,23,33	(3)
Serve	0,80,20,0	100	Serve	0,90,10,0	100	Serve	0,60,40, 0	100
Cost	14,10,10	34	Cost	28,14,14	56	Cost	20,10,10	40
Route	0, 8, 7,0	(3)	Route	0, 9, 1,0	(3)	Route	33,28,20,33	(3)
Serve	0,85,15,0	100	Serve	0,90,10,0	100	Serve	0,60,40, 0	100
Cost	14,10,10	34	Cost	28,14,14	56	Cost	20,10,10	40
Route	0, 3, 5,0	(3)	Route	0, 8,16,27,0	(4)	Route	33,18,26,33	(3)
Serve	0,80,20,0	100	Serve	0, 5,85,10,0	100	Serve	0,35,65, 0	100
Cost	14,10,10	34	Cost	14,14, 2,30	60	Cost	10,10,20	40
Route	0, 4, 6,0	(3)	Route	0, 6,14,25,0	(4)	Route	33,29,21,33	(3)
Serve	0,20,80,0	100	Serve	0,10,85, 5,0	100	Serve	0,60,40, 0	100
Cost	10,10,14	34	Cost	14,14, 2,30	60	Cost	20,10,10	40
Route	0, 2,10,0	(3)	Route	33,19,18,33	(3)	Route	33,25,17,33	(3)
Serve	0,40,60,0	100	Serve	0,75,25, 0	100	Serve	0,90,10, 0	100
Cost	10,10,20	40	Cost	14,10,10	34	Cost	28,14,14	56
Route	0, 4,12,0	(3)	Route	33,17,20,33	(3)	Route	33,32,24,33	(3)
Serve	0,40,60,0	100	Serve	0,80,20, 0	100	Serve	0,90,10, 0	100
Cost	10,10,20	40	Cost	14,10,10	34	Cost	28,14,14	56
Route	0, 5,13,0	(3)	Route	33,21,24,33	(3)	Route	33,19,27,33	(3)
Serve	0,40,60,0	100	Serve	0,20,80, 0	100	Serve	0,15,85, 0	100
Cost	10,10,20	40	Cost	10,10,14	34	Cost	14,14,28	56
Route	0,15, 7,0	(3)	Route	33,23,22,33	(3)	Route	33,22,30,33	(3)
Serve	0,55,45,0	100	Serve	0,20,80, 0	100	Serve	0,10,90, 0	100
Cost	20,10,10	40	Cost	10,10,14	34	Cost	14,14,28	56

Pb.[sol]			SQ2 (Split) [1588]					
Route	0, 8, 7,0	(3)	Route	49,20,17,49	(3)	Route	50,38,36,50	(3)
Serve	0,90,10,0	100	Serve	0,20,80, 0	100	Serve	0,90,10, 0	100
Cost	14,10,10	34	Cost	10,10,14	34	Cost	14,10,10	34
Route	0, 2, 1,0	(3)	Route	49,18,19,49	(3)	Route	50,33,34,50	(3)
Serve	0,20,80,0	100	Serve	0,60,40, 0	100	Serve	0,80,20, 0	100
Cost	10,10,14	34	Cost	10,10,14	34	Cost	14,10,10	34
Route	0, 4, 6,0	(3)	Route	49,21,24,49	(3)	Route	50,37,35,50	(3)
Serve	0,20,80,0	100	Serve	0,30,70, 0	100	Serve	0,20,80, 0	100
Cost	10,10,14	34	Cost	10,10,14	34	Cost	10,10,14	34
Route	0, 3, 5,0	(3)	Route	49,23,22,49	(3)	Route	50,40,39,50	(3)
Serve	0,80,20,0	100	Serve	0,20,80, 0	100	Serve	0,80,20, 0	100
Cost	14,10,10	34	Cost	10,10,14	34	Cost	14,10,10	34
Route	0,10, 2,0	(3)	Route	49,28,20,49	(3)	Route	50,44,36,50	(3)
Serve	0,60,40,0	100	Serve	0,60,40, 0	100	Serve	0,50,50, 0	100
Cost	20,10,10	40	Cost	20,10,10	40	Cost	20,10,10	40
Route	0, 4,12,0	(3)	Route	49,31,23,49	(3)	Route	50,34,42,50	(3)
Serve	0,40,60,0	100	Serve	0,60,40, 0	100	Serve	0,40,60, 0	100
Cost	10,10,20	40	Cost	20,10,10	40	Cost	10,10,20	40
Route	0,13, 5,0	(3)	Route	49,29,44,21,49	(4)	Route	50,39,47,50	(3)
Serve	0,60,40,0	100	Serve	0,65, 5,30, 0	100	Serve	0,40,60, 0	100
Cost	20,10,10	40	Cost	20, 2,12,10	44	Cost	10,10,20	40
Route	0,15, 7,0	(3)	Route	49,19,26,15,49	(4)	Route	50,37,45,50	(3)
Serve	0,50,50,0	100	Serve	0,30,65, 5, 0	100	Serve	0,40,60, 0	100
Cost	20,10,10	40	Cost	14,14, 2,22	52	Cost	10,10,20	40
Route	0,11, 3,0	(3)	Route	49,17,25,49	(3)	Route	50,35,43,50	(3)
Serve	0,90,10,0	100	Serve	0,10,90, 0	100	Serve	0,10,90, 0	100
Cost	28,14,14	56	Cost	14,14,28	56	Cost	14,14,28	56

Pb.[sol]	SQ2 (Split) [1588] Continued ...							
Route	0, 1, 9,0	(3)	Route	49,24,32,49	(3)	Route	50,33,41,50	(3)
Serve	0,10,90,0	100	Serve	0,20,80, 0	100	Serve	0,10,90, 0	100
Cost	14,14,28	56	Cost	14,14,28	56	Cost	14,14,28	56
Route	0,14,25, 6,0	(4)	Route	49,22,30,49	(3)	Route	50,40,48,50	(3)
Serve	0,85, 5,10,0	100	Serve	0,10,90, 0	100	Serve	0,10,90, 0	100
Cost	28, 2,16,14	60	Cost	14,14,28	56	Cost	14,14,28	56
Route	0,16,27,0	(3)	Route	49,19,27,49	(3)	Route	50,46,32,50	(3)
Serve	0,85,15,0	100	Serve	0,20,80, 0	100	Serve	0,85,15, 0	100
Cost	28, 2,30	60	Cost	14,14,28	56	Cost	28, 2,30	60

Pb.[sol]	SQ3 (Split) [2116]							
Route	0, 5, 3,0	(3)	Route	65,18,26,65	(3)	Route	66,48,39,66	(3)
Serve	0,15,85,0	100	Serve	0,45,55, 0	100	Serve	0,90,10, 0	100
Cost	10,10,14	34	Cost	10,10,20	40	Cost	28,22,10	60
Route	0, 7, 8,0	(3)	Route	65,31,23,65	(3)	Route	66,43,64,66	(3)
Serve	0,25,75,0	100	Serve	0,60,40, 0	100	Serve	0,85,15, 0	100
Cost	10,10,14	34	Cost	20,10,10	40	Cost	28, 2,30	60
Route	0, 2, 1,0	(3)	Route	65,28,20,65	(3)	Route	66,33,41,36,66	(4)
Serve	0,20,80,0	100	Serve	0,60,40, 0	100	Serve	0, 5,85,10, 0	100
Cost	10,10,14	34	Cost	20,10,10	40	Cost	14,14,22,10	60
Route	0, 4, 6,0	(3)	Route	65,21,29,44,65	(4)	Route	66,46,32,66	(3)
Serve	0,20,80,0	100	Serve	0,30,65, 5, 0	100	Serve	0,85,15, 0	100
Cost	10,10,14	34	Cost	10,10, 2,22	44	Cost	28, 2,30	60
Route	0,13, 5,0	(3)	Route	65,19,27,65	(3)	Route	67,55,54,67	(3)
Serve	0,55,45,0	100	Serve	0, 5,95, 0	100	Serve	0,15,85, 0	100
Cost	20,10,10	40	Cost	14,14,28	56	Cost	10,10,14	34
Route	0, 2,10,0	(3)	Route	65,17,25,65	(3)	Route	67,56,55,67	(3)
Serve	0,40,60,0	100	Serve	0,10,90, 0	100	Serve	0,90,10, 0	100
Cost	10,10,20	40	Cost	14,14,28	56	Cost	14,10,10	34
Route	0, 4,12,0	(3)	Route	65,24,32,65	(3)	Route	67,50,51,67	(3)
Serve	0,40,60,0	100	Serve	0,20,80, 0	100	Serve	0,20,80, 0	100
Cost	10,10,20	40	Cost	14,14,28	56	Cost	10,10,14	34
Route	0,26,15, 7,0	(4)	Route	65,22,30,65	(3)	Route	67,52,49,67	(3)
Serve	0,10,55,35,0	100	Serve	0,10,90, 0	100	Serve	0,25,75, 0	100
Cost	22, 2,10,10	44	Cost	14,14,28	56	Cost	10,10,14	34
Route	0, 9, 1,0	(3)	Route	66,37,40,66	(3)	Route	67,52,60,67	(3)
Serve	0,90,10,0	100	Serve	0,10,90, 0	100	Serve	0,35,65, 0	100
Cost	28,14,14	56	Cost	10,10,14	34	Cost	10,10,20	40
Route	0, 8,16,0	(3)	Route	66,33,34,66	(3)	Route	67,61,53,67	(3)
Serve	0,15,85,0	100	Serve	0,85,15, 0	100	Serve	0,60,40, 0	100
Cost	14,14,28	56	Cost	14,10,10	34	Cost	20,10,10	40
Route	0, 3,11,57,0	(4)	Route	66,37,35,66	(3)	Route	67,55,63,67	(3)
Serve	0, 5,85,10,0	100	Serve	0,10,90, 0	100	Serve	0,35,65, 0	100
Cost	14,14, 2,30	60	Cost	10,10,14	34	Cost	10,10,20	40
Route	0,14,25, 6,0	(4)	Route	66,38,39,66	(3)	Route	67,58,50,67	(3)
Serve	0,85, 5,10,0	100	Serve	0,90,10, 0	100	Serve	0,60,40, 0	100
Cost	28, 2,16,14	60	Cost	14,10,10	34	Cost	20,10,10	40
Route	65,24,21,65	(3)	Route	66,37,45,66	(3)	Route	67,54,62,67	(3)
Serve	0,70,30, 0	100	Serve	0,40,60, 0	100	Serve	0, 5,95, 0	100
Cost	14,10,10	34	Cost	10,10,20	40	Cost	14,14,28	56
Route	65,20,17,65	(3)	Route	66,36,44,66	(3)	Route	67,49,57,67	(3)
Serve	0,20,80, 0	100	Serve	0,50,50, 0	100	Serve	0,15,85, 0	100
Cost	10,10,14	34	Cost	10,10,20	40	Cost	14,14,28	56
Route	65,18,19,65	(3)	Route	66,34,42,66	(3)	Route	67,59,51,67	(3)
Serve	0,15,85, 0	100	Serve	0,45,55, 0	100	Serve	0,90,10, 0	100
Cost	10,10,14	34	Cost	10,10,20	40	Cost	28,14,14	56
Route	65,23,22,65	(3)	Route	66,47,39,66	(3)	Route	67,53,64,67	(3)
Serve	0,20,80, 0	100	Serve	0,60,40, 0	100	Serve	0,20,80, 0	100
Cost	10,10,14	34	Cost	20,10,10	40	Cost	10,22,28	60

Pb.[sol]			SQ9 (Split) [7047]					
Route	0,28,36,0	(3)	Route	0, 4, 1,0	(3)	Route	97,76,84,92,97	(4)
Serve	0,60,40,0	100	Serve	0,60,40,0	100	Serve	0,30,10,60, 0	100
Cost	40,10,50	100	Cost	10,10,14	34	Cost	40,10,10,60	120
Route	0,39,31,0	(3)	Route	0, 6,15,0	(3)	Route	97,82,90,47,97	(4)
Serve	0,60,40,0	100	Serve	0,50,50,0	100	Serve	0,30,65, 5, 0	100
Cost	50,10,40	100	Cost	14,14,20	48	Cost	50,10, 2,62	124
Route	0,21,29,37,0	(4)	Route	0, 9, 1,0	(3)	Route	97,72,88,97	(3)
Serve	0,20,60,20,0	100	Serve	0,50,50,0	100	Serve	0,10,90, 0	100
Cost	30,10,10,50	100	Cost	28,14,14	56	Cost	42,28,71	141
Route	0,26,34,18,0	(4)	Route	0, 8,16,0	(3)	Route	97,70,86,97	(3)
Serve	0,40,40,20,0	100	Serve	0,10,90,0	100	Serve	0,20,80, 0	100
Cost	40,10,20,30	100	Cost	14,14,28	56	Cost	42,28,71	141
Route	0,15,23,24,0	(4)	Route	0, 3,11,0	(3)	Route	97,65,81,97	(3)
Serve	0,10,30,60,0	100	Serve	0,50,50,0	100	Serve	0,10,90, 0	100
Cost	20,10,30,42	102	Cost	14,14,28	56	Cost	42,28,71	141
Route	0,22,30,0	(3)	Route	0, 6,14,0	(3)	Route	97,83,67,97	(3)
Serve	0,10,90,0	100	Serve	0,40,60,0	100	Serve	0,90,10, 0	100
Cost	42,14,57	113	Cost	14,14,28	56	Cost	71,28,42	141
Route	0,32,24,0	(3)	Route	0,12,20,0	(3)	Route	97,91,67,97	(3)
Serve	0,90,10,0	100	Serve	0,60,40,0	100	Serve	0,90,10, 0	100
Cost	57,14,42	113	Cost	20,10,30	60	Cost	85,42,42	169
Route	0,25,17,0	(3)	Route	0,13,21,0	(3)	Route	97,65,89,97	(3)
Serve	0,90,10,0	100	Serve	0,60,40,0	100	Serve	0,20,80, 0	100
Cost	57,14,42	113	Cost	20,10,30	60	Cost	42,42,85	169
Route	0,19,27,0	(3)	Route	0,10,18,0	(3)	Route	97,96,72,97	(3)
Serve	0,10,90,0	100	Serve	0,60,40,0	100	Serve	0,90,10, 0	100
Cost	42,14,57	113	Cost	20,10,30	60	Cost	85,42,42	169
Route	0,26,34,42,0	(4)	Route	0,14,22,0	(3)	Route	97,86,94,97	(3)
Serve	0,20,20,60,0	100	Serve	0,30,70,0	100	Serve	0,10,90, 0	100
Cost	40,10,10,60	120	Cost	28,14,42	84	Cost	71,14,85	170
Route	0,23,31,47,0	(4)	Route	0, 9,17,0	(3)	Route	97,52,55,97	(3)
Serve	0,30,20,50,0	100	Serve	0,40,60,0	100	Serve	0,60,40, 0	100
Cost	30,10,20,60	120	Cost	28,14,42	84	Cost	10,14,10	34
Route	0,37,45,0	(3)	Route	0,11,19,0	(3)	Route	97,53,56,97	(3)
Serve	0,40,60,0	100	Serve	0,40,60,0	100	Serve	0,10,90, 0	100
Cost	50,10,60	120	Cost	28,14,42	84	Cost	10,10,14	34
Route	0,20,36,44,0	(4)	Route	97,66,74,82,97	(4)	Route	97,53,51,97	(3)
Serve	0,20,20,60,0	100	Serve	0,10,60,30, 0	100	Serve	0,10,90, 0	100
Cost	30,20,10,60	120	Cost	30,10,10,50	100	Cost	10,10,14	34
Route	0,17,33,0	(3)	Route	97,68,76,84,97	(4)	Route	97,50,49,97	(3)
Serve	0,20,80,0	100	Serve	0,20,30,50, 0	100	Serve	0,20,80, 0	100
Cost	42,28,71	141	Cost	30,10,10,50	100	Cost	10,10,14	34
Route	0,24,40,0	(3)	Route	97,69,77,85,97	(4)	Route	97,54,55,97	(3)
Serve	0,20,80,0	100	Serve	0,20,40,40, 0	100	Serve	0,90,10, 0	100
Cost	42,28,71	141	Cost	30,10,10,50	100	Cost	14,10,10	34
Route	0,19,35,0	(3)	Route	97,79,87,97	(3)	Route	97,50,58,97	(3)
Serve	0,20,80,0	100	Serve	0,50,50, 0	100	Serve	0,40,60, 0	100
Cost	42,28,71	141	Cost	40,10,50	100	Cost	10,10,20	40
Route	0,22,38,0	(3)	Route	97,69,72,97	(3)	Route	97,53,61,97	(3)
Serve	0,10,90,0	100	Serve	0,40,60, 0	100	Serve	0,40,60, 0	100
Cost	42,28,71	141	Cost	30,30,42	102	Cost	10,10,20	40
Route	0,33,41,0	(3)	Route	97,60,68,65,97	(4)	Route	97,49,57,97	(3)
Serve	0,10,90,0	100	Serve	0,10,40,50, 0	100	Serve	0,10,90, 0	100
Cost	71,14,85	170	Cost	20,10,30,42	102	Cost	14,14,28	56
Route	0,35,43,0	(3)	Route	97,78,70,97	(3)	Route	97,63,71,97	(3)
Serve	0,10,90,0	100	Serve	0,90,10, 0	100	Serve	0,50,50, 0	100
Cost	71,14,85	170	Cost	57,14,42	113	Cost	20,10,30	60
Route	0,40,48,91,0	(4)	Route	97,72,80,97	(3)	Route	97,55,64,97	(3)
Serve	0,10,85, 5,0	100	Serve	0,10,90, 0	100	Serve	0,10,90, 0	100
Cost	71,14, 2,86	173	Cost	42,14,57	113	Cost	10,22,28	60

Pb.[sol]	SQ9 (Split) [7047] Continued...								
Route	0,46,89,0	(3)	Route	97,75,67,97	(3)	Route	97,60,62,97	(3)
Serve	0,85,15,0	100	Serve	0,90,10, 0	100	Serve	0,50,50, 0	100
Cost	85, 2,86	173	Cost	57,14,42	113	Cost	20,20,28	68
Route	0, 7, 8,0	(3)	Route	97,65,73,97	(3)	Route	97,59,66,97	(3)
Serve	0,60,40,0	100	Serve	0,10,90, 0	100	Serve	0,50,50, 0	100
Cost	10,10,14	34	Cost	42,14,57	113	Cost	28,22,30	80
Route	0, 5, 8,0	(3)	Route	97,63,71,79,87,95,97	(6)	Route	97,67,59,97	(3)
Serve	0,60,40,0	100	Serve	0,10,10,10,10,60, 0	100	Serve	0,60,40, 0	100
Cost	10,10,14	34	Cost	20,10,10,10,10,60	120	Cost	42,14,28	84
Route	0, 2, 3,0	(3)	Route	97,93,85,77,97	(4)	Route	97,62,70,97	(3)
Serve	0,60,40,0	100	Serve	0,60,20,20, 0	100	Serve	0,40,60, 0	100
Cost	10,10,14	34	Cost	60,10,10,40	120	Cost	28,14,42	84

Table 7.7: Solutions from SDVRP instances with better cost than best known

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
eilB76 (split) [1010]	Route	1,59,11,32,56,26,1	(6)	Route	1,46,30,16,58,55,14,28,1	(8)
	Serve	0,21,26,25, 7,14,0	93	Serve	0,21,12, 8,14,16,12,17,0	100
	Cost	20, 6,13,22, 9,33	103	Cost	14, 4,10, 4,14, 8, 7,16	77
	Route	1,76, 5,53,35,1	(5)	Route	1, 7,34, 2,57,24,64,1	(7)
	Serve	0,20,30,19,19,0	88	Serve	0,19,27,11,26, 6,11,0	100
	Cost	3, 5, 9, 4,10	31	Cost	9, 9, 8,16, 6, 9,22	79
	Route	1,27,13,41,18,1	(5)	Route	1,52,17,50,25,19,51,33,1	(8)
	Serve	0,18,16,33,20,0	87	Serve	0,12,19, 5,27,13,22, 2,0	100
	Cost	6, 8, 5, 7, 8	34	Cost	11, 9, 9, 7,13, 6, 8,22	85
	Route	1,68,47, 9,36, 8,1	(6)	Route	1,54,12,60,15,20,1	(6)
	Serve	0,30,27,16,10,15,0	98	Serve	0,22, 8,24,31,15,0	100
	Cost	5, 6, 5, 5, 5,14	40	Cost	23, 8,15,11, 9,23	89
	Route	1,69, 3,63,29,75,31,1	(7)	Route	1,22,62,70,48,49,31,1	(7)
	Serve	0,10,26,15,29,10,10,0	100	Serve	0,28,15, 8,19,20,10,0	100
	Cost	7, 7, 8, 6, 6, 7,14	55	Cost	27,11,14,11, 6, 7,14	90
	Route	1,73,40,10,33,45, 4,1	(7)	Route	1,30, 6,38,21,71,61,72,37,31,1	(10)
	Serve	0, 1,16,29,26,17,11,0	100	Serve	0, 1,21,14,22,11,13, 3,12, 2,0	99
	Cost	21, 5, 4, 7, 5, 3,20	65	Cost	18, 7, 7, 6, 6, 4, 5, 7,19,14	93
	Route	1,39,66,67,12,1	(5)	Route	1,74,2,44,42,43,65,23,63,1	(9)
	Serve	0,24, 9,37,29,0	99	Serve	0, 6,7,18,15,11,28,12, 3,0	100
	Cost	27, 5, 7, 7,29	75	Cost	21, 5,7, 4, 4, 9,14, 8,22	94
S51D2 (split) [707]	Route	1,46,34,40,31,11, 6,1	(7)			
	Serve	0,43,19,47,18,20,12,0	159			
	Cost	31, 7,14,12, 9,14,14	101			
	Route	1,13,38,16,45,18,48,1	(7)			
	Serve	0,17,25,41,17,45,15,0	160			
	Cost	8,10, 7, 6, 9, 9, 9	58			
	Route	1,33,12,39,10,50, 6,1	(7)			
	Serve	0,15,46,18,24,47, 9,0	159			
	Cost	10, 6, 7, 7, 6, 8,14	58			
	Route	1, 7,15,26,14,19,47,1	(7)			
	Serve	0,21,19,43,37,18,19,0	157			
	Cost	11,10, 6,13,14,16, 2	72			
	Route	1,28, 9,27,32,29, 2,1	(7)			
	Serve	0,24,22,18,37,23,33,0	157			
	Cost	8,14, 7,10, 6,16,14	75			
	Route	1,17,51,35,22,30, 3,1	(7)			
	Serve	0,18,20,18,47,33,23,0	159			
	Cost	22, 6, 6, 9, 7, 9,21	80			
	Route	1,49,24, 8,44,25,1	(6)			
	Serve	0,19,20,47,43,23,0	152			
	Cost	16, 9, 6,12,12,25	80			
	Route	1,23, 4,37,36,21,33,1	(7)			
	Serve	0,18,46,22,45,21, 8,0	160			
	Cost	21,12,12, 6, 7,22,10	90			
	Route	1,48, 5,42,20,41,43,1	(7)			
	Serve	0,18,28,36,32,20,18,0	152			
	Cost	9, 8,13, 5,11,16,31	93			

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
S51D3 (split) [953]	Route	1,13,48,19,1	(4)	Route	1, 2,23,21, 3,1	(5)
	Serve	0,25,76,59,0	160	Serve	0,18,72,43,27,0	160
	Cost	8, 6, 8,15	37	Cost	14, 7,15,12,21	69
	Route	1,33,12,39,1	(4)	Route	1, 6,50,31,35,10,39,1	(7)
	Serve	0,79,31,50,0	160	Serve	0,20,51,26,33,20, 7,0	157
	Cost	10, 6, 7,16	39	Cost	14, 8,10, 7, 9, 7,16	71
	Route	1,47,1	(2)	Route	1,24,44,25,1	(4)
	Serve	0,79,0	79	Serve	0,18,68,70,0	156
	Cost	2, 2	4	Cost	22,13,12,25	72
	Route	1, 7,15,26,19,1	(5)	Route	1,17,51,22,30,1	(5)
	Serve	0,31,74,23,20,0	148	Serve	0,32,78,21,25,0	156
	Cost	11,10, 6,11,15	53	Cost	22, 6, 8, 7,29	72
	Route	1,18,43, 5,1	(4)	Route	1,38,34,40,11,1	(5)
	Serve	0,20,72,68,0	160	Serve	0, 5,28,56,56,0	145
	Cost	17,14,16,17	64	Cost	18,18,14,10,28	88
	Route	1,49, 8,27, 9,1	(5)	Route	1, 4,37,36,21,1	(5)
	Serve	0,53,31,52,24,0	160	Serve	0,30,76,43,11,0	160
	Cost	16,11,11, 7,22	67	Cost	33,12, 6, 7,32	90
	Route	1,16,46,45,38,1	(5)	Route	1, 5,20,41,42,14,1	(6)
	Serve	0,29,43,61,27,0	160	Serve	0, 3,25,69,37,20,0	154
	Cost	25, 7,10, 7,18	67	Cost	17,15,11,12, 9,29	93
	Route	1,28, 9,32,29, 2,1	(6)			
	Serve	0,21,27,32,78, 2,0	160			
	Cost	8,14, 9, 6,16,14	67			
S51D4 (split) [1561]	Route	1,47, 33,1	(3)	Route	1,19,42,1	(3)
	Serve	0,36,124,0	160	Serve	0,89,71,0	160
	Cost	2, 9, 10	21	Cost	15,17,30	62
	Route	1,13, 48,1	(3)	Route	1,16, 46,13,1	(4)
	Serve	0,46,114,0	160	Serve	0,34,112,14,0	160
	Cost	8, 6, 9	23	Cost	25, 7, 23, 8	63
	Route	1,48, 5,1	(3)	Route	1, 9, 29, 2,1	(4)
	Serve	0,17,143,0	160	Serve	0, 5,127,28,0	160
	Cost	9, 8, 17	34	Cost	22,13, 16,14	65
	Route	1, 38,13,1	(3)	Route	1,51,22,17,12,1	(5)
	Serve	0,134,26,0	160	Serve	0,30,69,42,19,0	160
	Cost	18, 10, 8	36	Cost	26, 8,10,10,12	66
	Route	1,28,49, 7,1	(4)	Route	1,39,10,31,11,1	(5)
	Serve	0,58,81,21,0	160	Serve	0, 5,23,63,69,0	160
	Cost	8, 9, 9,11	37	Cost	16, 7, 8, 9,28	68
	Route	1, 15, 7,1	(3)	Route	1,47, 34,13,1	(4)
	Serve	0,127,33,0	160	Serve	0,16,136, 8,0	160
	Cost	18, 10,11	39	Cost	2,32, 27, 8	69
	Route	1, 2, 23,33,1	(4)	Route	1,39,31,35,51,1	(5)
	Serve	0,13,137,10,0	160	Serve	0,10,45,92,13,0	160
	Cost	14, 7, 12,10	43	Cost	16,15, 7, 6,26	70
	Route	1,39,50, 6,1	(4)	Route	1, 8, 44, 7,1	(4)
	Serve	0,62,70,28,0	160	Serve	0,27,117,16,0	160
	Cost	16, 8, 8,14	46	Cost	26,12, 23,11	72
	Route	1,19,26,15,1	(4)	Route	1,43,20,42,1	(4)
	Serve	0,54,94,12,0	160	Serve	0,63,40,57,0	160
	Cost	15,11, 6,18	50	Cost	31, 9, 5,30	75
	Route	1,38, 45,18,1	(4)	Route	1, 6,11, 40,1	(4)
	Serve	0, 9,106,45,0	160	Serve	0, 7,53,100,0	160
	Cost	18, 7, 9,17	51	Cost	14,14,10, 38	76
	Route	1, 7,25,24,1	(4)	Route	1,33, 3,21,36, 4,1	(6)
	Serve	0,19,93,48,0	160	Serve	0, 8,12,45,84,11,0	160
	Cost	11,14, 9,22	56	Cost	10,11,12, 7,10,33	83
	Route	1, 9, 27,1	(3)			
	Serve	0,36,124,0	160			
	Cost	22, 7, 28	57			

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
S51D4 (split) [1561] Continued...	Route	1, 3, 30,12,1	(4)	Route	1, 2, 4, 37,1	(4)
	Serve	0,31,118,11,0	160	Serve	0, 2,14,141,0	157
	Cost	21, 9, 17,12	59	Cost	14,19,12, 44	89
	Route	1,32, 9,1	(3)	Route	1,14, 41,42,1	(4)
	Serve	0,70,90,0	160	Serve	0,25,127, 8,0	160
	Cost	30, 9,22	61	Cost	29,19, 12,30	90
S51D5 (split) [1337]	Route	1,13,47,1	(3)	Route	1,18, 43,48,1	(4)
	Serve	0,70,90,0	160	Serve	0,40,111, 9,0	160
	Cost	8, 7, 2	17	Cost	17,14, 22, 9	62
	Route	1,28, 33,1	(3)	Route	1, 29,32,1	(3)
	Serve	0,52,108,0	160	Serve	0,108,52,0	160
	Cost	8, 8, 10	26	Cost	30, 6,30	66
	Route	1,48, 5,1	(3)	Route	1,50,35,10,1	(4)
	Serve	0,52,108,0	160	Serve	0,54,91,15,0	160
	Cost	9, 8, 17	34	Cost	22,14, 9,23	68
	Route	1,12, 39,1	(3)	Route	1,19,14,42,1	(4)
	Serve	0,53,104,0	157	Serve	0,47,59,54,0	160
	Cost	12, 7, 16	35	Cost	15,14, 9,30	68
	Route	1, 7,15,1	(3)	Route	1, 3,30,22,17,1	(5)
	Serve	0,94,66,0	160	Serve	0,33,60,64, 3,0	160
	Cost	11,10,18	39	Cost	21, 9, 7,10,22	69
	Route	1, 2,23,33,47,1	(5)	Route	1,24,44,25,1	(4)
	Serve	0,59,78, 3,20,0	160	Serve	0, 6,58,93,0	157
	Cost	14, 7,12, 9, 2	44	Cost	22,13,12,25	72
	Route	1,15,26,19,1	(4)	Route	1, 6,34,46,1	(4)
	Serve	0,45,63,52,0	160	Serve	0,23,56,66,0	145
	Cost	18, 6,11,15	50	Cost	14,21, 7,31	73
	Route	1,49, 8,24,1	(4)	Route	1, 3,21, 4,1	(4)
	Serve	0,60,54,46,0	160	Serve	0,36,68,56,0	160
	Cost	16,11, 6,22	55	Cost	21,12, 8,33	74
	Route	1,13,38,16,45,1	(5)	Route	1,10,31,40,1	(4)
	Serve	0, 8,49,52,51,0	160	Serve	0,34,70,55,0	159
	Cost	8,10, 7, 6,25	56	Cost	23, 8,12,38	81
	Route	1,17, 51,10,1	(4)	Route	1,18,20,41,42,1	(5)
	Serve	0,50,106, 4,0	160	Serve	0,22,52,79, 1,0	154
	Cost	22, 6, 6,23	57	Cost	17,17,11,12,30	87
	Route	1,47,50, 11, 6,1	(5)	Route	1,21,36, 37,1	(4)
	Serve	0, 1,24,109,26,0	160	Serve	0, 1,54,100,0	155
	Cost	2,19, 8, 14,14	57	Cost	32, 7, 6, 44	89
	Route	1,49,27, 9,1	(4)			
	Serve	0,11,51,96,0	158			
	Cost	16,13, 7,22	58			

Problem[sol]	Solution Details (segments):			tour serve / tour cost
S51D6 (split) [2182]	Route	1, 13,1	(2)	
	Serve	0,131,0	131	
	Cost	8, 8	16	
	Route	1, 48,1	(2)	
	Serve	0,140,0	140	
	Cost	9, 9	18	
	Route	1,28, 7,1	(3)	
	Serve	0,39,121,0	160	
	Cost	8, 9, 11	28	
	Route	1,12,33,1	(3)	
	Serve	0,77,83,0	160	
	Cost	12, 6,10	28	
	Route	1,28, 2,1	(3)	
	Serve	0,42,118,0	160	
	Cost	8, 8, 14	30	
	Route	1,12, 39,1	(3)	
	Serve	0,33,127,0	160	
	Cost	12, 7, 16	35	
	Route	1,19, 5,1	(3)	
	Serve	0,50,110,0	160	
	Cost	15, 8, 17	40	
	Route	1,18, 38,1	(3)	
	Serve	0,36,117,0	153	
	Cost	17, 5, 18	40	
	Route	1, 3,33,1	(3)	
	Serve	0,118,42,0	160	
	Cost	21, 11,10	42	
	Route	1,19, 15,1	(3)	
	Serve	0,47,113,0	160	
	Cost	15,10, 18	43	
	Route	1,33, 23,1	(3)	
	Serve	0,18,142,0	160	
	Cost	10,12, 21	43	
	Route	1,28, 9,1	(3)	
	Serve	0,20,140,0	160	
	Cost	8,14, 22	44	
	Route	1,24,49,1	(3)	
	Serve	0,78,82,0	160	
	Cost	22, 9,16	47	
	Route	1,15, 26,1	(3)	
	Serve	0,29,131,0	160	
	Cost	18, 6, 23	47	
	Route	1, 47,1	(2)	
	Serve	0,121,0	121	
	Cost	2, 2	4	
	Route	1,38, 45,1	(3)	
	Serve	0,26,134,0	160	
	Cost	18, 7, 25	50	
	Route	1, 6,50,10,1	(4)	
	Serve	0,33,76,51,0	160	
	Cost	14, 8, 6,23	51	
	Route	1,49, 8,1	(3)	
	Serve	0,46,114,0	160	
	Cost	16,11, 26	53	
	Route	1, 6,16,1	(3)	
	Serve	0,65,95,0	160	
	Cost	14,15,25	54	
	Route	1, 25,24,1	(3)	
	Serve	0,131,29,0	160	
	Cost	25, 9,22	56	
	Route	1, 27,28,1	(3)	
	Serve	0,139,21,0	160	
	Cost	28, 20, 8	56	
	Route	1,17, 51,10,1	(4)	
	Serve	0,34,118, 8,0	160	
	Cost	22, 6, 6,23	57	
	Route	1,50, 11,1	(3)	
	Serve	0,49,111,0	160	
	Cost	22, 8, 28	58	
	Route	1, 14,19,1	(3)	
	Serve	0,114,46,0	160	
	Cost	29, 14,15	58	
	Route	1,12, 30,17,1	(4)	
	Serve	0, 5,106,49,0	160	
	Cost	12,17, 9,22	60	
	Route	1, 42, 5,1	(3)	
	Serve	0,129,31,0	160	
	Cost	30, 13,17	60	
	Route	1,10, 31,1	(3)	
	Serve	0,25,135,0	160	
	Cost	23, 8, 31	62	
	Route	1,18, 43,1	(3)	
	Serve	0,35,123,0	158	
	Cost	17,14, 31	62	
	Route	1,16, 46,1	(3)	
	Serve	0,21,136,0	157	
	Cost	25, 7, 31	63	
	Route	1, 22,17,1	(3)	
	Serve	0,125,35,0	160	
	Cost	32, 10,22	64	
	Route	1,10, 35,1	(3)	
	Serve	0,29,131,0	160	
	Cost	23, 9, 32	64	
	Route	1,18, 20, 5,1	(4)	
	Serve	0,48,110, 2,0	160	
	Cost	17,17, 15,17	66	
	Route	1,29, 32,1	(3)	
	Serve	0,35,125,0	160	
	Cost	30, 6, 30	66	
	Route	1, 34, 6,1	(3)	
	Serve	0,142,18,0	160	
	Cost	34, 21,14	69	
	Route	1, 7, 44,24,1	(4)	
	Serve	0, 9,137,13,0	159	
	Cost	11,23, 13,22	69	
	Route	1, 29, 4,1	(3)	
	Serve	0,104,56,0	160	
	Cost	30, 9,33	72	
	Route	1, 21, 4,1	(3)	
	Serve	0,119,41,0	160	
	Cost	32, 8,33	73	
	Route	1,11, 40,1	(3)	
	Serve	0,27,133,0	160	
	Cost	28,10, 38	76	
	Route	1,30, 36,1	(3)	
	Serve	0,31,129,0	160	
	Cost	29,16, 39	84	
	Route	1,42, 41,20,1	(4)	
	Serve	0,13,139, 8,0	160	
	Cost	30,12, 11,32	85	
	Route	1, 4, 37,1	(3)	
	Serve	0,17,143,0	160	
	Cost	33,12, 44	89	

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
S76D2 (split) [1091]	Route	1,68,35,47,53, 5,1	(6)	Route	1,64,24,57, 2,7,1	(6)
	Serve	0,18,46,40,46,10,0	160	Serve	0,19,46,47,40,8,0	160
	Cost	5, 5, 2, 5, 9, 7	33	Cost	22, 9, 6,16,16,9	78
	Route	1,76,31,49,30,46,5,1	(7)	Route	1,75,29,23,62,22,31,1	(7)
	Serve	0,18,10,44,43,37,8,0	160	Serve	0,40,26,29,16,23,26,0	160
	Cost	3,11, 7, 6, 4, 7,7	45	Cost	20, 6, 9,12,11,13,14	85
	Route	1,69, 3,63,74,34,1	(6)	Route	1,27,59,11,32,66,39,1	(7)
	Serve	0,28,22,40,23,47,0	160	Serve	0, 2,29,17,39,19,47,0	153
	Cost	7, 7, 8, 5, 5,18	50	Cost	6,14, 6,13,20, 5,27	91
	Route	1,27, 8,36,20, 9,1	(6)	Route	1,2,44,42,43,65,1	(6)
	Serve	0,33,27,35,47,18,0	160	Serve	0,4,27,43,46,38,0	158
	Cost	6,10, 5, 7, 8,16	52	Cost	25,7, 4, 4, 9,43	92
	Route	1,18,41,10,40,73,13,1	(7)	Route	1,8,54,15,60,67,12,1	(7)
	Serve	0,13,37,44,31,16,19,0	160	Serve	0,7,26,18,20,47,42,0	160
	Cost	8, 7,10, 4, 5, 9,12	55	Cost	14,9, 7,11,15, 7,29	92
	Route	1,46, 6,38,37,48,1	(6)	Route	1,18,33,26,56,19,51,45,1	(8)
	Serve	0, 1,47,47,33,25,0	153	Serve	0, 9,29,35,21,23,29,14,0	160
	Cost	14,11, 7, 8, 6,27	73	Cost	8,14,12, 9,14, 6,10,21	94
	Route	1,28,16,58,14,55,53,1	(7)	Route	1,16,21,71,61,72,70,31,1	(8)
	Serve	0,43,44,20,35,16, 1,0	159	Serve	0, 2,22,46,45,18,22, 5,0	160
	Cost	16,12, 4, 9, 8,13,14	76	Cost	27,11, 6, 4, 5, 9,23,14	99
	Route	1, 7,17,50,25,45, 4,52,1	(8)			
	Serve	0,12,20,32,24,27,22,23,0	160			
	Cost	9,12, 9, 7,15, 3,10,11	76			
S76D3 (split) [1440]	Route	1,13,40,32,56,26,10,1	(7)	Route	1,46,30, 6,37,48,49,1	(7)
	Serve	0,19,23,20,38,31,21,0	152	Serve	0, 9,25,17,68,29, 7,0	155
	Cost	12,10,16,22, 9,10,24	103	Cost	14, 4, 7,10, 6, 6,21	68
	Route	1,76, 5,1	(3)	Route	1,17,50,25, 4,1	(5)
	Serve	0,75,76,0	151	Serve	0,21,46,61,24,0	152
	Cost	3, 5, 7	15	Cost	19, 9, 7,14,20	69
	Route	1,69, 7,1	(3)	Route	1,45,19,51,1	(4)
	Serve	0,68,62,0	130	Serve	0,19,67,74,0	160
	Cost	7, 5, 9	21	Cost	21,14, 6,30	71
	Route	1,68,35,53,1	(4)	Route	1,64,24,57,1	(4)
	Serve	0,44,59,55,0	158	Serve	0,52,20,75,0	147
	Cost	5, 5, 4,14	28	Cost	22, 9, 6,37	74
	Route	1,68, 9,47,1	(4)	Route	1,12,67,66,1	(4)
	Serve	0,17,64,79,0	160	Serve	0,21,44,79,0	144
	Cost	5,10, 5,11	31	Cost	29, 7, 7,32	75
	Route	1,18,33,41,1	(4)	Route	1,49,22,62,1	(4)
	Serve	0,30,79,47,0	156	Serve	0,32,32,78,0	142
	Cost	8,14, 9,14	45	Cost	21, 9,11,34	75
	Route	1,27,59,73,13,1	(5)	Route	1,44,42,43, 2,1	(5)
	Serve	0,19,57,57,27,0	160	Serve	0,26,23,79,27,0	155
	Cost	6,14, 5, 9,12	46	Cost	32, 4, 4,11,25	76
	Route	1,31,75,29, 3,1	(5)	Route	1,70,72,38,1	(4)
	Serve	0,38,32,76,14,0	160	Serve	0,73,69,17,0	159
	Cost	14, 7, 6,10,15	52	Cost	37, 9,10,32	88
	Route	1,52,34,74,63,1	(5)	Route	1, 3,63,23,65,1	(5)
	Serve	0,33,24,58,45,0	160	Serve	0,23, 6,46,77,0	152
	Cost	11,10, 5, 5,22	53	Cost	15, 8, 8,14,43	88
	Route	1,54,15,36,1	(4)	Route	1,46,30,21,71,61,1	(6)
	Serve	0,59,79,22,0	160	Serve	0, 2, 3,37,37,79,0	158
	Cost	23, 7,10,18	58	Cost	14, 4,18, 6, 4,43	89
	Route	1,59,11,39,1	(4)	Route	1, 8,60,20,55,53,1	(6)
	Serve	0, 2,77,72,0	151	Serve	0,22,45,20,67, 6,0	160
	Cost	20, 6, 7,27	60	Cost	14,24,18, 9,13,14	92
	Route	1,28,14,58,16,46,1	(6)			
	Serve	0,20,27,70,20,23,0	160			
	Cost	16, 7, 9, 4,13,14	63			

Problem[sol]	Solution Details (segments):			tour serve / tour cost
S76D4 (split) [2096]	Route	1,51,56,19,25,1	(5)	
	Serve	0,35,55,26,44,0	160	
	Cost	30,15,14,13,33	105	
	Route	1, 68,1	(2)	
	Serve	0,143,0	143	
	Cost	5, 5	10	
	Route	1, 69,1	(2)	
	Serve	0,138,0	138	
	Cost	7, 7	14	
	Route	1, 18,1	(2)	
	Serve	0,135,0	135	
	Cost	8, 8	16	
	Route	1, 8,1	(2)	
	Serve	0,143,0	143	
	Cost	14, 14	28	
	Route	1, 46, 5,1	(3)	
	Serve	0,126,34,0	160	
	Cost	14, 7, 7	28	
	Route	1,47, 9,36,1	(4)	
	Serve	0,31,41,88,0	160	
	Cost	11, 5, 5,18	39	
	Route	1,13, 73,1	(3)	
	Serve	0,21,139,0	160	
	Cost	12, 9, 21	42	
	Route	1, 7,74, 3,1	(4)	
	Serve	0,39,74,46,0	159	
	Cost	9,12, 9,15	45	
	Route	1,35,53, 14,28,1	(5)	
	Serve	0,24,17,102,17,0	160	
	Cost	10, 4, 9, 7,16	46	
	Route	1, 64,34,1	(3)	
	Serve	0,131,21,0	152	
	Cost	22, 6,18	46	
	Route	1,41,45, 4,1	(4)	
	Serve	0, 1,20,139,0	160	
	Cost	14, 9, 3, 20	46	
	Route	1,36, 54,1	(3)	
	Serve	0,15,143,0	158	
	Cost	18, 7, 23	48	
	Route	1,63,29,75,1	(4)	
	Serve	0,60,47,53,0	160	
	Cost	22, 6, 6,20	54	
	Route	1,75, 22,31,1	(4)	
	Serve	0,12,143, 5,0	160	
	Cost	20, 8, 13,14	55	
	Route	1,40,10,33,1	(4)	
	Serve	0,66,51,43,0	160	
	Cost	22, 4, 7,22	55	
	Route	1, 50,52,1	(3)	
	Serve	0,141,19,0	160	
	Cost	28, 17,11	56	
	Route	1, 5,30,48,49,1	(5)	
	Serve	0,17,42,73,28,0	160	
	Cost	7,11,11, 6,21	56	
	Route	1,30, 16,28,1	(4)	
	Serve	0,34,118, 8,0	160	
	Cost	18,10, 12,16	56	
	Route	1,28, 58,1	(3)	
	Serve	0,25,135,0	160	
	Cost	16,12, 28	56	
	Route	1,27, 12,1	(3)	
	Serve	0,24,133,0	157	
	Cost	6,24, 29	59	
	Route	1,53,55,20,1	(4)	
	Serve	0,32,41,82,0	155	
	Cost	14,13, 9,23	59	
	Route	1,13,40,11,59,1	(5)	
	Serve	0, 1,77,33,49,0	160	
	Cost	12,10,12, 6,20	60	
	Route	1, 2, 44,34,1	(4)	
	Serve	0,24,126,10,0	160	
	Cost	25, 7, 14,18	64	
	Route	1,41, 26,33,1	(4)	
	Serve	0,32,124, 4,0	160	
	Cost	14,19, 12,22	67	
	Route	1,75, 62,1	(3)	
	Serve	0,38,121,0	159	
	Cost	20,15, 34	69	
	Route	1, 76,1	(2)	
	Serve	0,124,0	124	
	Cost	3, 3	6	
	Route	1,39,66,67,12,1	(5)	
	Serve	0,25,49,72, 5,0	151	
	Cost	27, 5, 7, 7,29	75	
	Route	1,42,43, 2,1	(4)	
	Serve	0,96,27,37,0	160	
	Cost	36, 4,11,25	76	
	Route	1,36,15, 60,1	(4)	
	Serve	0,30,29,101,0	160	
	Cost	18,10,11, 38	77	
	Route	1,48,37,70,31,1	(5)	
	Serve	0,13,37,89,19,0	158	
	Cost	27, 6, 7,23,14	77	
	Route	1, 5, 72,37,1	(4)	
	Serve	0,14,139, 5,0	158	
	Cost	7,33, 7,33	80	
	Route	1,52,17,24,57, 2,1	(6)	
	Serve	0,13,25,35,73,14,0	160	
	Cost	11, 9,13, 6,16,25	80	
	Route	1,30,21,71,38,1	(5)	
	Serve	0, 8,37,87,28,0	160	
	Cost	18,18, 6, 9,32	83	
	Route	1, 2, 65,23,1	(4)	
	Serve	0,11,121,25,0	157	
	Cost	25,18, 14,30	87	
	Route	1, 6,61,38,1	(4)	
	Serve	0,36,98,24,0	158	
	Cost	25,18,12,32	87	
	Route	1,73, 32,66,1	(4)	
	Serve	0, 4,100,56,0	160	
	Cost	21,16, 20,32	89	

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
S101D3 (split) [1889]	Route	1,11,64,65,50,1	(5)			
	Serve	0,44,20,69,24,0	157			
	Cost	25, 9,14,13,44	105			
	Route	1, 2,21,67,72,66,36,1	(7)	Route	1, 7,94,86,92,62, 6,1	(7)
	Serve	0, 4,52,24,21,30,29,0	160	Serve	0,15, 3,40,35,31,36,0	160
	Cost	15,16, 9, 9,10,12,41	112	Cost	11, 9, 3, 3, 6, 7,21	60
	Route	1,54,27,29,1	(4)	Route	1,42,23,75,1	(4)
	Serve	0,29,74,57,0	160	Serve	0,22,79,59,0	160
	Cost	4, 8, 8, 6	26	Cost	29, 4, 3,25	61
	Route	1,14,95,1	(3)	Route	1,61,6,85,18,1	(5)
	Serve	0,74,64,0	138	Serve	0,48,1,24,79,0	152
	Cost	11, 4,12	27	Cost	18, 4,4, 6,30	62
	Route	1,90, 7,97,1	(4)	Route	1,34,82,10,52,1	(5)
	Serve	0,52,30,78,0	160	Serve	0,45,28,60,20,0	153
	Cost	9, 5, 4,15	33	Cost	25, 3, 6, 6,27	67
	Route	1,59, 3,41,1	(4)	Route	1,30,25,55,1	(4)
	Serve	0,59,33,68,0	160	Serve	0,58,60,39,0	157
	Cost	9, 9, 9,11	38	Cost	30, 7,10,23	70
	Route	1,28,70, 2,51,1	(5)	Route	1,33,91,11,1	(4)
	Serve	0,38,26,17,79,0	160	Serve	0,75,71,14,0	160
	Cost	5, 7, 4, 6,17	39	Cost	34, 4, 7,25	70
	Route	1,96,98,93,1	(4)	Route	1,88,43,44,16,58,1	(6)
	Serve	0,41,74,45,0	160	Serve	0,31,15,23,60,31,0	160
	Cost	15, 3, 3,18	39	Cost	18, 7, 9, 7, 7,23	71
	Route	1,41,22,73,1	(4)	Route	1,29,80,35,79,1	(5)
	Serve	0, 7,77,76,0	160	Serve	0, 8,24,36,76,0	144
	Cost	11, 7, 4,22	44	Cost	6,19,11, 5,31	72
	Route	1, 4,78,77,29,1	(5)	Route	1,63,12,20,1	(4)
	Serve	0,79,18,60, 3,0	160	Serve	0,34,68,52,0	154
	Cost	22, 3, 4, 9, 6	44	Cost	25, 8, 7,32	72
	Route	1,13,81,69,1	(4)	Route	1,43,15,45,1	(4)
	Serve	0,38,35,79,0	152	Serve	0,34,69,57,0	160
	Cost	15, 6, 2,21	44	Cost	25, 9, 6,32	72
	Route	1,100,94,99,1	(4)	Route	1, 9,47,46,84,1	(5)
	Serve	0, 48,60,52,0	160	Serve	0,33,70,17,27,0	147
	Cost	17, 3, 3,21	44	Cost	26, 9,11, 8,21	75
	Route	1,53, 8,89,1	(4)	Route	1,55,56,26,40, 5,1	(6)
	Serve	0,35,75,50,0	160	Serve	0,36,20,69, 4,29,0	158
	Cost	11,10, 6,19	46	Cost	23, 8, 4, 9, 9,25	78
	Route	1,19,83,1	(3)	Route	1,49,48,37,1	(4)
	Serve	0,77,79,0	156	Serve	0,57,21,79,0	157
	Cost	16, 9,23	48	Cost	28, 6, 7,41	82
	Route	1,60,38,101,99,93,1	(6)	Route	1,94,17,87,39,45,1	(6)
	Serve	0,20,33, 79, 3,23,0	158	Serve	0, 1,44,77,22,14,0	158
	Cost	18, 4, 3, 3, 3,18	49	Cost	20, 9, 6,13,11,32	91
	Route	1,32,71,31,1	(4)	Route	1,41,74,75,76,24,68,40,57,1	(9)
	Serve	0,33,48,59,0	140	Serve	0, 3,21, 7,22,44,20,21,22,0	160
	Cost	17, 7, 5,25	54	Cost	11, 9, 4, 4, 8,12,10, 7,29	94
S101D5 (split) [2814]	Route	1,52,72, 66,21,1	(5)	Route	1,73,75,23,1	(4)
	Serve	0,16,27,111, 6,0	160	Serve	0,25,48,87,0	160
	Cost	27,13,10, 21,32	103	Cost	22, 3, 3,27	55
	Route	1,12,65,50,1	(4)	Route	1,34, 80,1	(3)
	Serve	0,16,61,83,0	160	Serve	0,56,104,0	160
	Cost	34,13,13,44	104	Cost	25, 6, 26	57
	Route	1,54,29,1	(3)	Route	1,84,46, 9,1	(4)
	Serve	0,76,84,0	160	Serve	0, 8,75,77,0	160
	Cost	4, 7, 6	17	Cost	21, 8, 6,26	61
	Route	1,28, 70,1	(3)	Route	1, 55,56,1	(3)
	Serve	0,53,107,0	160	Serve	0,102,58,0	160
	Cost	5, 7, 12	24	Cost	23, 8,30	61

Problem[sol]	Solution Details (segments):			tour serve / tour cost		
	Route	1,59,41,1	(3)	Route	1,51,34,82,52,1	(5)
	Serve	0,91,69,0	160	Serve	0, 2, 4,71,83,0	160
	Cost	9, 4,11	24	Cost	17, 8, 3, 7,27	62
	Route	1,90, 7,95,1	(4)	Route	1,61, 6,85,18,1	(5)
	Serve	0,77,31,52,0	160	Serve	0,34,21,52,52,0	159
	Cost	9, 5, 3,12	29	Cost	18, 4, 4, 6,30	62
	Route	1,27,13,29,1	(4)	Route	1,57, 5,1	(3)
	Serve	0,84,57,19,0	160	Serve	0,71,89,0	160
	Cost	11, 7, 9, 6	33	Cost	29, 8,25	62
	Route	1,14, 98,1	(3)	Route	1,69,30, 25,1	(4)
	Serve	0,52,108,0	160	Serve	0, 1,57,102,0	160
	Cost	11, 6, 17	34	Cost	21, 9, 7, 30	67
	Route	1,53,19,1	(3)	Route	1,100,17,45,38,1	(5)
	Serve	0,77,83,0	160	Serve	0, 31,17,93,19,0	160
	Cost	11, 8,16	35	Cost	17, 12, 6,11,21	67
	Route	1,97,60,96,1	(4)	Route	1,83,49,48,1	(4)
	Serve	0,46,93,21,0	160	Serve	0,17,53,90,0	160
	Cost	15, 3, 4,15	37	Cost	23, 5, 6,34	68
	Route	1, 51, 2,1	(3)	Route	1,70,31, 33,1	(4)
	Serve	0,108,52,0	160	Serve	0, 4,45,111,0	160
	Cost	17, 6,15	38	Cost	12,13,10, 34	69
	Route	1,97,100,94, 7,1	(5)	Route	1,11,64,91,1	(4)
	Serve	0, 6, 46,98,10,0	160	Serve	0,24,72,64,0	160
	Cost	15, 2, 3, 9,11	40	Cost	25, 9, 4,32	70
	Route	1,41,22, 74,1	(4)	Route	1,83, 47, 9,1	(4)
	Serve	0,32,23,105,0	160	Serve	0,43,111, 6,0	160
	Cost	11, 7, 3, 20	41	Cost	23,13, 9,26	71
	Route	1,95,96,38,99,93,1	(6)	Route	1,62,17,87, 6,1	(5)
S101D5	Serve	0,10,31,31,38,50,0	160	Serve	0,19,47,63,31,0	160
(split)	Cost	12, 3, 6, 1, 3,18	43	Cost	25, 4, 6,16,21	72
[2814]	Route	1, 4,78,77,1	(4)	Route	1,32,12,20, 8,1	(5)
Continued...	Serve	0,62,91, 7,0	160	Serve	0,24,47,57,32,0	160
	Cost	22, 3, 4,16	45	Cost	17,17, 7,11,21	73
	Route	1,19, 84,61,1	(4)	Route	1,58,16,42,23,1	(5)
	Serve	0,28,101,31,0	160	Serve	0,13,61,69,17,0	160
	Cost	16, 7, 4,18	45	Cost	23, 7,12, 4,27	73
	Route	1,81,69,77,29,1	(5)	Route	1,40,26,56,1	(4)
	Serve	0,53,57,47, 3,0	160	Serve	0,56,55,49,0	160
	Cost	21, 2, 8, 9, 6	46	Cost	34, 9, 4,30	77
	Route	1, 3,58,88,1	(4)	Route	1,88,43, 44,15,1	(5)
	Serve	0,59,86,15,0	160	Serve	0,10,13,100,37,0	160
	Cost	18, 6, 7,18	49	Cost	18, 7, 9, 11,32	77
	Route	1,88,43,1	(3)	Route	1,29,80,79,35,10,1	(6)
	Serve	0,69,91,0	160	Serve	0, 4, 7,55,65,29,0	160
	Cost	18, 7,25	50	Cost	6,19, 6, 5,11,32	79
	Route	1,93,99,92,101,38,1	(6)	Route	1,31,21,67,1	(4)
	Serve	0, 2,21,70, 53,14,0	160	Serve	0,13,80,67,0	160
	Cost	18, 3, 4, 3, 3,21	52	Cost	25, 7, 9,40	81
	Route	1, 7,62,86,60,1	(5)	Route	1,8,49,48, 37,1	(5)
	Serve	0,22,87,50, 1,0	160	Serve	0,2,32,18,108,0	160
	Cost	11,14, 4, 5,18	52	Cost	21,7, 6, 7, 41	82
	Route	1,71,11,32,1	(4)	Route	1,98,15,39,1	(4)
	Serve	0,56,64,40,0	160	Serve	0, 2,69,89,0	160
	Cost	21, 8, 8,17	54	Cost	17,15,11,42	85
	Route	1,54,76,73,22,1	(5)	Route	1,52,72,36,10,1	(5)
	Serve	0,15,92,23,30,0	160	Serve	0,12,50,75,23,0	160
	Cost	4,23, 5, 4,18	54	Cost	27,13, 7, 9,32	88
	Route	1,89,63, 8,1	(4)	Route	1,73,76,24,68,40, 5,1	(7)
	Serve	0,52,79,29,0	160	Serve	0, 7, 7,50,50,32,14,0	160
	Cost	19, 6, 9,21	55	Cost	22, 5, 8,12,10, 9,25	91