Modeling and Analysis of the Generalized Warehouse Location Problem with Staircase Costs

Iman Niroomand

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Industrial Engineering) at

Concordia University

Montreal, Quebec, Canada

June 2008

Your file Votre référence
ISBN: 978-0-494-42523-7
Our file Notre référence
ISBN: 978-0-494-42523-7

ABSTRACT

Modeling and Analysis of the Generalized Warehouse Location Problem with Staircase Costs

Iman Niroomand

The Capacitated Warehouse Location consists of determining the number and locations of capacitated warehouses on a set of potential sites such that demands of predefined customers are met. Two typical assumptions in modeling this problem are: the capacity of warehouses is constant and that warehouses are able to truly satisfy customer demands. However, while these kinds of assumptions define a well structured problem from the mathematical modeling perspective, they are not realistic. In this thesis we relaxed such constraints based on the fact that warehouses can be built in various sizes and also warehouses can put in orders for unsatisfied customers' demand directly to the manufacturing plant with additional costs. This flexibility can lead to best decision making ability for managers and supply chain specialists to decide between higher capacity level with higher fixed and variable costs at the warehouse or direct ordering from the manufacturing plant. A new non linear integer programming formulation with staircase costs for multiple commodities in supply chain network is presented, and new method for linearizing the model is described. Computational results indicate that reasonably good solution can be obtained by the proposed linear model. Also for solving larger problems we developed a Tabu Search algorithm. The comparisons of the result between nonlinear/linear model and the Tabu Search algorithm are also presented.

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

I am deeply grateful to my supervisor, Dr.Bulgak and Dr.Bektas for their enthusiasm, inspiration, and great efforts to explain things clearly and simply. This thesis would not have been possible without their encouragement and useful comments.

My warm thanks are due to Dr.Defersha , my friends, and my colleagues for their precious comments and support in fulfillment of my thesis. I wish to thank my entire extended family for providing a loving environment for me. My brothers, my sister-in-law, my cousins were particularly supportive.

Lastly, and most importantly, I wish to thank my parents, Shahla Tabrizi and Bahaeddin Niroomand. They bore me, raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

Iman Niroomand, June 2008.

Table of Contents

Special Symbols

$G = (V, E)$ Where V is the set of the nodes and E is the set of edges,

V is partitioned as $V = \{0\} \cup W \cup C$ with W as the set of potential warehouse nodes and C as the customer nodes

$k \in K$, The set of commodities.

$j \in W$, The set of nodes for potential warehouses.

$m \in Q$, $Q = \{1,...,q\}$ The index set of capacity levels

$i \in C$, The set of customers

b_k, The size of commodity k

$f_{j,m}$, The cost of opening a warehouse on a potential node $j \in W$ at capacity level m.

$S_{j,m}$, Capacity of warehouse j at level m.

$d_{i,k}$, The amount of demand of customer i for commodity k

$h_{j,m,k}$, The holding cost of commodity k at warehouse j at capacity level m.

$Z^1_{j,m,k}$, The amount of commodity that is stored at each level m.

$AZ_{j,m,k}$, Aggregate of product k that is stored in warehouse j till level m.

$\hat{c}_{j,k}$, The lost opportunity cost of unsatisfied demand from warehouse(s).

$t_{j,m}$, Auxiliary Binary variable that run warehouse j with maximum opened level m.

$y_{j,m}$, Binary variable equal to 1 if warehouse j is opened at capacity level m.

$x_{i,j}$, Binary variable equal to 1 if customer i that is assigned to warehouse j, and 0 otherwise

$Z^1_{j,k}$, The total amount of commodity $k \in K$ stored at warehouse $j \in W$

$Z^2_{j,k}$, The total amount of commodity $k \in K$ requested from the plant by warehouse $j \in W$

List of Tables

List of Figures

Chapter 1: Introduction

The problem of locating warehouses and allocating customers covers the core component of distribution system design. The ability to produce and market plant product is dependent in part on the location of the warehouses and ability of customer demand fulfillment. Capacitated Warehouse location problem (CWLP) is defined as opening capacitated warehouses at some candidate locations in order that the total cost of meeting the customer demands is minimized.

The facility location problem is applicable in many sections such as industrial firms and assembly plants. It is applicable to government agencies which must decide about the location of offices, schools, hospitals and fire stations. Communication companies and air flight controllers also use this problem for servicing their customers. In every case, service quality depends on the location of the facilities in relation to other facilities and customers.

A very common assumption in most of the existing research is that the total capacity of all potential warehouses is sufficient to meet the total demand. Although this assumption helps define a well-structured problem from the mathematical modeling perspective, it is in fact restrictive and not realistic, hence rarely held in practice (Bektas and Bulgak, 2008). The modeling approach in this thesis breaks away from the existing research in relaxing this restrictive assumption.

Another approach to locate and build warehouses is motivated by the fact that these facilities can be built in different sizes. Therefore, there would be a tradeoff between choosing larger size warehouses and direct ordering from the manufacturing plant whenever demand is not fully satisfied.

This fact (what fact?) prompts us consider Staircase cost function for setting up new warehouse size for each potential location. In practice, there is often a need for considering several different possible sizes of each warehouse/plant. To deal with this situation (what situation?) we consider a facility location problem with staircase shaped costs. This approach not only will allow us to deal with different sizes, but also with different holding costs/production costs at different levels of production at a plant (Holmberg and Ling, 1997). For instance, consider a firm willing to operate warehouses in order to facilitate its distribution operation for multiple products. An appropriate warehouse size will have two advantages, first it eliminates extra cost of running large size warehouses, second it allows customer demands to be fully satisfied with minimum cost.

We consider the problem in a supply chain setting with multiple commodities and propose a model that simultaneously determines the number and the location of the warehouses which are opened among the set of potential locations (location problem), the assignment of customers to warehouses where their demand will be satisfied with minimum cost, the amount of products which are stored at each warehouse at appropriate

level by going through the staircase shaped costs and finally the most suitable size is selected for each nominated warehouse that leads to minimum cost for entire network .

This problem provides an opportunity for managers or supply chain specialists to come with a trade off between larger capacity size level of warehouse or direct ordering to plant by extra cost. This trade off at end converges to best minimum cost for decision maker in entire network system.

1-1 Contribution of this research

The focus of this thesis is on modeling and solution of a new issue in warehouse location problem with staircase costs that helps supply chain specialists and managers to develop better supply chain network by reducing the total cost of establishing warehouses, commodities and customer assignment. Not only this research considers opportunity of having warehouses with different levels but also each warehouse is capable of satisfying extra demands by direct ordering from plant, an issue that has not been studied before. As a matter of fact, by considering these two options (ordering to plant directly or having larger size warehouse) simultaneously we are able to bring a trade-off for decision maker.

We provide a literature review in chapter 2. Formal description of the problem along with mathematical notation and integer Non-linear programming formulation will be described in chapter 3. Integer linear model of problem will be described in Chapter 4.Chapter 5 shows the experimental problems with Lingo 8.0 for both non linear model and linear

model. Chapter 5 explains the Tabu Search frame work for generating feasible solution. Chapter 6 shows the Tabu Search implementation for problem and computation experiences. Tabu Search result and Lingo solution is compared in Chapter 7 and finally the conclusion and suggestions for further research comes in chapter 8.

Chapter 2: Literature Review

2-1 Facility location problem

There is an extensive literature on facility location problems. Klose and Drexl (2005) reviewed most cases in facility location models which have contributed to the current state-of-the-art. There are different models in facility locations that Klose and Drexl classified them in nine categories:

1. The shape or topography of models in the plane, network location models or mixed –integer programming models.
2. Minimum vs. Maximum objective function.
3. Models without capacity constraint vs. with capacity constraint.
4. Single stage models vs. multi-stage models.
5. Single product models vs. multi-product models
6. Inelastic demand vs. elastic demand.
7. Static models vs. dynamic models.
8. Deterministic models vs. probabilistic models.
9. Each pair supply and demand models vs. combined location models

Some of popular models in literature are:

- Continues location models.
- Network location models
- Mixed-integer programming models.

This review paper would be a comprehensive survey of the related problems for facility locations.

Holmberg *et al.* (1994) studied solving the staircase cost facility location problem with decomposition and piecewise linearization. Facility location problems with linear transportation costs and one fixed cost for each possible facility is the objective of this paper and staircase structured costs are introduced. Author uses staircase costs at several level of production. For obtaining solution a combination of piecewise linearization and Benders decomposition is used. This method provides the possibility of getting upper and lower bounds on the optimal objective function.

Sridharan (1995) reviewed heuristic and exact procedures for the capacitated plant locations problem. This author has studied scheduling problem for several machines for a given operation. The objective of this problem is minimizing the total purchase and fixed cost of operating the machines. The model of this problem is the same as the Capacitated Plant Location Problem (CPLP). The first stage of this problem chooses a subset of machines and the second stage of this model assigns the parts to the chosen machines. He examined different heuristic methods such as the greedy heuristic, and Lagrangean heuristic. For exact procedures alter methods such as LP relaxation and Benders decomposition has been used.

A Lagrangean heuristic for the facility location problem with staircase costs has proposed by Holmberg and Ling (1997). The authors developed a heuristic solution for the capacitated facility location problem with staircase shaped production cost functions. This approach gives this opportunity to deal with different sizes and different production

costs at different levels of production at a plant. A Lagrangean heuristic is used to obtain a near optimal dual solution.

A multi commodity, multi plant facility location problem has been studied by Pirkul *et al.* (1998) where a heuristic solution procedure was developed for a mixed integer programming model. In this proposed model customers get their multiple product of by open warehouses while warehouses receive these products from several manufacturing plants. The objective function of this model minimizes the fixed cost of establishing and operating the plants and the warehouses plus the variable cost of transporting units of products. For solving this model, Lagrangian relaxation of the model is provided and a heuristic solution procedure is introduced.

Hindi *et al.* (1999) studied Efficient solution of larger scale, single-source, capacitated plant location problems. The contribution of this paper is about assigning of all one particular customer demands to only one single plant. By this assumption the capacitated plant location problem reduces to the single-source plant location problem. The objective of this work is to develop a solution procedure capable of providing solution to large scale problems. For reaching this goal, a heuristic solution that combines Lagrangian relaxation with restricted neighbourhood search is provided that can solve large problem instances.

The capacitated plant location problem with multiple facilities in the same site is studied by Ghiani *et al*, (2002). The contribution of this paper is to consider several facilities in the same site such as the location of polling stations. A Lagrangian relaxation and a tailored Lagrangian heuristic are proposed in this paper.

Cortinhal and Captivo (2003) presented upper and lower bounds for the single source capacitated location problem with a Lagrangean relaxation. This paper considers a subset of plants and customers which each customer is assigned to one of these plants such that the total cost is minimized. The objective of this paper is to develop solution procedure can provide good solution for SSCLP. Therefore after presenting Lagrangian relaxation upper bounds are given by Lagrangian heuristics followed by search methods and by one Tabu Search meta-heuristic.

Lorena and Senne (2004) studied a column generation approach to capacitated p-median problems. The capacitated p-median problem (CPMP) tries to find optimal location of p facilities with regard to distances and capacities for the service to be given by each median. In this paper, Lagrangean relaxation directly identified from the master problem dual and provides new bounds and new productive columns through a modified knapsack sub problem.

Wu *et al.* (2006) expanded the model proposed by Gianpaolo Ghiani *et al.* They considered capacitated facility location problem with general setup cost which allows multiple facilities in the same site. This model is a mixed integer programming and the new features of this model considers the setup costs as fixed term plus a second term that depends on the size of the facility. Both Uncapacitated and Capacitated models are formulated in this paper and solved by general MIP solver. Also, a Lagrangean heuristic algorithm is proposed for solving the problem.

Keskin and Uster (2007) developed a meta-heuristic approach for a multi-product production/distribution system design problem. This mixed integer problem considers a multi-product, two-stage production/distribution system problem where a fixed number of capacitated distribution centers with attention to capacitated suppliers and retail locations are to be located to minimize the total costs. The authors provide meta-heuristic procedures such as population-based scatter search and tube search for the solution of the problem. This two-stage balances the amount of products that are transported to customer and the products which are received by DC (Distribution Center).

A branch-and-price algorithm for the capacitated facility location problem has been studied by Klose and Gortz (2007), where the authors employ column generation method in order to solve a corresponding master problem exactly. This approach is based on relaxing the demand constraints in a Lagrangean manner. A hybrid mixture of sub gradient optimization and a "weighted" decomposition method is applied for master

problem. They also use column generation procedure embedded in branch and price algorithm for computing optimal solution.

Bektas and Bulgak (2008) have developed Lagrangean based solution approaches for the generalized problem of locating capacitated warehouses. The novelty of this paper supports the relaxation of the assumption that the total capacity of all potential warehouses is sufficient to meet the total demand. The authors relax this assumption by having no restriction on the total capacity and the demand. A new integer programming formulation for this problem is presented, and algorithm based on Lagrangean relaxation and decomposition is described for its solution.

2-2 Tabu Search heuristic in location problem

The Tabu Search algorithm is a heuristic algorithm used to solve a variety of problems in operation research field such as scheduling, healthcare, facility location and production. Among large number of Tabu Search articles that exist in literature, we selected those applied to facility location problems which share common terms to our proposed model.

Rolland *et al.* (1996) considered an efficient Tabu Search procedure for the p-median problem. Their model investigates a set of nodes (facility) of size p in which the weighted sum of the distances is minimized. Some feature of used Tabu Search can be summarised here. First the search considers Add and Drop moves. Second to move from one local optima to another one efficiently where search path includes infeasible

solutions. Third they used random Tabu time. The result of Tabu Search shows Tabu Search algorithm performs better and other available heuristics.

Delmaire *et al.* (1999) studied the implementation of TB for the single source capacitated plant location problem. A reactive GRASP heuristic, a Tabu Search heuristic and two different hybridization schemes that combine the GRASP and Tabu Search methodologies are used in this paper. Two phases have been investigated in this paper: constructive phase which at this level different sets of open plants are selected and initial allocations within the open plants are obtained. Tabu Search is used as improving phase in second phase. Tabu Search provides a mechanism to strengthen the local search.

Gendron *et al.* (2003) studied a Tabu Search with slope scaling for the multi commodity capacitated location problem with balancing requirements. The authors have utilized slope scaling approach to provide initial solutions for the Tabu Search. This method takes into account the capacities and their impact on each move. The proposed version includes iterative procedure where a multi commodity network flow is solved at each iteration. Then this initial solution is improved by the Tabu Search.

Minghe Sun (2006) studied solving the uncapacitated facility location problem using Tabu Search. In this paper the Tabu Search performance is compared against the

Lagrangian method and heuristic method that exists in literature. The result of Tabu Search matches or dominates other competitive methods.

Keskin and Üster (2007) studied meta-heuristic approaches with memory and evolution for a multi-product production/distribution system design problem. They developed a mixed integer problem for a fixed number of capacitated distribution centers which are located with respect to capacitated suppliers and retail locations. They provided meta-heuristic procedures, including a population-based scatter search and trajectory-based local and Tabu Search for this model.

2-3 Literature Review Summary

In this chapter we reviewed facility location literature. We reviewed the existing models and the solution techniques to solve these models. From this review, we concluded that further improvements to the existing models can make the existing location problem more realistic. By considering this fact that every single facility can come with different level size with different fixed and variable costs can make the existing models more challenging. Holmberg (1996) who has considered staircase cost function in a production problem developed a mathematical model. By extending the proposed model of Bektas and Bulgak (2008), we could reach to new model formulation that will be introduced in Chapter 3.

.

Chapter 3: Problem Description and Model Formulation

The problem is formally defined as opening warehouses on a subset $\overline{W} \subseteq W$ with different potential sizes $m \in Q$, assigning each customer to a single opened warehouse and determining the amount of each commodity to be stored at each opened warehouse with specific size, such that the total cost of distributing the commodities to customers and holding cost of commodities at warehouses are minimized. If an opened warehouse $j \in W$ is unable to fully satisfy the demand of the customers assigned to it, then the demand is partially satisfied. Any amount of unsatisfied demand for commodity $k \in K$ is requested further from the production plant by warehouse $j \in W$ with an additional cost of producing/delivering the product as well as the lost opportunity cost of supplying in full the customer's demand.

Figure 3-1 shows a schematic illustration of proposed model.

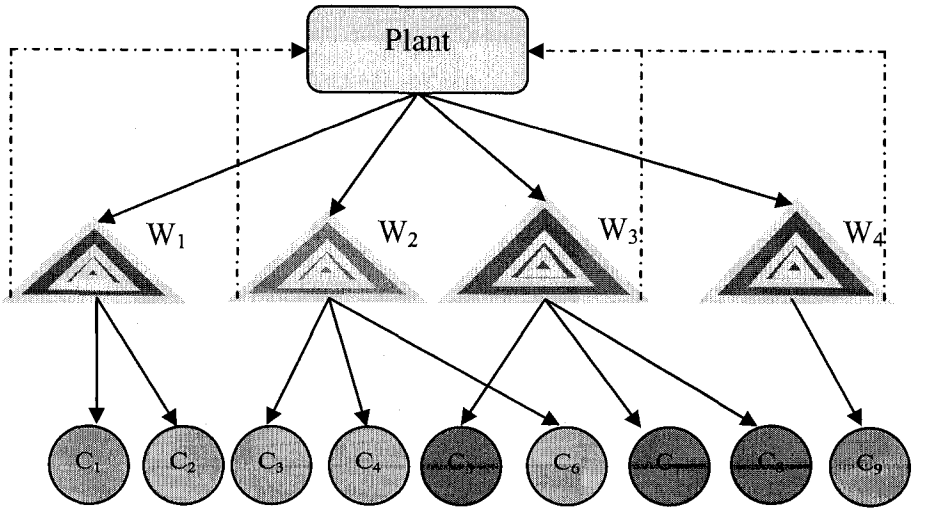


Figure 3-1 *Schematic of proposed model*

By appropriate decision level of warehouse capacity in competition market, the supply chain specialists and managers can have trade off between larger capacity size and direct order to plant. The right capacity can satisfy the customer demand fully and reduce the cost of fixed costs and variable costs of larger or smaller one. Therefore, we can improve the model by staircase structured costs. The staircase model allows fixed cost to appear at several levels of warehouse capacity, and also allows the linear holding cost coefficients to vary between different intervals of storage amount.

In reality a warehouse can be built in different sizes, finite set with not too many elements. Each possible size yields a certain fixed cost and a certain capacity of the facility. We thus have a cost function with several fixed costs at different levels ($f_{j,m}$) .This fixed cost appears for building a warehouse of size m at location j with possible sizes of $m \in Q$ at location j

We model the problem in an integer linear programming and define the following three sets of decision variables. The two first sets of binary variables associate with warehouse selection with specific size, and assignment of customers to the warehouses, respectively.

$$y_{j,m} = \begin{cases} 1 \text{ if node } j \in \mathcal{W} \text{ is selected as a warehouse with size } m, \\ 0 \text{ otherwise} \end{cases}$$

$$x_{i,j} = \begin{cases} 1 \text{ if customer i}\in\text{C is assigned to the (opened) warehouse j}\in\text{W}, \\ 0 \text{ otherwise} \end{cases}$$

If an opened warehouse $j \in W$ is unable to fully satisfy the demand of the customers assigned to it, then the demand is partially satisfied. Any amount of unsatisfied demand for commodity $k \in K$ is requested further from the production plant by this warehouse with an additional cost of producing/delivering the product as well as the lost opportunity cost of supplying in full the customer's demand.

The third set includes the following two variables that are related to the amount of commodities. The first variable in this set denotes the amount of commodities stored at each warehouse, and is defined as follows:

$$Z^1_{j,k} = \text{the amount of commodity k}\in\text{K stored at warehouse j}\in\text{W}$$

The second variable denotes the additional amount of requests that are made from a warehouse j∈W to the facility and is shown below.

$$Z^2_{j,k} = \text{the amount of commodity k}\in\text{K requested from the facility by warehouse j}\in\text{W}$$

Variables$Z^1_{j,k}$ and $Z^2_{j,k}$ are defined as non-negative general integers to denote the specific amount of commodities stored and transported.

For Staircase cost function term we define auxiliary variable$Z^1_{j,m,k}$ which is a non-negative integer variable and denotes the amount of product k∈K that is stored at

warehouse j size$m$. The total warehouse cost of level m is sum of fixed open cost and stored product holding cost. If $t_j$shows required space by warehouse j we have:

$$f_j(t_j) = \begin{cases} 0 & \text{if } t_j = 0 \\ f_{j,m} + \sum_{k \in K} h_{j,m,k} Z^1_{j.m.k} & \text{if } S_{j,m-1} < t_j \leq S_{j,m}, m \in Q \end{cases} \quad (3\text{-}1)$$

Where $S_{j,0} = 0$,

In order to increase the capacity from certain level to larger one we define:

$$\Delta S_{j,m} = S_{j,m} - S_{j,m-1} \; \forall j \in \mathcal{W}, m \in Q \quad (3\text{-}2)$$

And the cost of increasing capacity would be:

$$\Delta f_{j,m} = f_{j,m} - f_{j,m-1} + \sum_{k \in K} (h_{j,m,k} - h_{j,m-1,k}) \sum_{m \in Q} Z^1_{j,m-1,k} \; \forall j \in \mathcal{W}, m \in Q \quad (3\text{-}3)$$

As figure 2 shows, the cost of increasing the capacity from size S_1 to size S_2would increase the fix cost from f_1 to f_2 and inventory holding cost from $h_{j,1,k}$ to$h_{j,2,k}$.

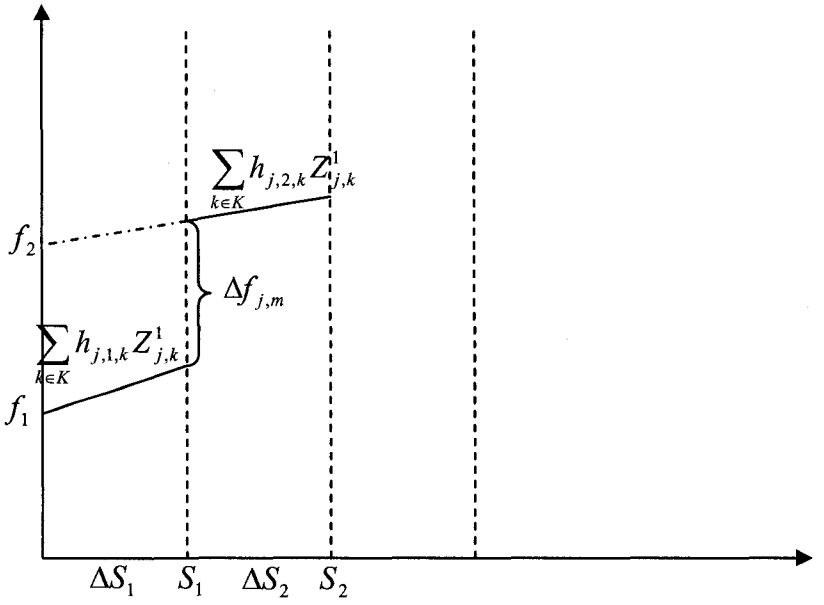


Figure 3-2 *staircase cost function*

Where$f_{j,0} = 0$ and $Z^1_{j,0,k} = 0$

Appropriate level of warehouse $j \in \mathcal{W}$ will be defined by the following formula:

$$\text{Let} \quad \sum_{k\in K} Z^1_{j,m,k} = \begin{cases} 0 & \text{if } t_j \le S_{j,m-1}, \\ t_j - S_{j,m-1} & \text{if } S_{j,m-1} < t_j \le S_{j,m} \\ \Delta S_{j,m} & \text{if } t_j \ge S_{j,m}, \end{cases} \tag{3-4}$$

Therefore we have:

$$t_j = \sum_{k\in K} b_k Z^1_{j,k} = \sum_{m\in Q} \sum_{k\in K} b_k Z^1_{j,m,k} \text{ where } 0 \le \sum_{k\in K} b_k Z^1_{j,m,k} \le \Delta S_{j,m} \ \forall j \in \mathcal{W}, m \in Q \tag{3-5}$$

3-1Non-linear mathematical model

Minimize:

$$\sum_{j\in W}\sum_{m\in Q}(\sum_{k\in K} h_{j,m,k}Z^1_{j,m,k} + \Delta f_{j,m}y_{j,m}) + \sum_{j\in W}\sum_{i\in C}\sum_{k\in K} d_{i,k}x_{i,j}c_{i,j} + \sum_{j\in W}\sum_{m\in Q} Z^2_{j,k}\hat{c}_{j,k} \quad (3\text{-}1\text{-}1)$$

Subject to

$$\sum_{j\in W} x_{i,j} = 1, \forall i \in C \quad (3\text{-}1\text{-}2)$$

$$x_{i,j} \leq y_{j,1}, \forall i \in C, j \in W \quad (3\text{-}1\text{-}3)$$

$$\sum_{k\in K} b_k Z^1_{j,m,k} \leq \Delta S_{j,m}y_{j,m}, \forall j \in W, m \in Q \quad (3\text{-}1\text{-}4)$$

$$\sum_{k\in K} b_k Z^1_{j,m-1,k} \geq \Delta S_{j,m-1}y_{j,m}, \forall j \in W, m \in Q \quad (3\text{-}1\text{-}5)$$

$$Z^1_{j,k} = \sum_{m\in Q} Z^1_{j,m,k}, \forall j \in W, k \in K \quad (3\text{-}1\text{-}6)$$

$$Z^1_{j,k} + Z^2_{j,k} = \sum_{i\in C} d_{i,k}x_{i,j}, \forall j \in W, k \in K \quad (3\text{-}1\text{-}7)$$

$$x_{i,j} \in \{0,1\}, \forall i \in C, j \in W \quad (3\text{-}1\text{-}8)$$

$$y_{j,m} \in \{0,1\}, \forall j \in W, m \in Q, \quad (3\text{-}1\text{-}9)$$

$$Z^1_{j,k}, Z^2_{j,k} \in \mathbb{Z}^+, \forall j \in \mathcal{W}, k \in K \qquad (3\text{-}1\text{-}10)$$

$$Z^1_{j,m,k} \in \mathbb{Z}^+, \forall j \in \mathcal{W}, m \in \mathcal{Q}, k \in K \qquad (3\text{-}1\text{-}11)$$

This is a nonlinear MIP capacitated facility location problem with staircase cost function. The objective function of this problem is composed of three cost elements. The first part is the total cost of opening the warehouse with a specific capacity from the available capacities and holding cost of each product at that particular warehouse level. The second term denotes the total shipping cost of each product to each customer that has been assigned to a specific warehouse. The last part shows the cost of ordering the product $k \in K$directly from the production plant that cannot be satisfied by the warehouse due to the capacity restriction. In objective function, the non-linearity term causes by different cost between the two continue level of capacity and decision of opening the higher level $(\Delta f_{j,m})$.

In this model constraint (3-1-2) assigns each customer to only one warehouse, and constraint (3-1-3) implies that customers are assigned only to warehouses that already exist. Constraint sets (3-1-4) and (3-1-5) ensure that the level of storage corresponds to the correct level on the staircase cost function for each warehouse. Number (3-1-6) keeps amount of product $k \in K$ in all level $m \in Q$ for warehouse $j \in \mathcal{W}$ .Constraint (3-1-7) ensures that the demand for all the customers will be met, either by the warehouse or the production facility.

3-2 Linear Mathematical model

In non-linear model we had:

$$\sum_{j\in W}\sum_{m\in Q}\Delta f_{j,m}y_{j,m} = \tag{3-2-1}$$

$$\sum_{j\in W}\sum_{m\in Q}(f_{j,m} - f_{j,m-1} + \sum_{k\in K}(h_{j,m,k} - h_{j,m-1,k}))y_{j,m}\ \forall j \in \mathcal{W}, m \in \mathcal{Q}$$

That,

$$\sum_{j\in W}\sum_{m\in Q}(\sum_{k\in K}(h_{j,m,k} - h_{j,m-1,k})\sum_{m\in Q} Z^{1}_{j,m-1,k})y_{j,m}\ ,$$

is caused nonlinearity in model. If we try to re-write this phrase in such way that non-linearity eliminated we have linear model.

For this purpose, we re-define $\Delta f_{j,m}$ and we add aggregate product variable which is denoted by $(AZ_{j,m,k})$. This variable shows sum of each product that is stored up to maximum opened capacity level. For example, if a warehouse $j \in \mathcal{W}$ opened with third capacity level,$AZ_{j,3,k}$ shows sum of product $k \in K$ that is stored in first, second, and third level capacity of warehouse$j \in \mathcal{W}$.

We use new definition for $\Delta f_{j,m}$ as follow:

$$\Delta f_{j,m} = f_{j,m} - f_{j,m-1}\ \forall j \in \mathcal{W}, m \in \mathcal{Q} \tag{3-2-2}$$

So, we can substitute old objective function terms by new ones as described above:

$$\sum_{j\in W}\sum_{m\in Q}\sum_{k\in K} h_{j,m,k}AZ_{j,m,k} + \sum_{j\in W}\sum_{m\in Q}\Delta f_{j,m}y_{j,m} \quad (3\text{-}2\text{-}3)$$

At this time we need to add some constraints that let $AZ_{j,m,k}$ stores sum of product $k \in K$ up to maximum opened capacity$m \in Q$.

In reference to Defersha and Chen (2008) we can use following constraints for this purpose:

$$AZ_{j,m,k} \geq \sum_{m=1}^{m} Z_{j,m,k}^{1} + Mt_{j,m} - M, \forall j \in W, m \in Q, k \in K \quad (3\text{-}2\text{-}4)$$

$$AZ_{j,m,k} \leq \sum_{m=1}^{m} Z_{j,m,k}^{1}, \forall j \in W, m \in Q, k \in K \quad (3\text{-}2\text{-}5)$$

$$AZ_{j,m,k} \leq Mt_{j,m}, \forall j \in W, m \in Q, k \in K \quad (3\text{-}2\text{-}6)$$

Constraint (3-2-4) implies if new binary variable $t_{j,m}$get value 1 then constraint (3-2-4) will be:

$$AZ_{j,m,k} \geq \sum_{m=1}^{m} Z_{j,m,k}^{1}, \forall j \in \mathcal{W}, m \in Q, k \in K \qquad (3\text{-}2\text{-}7)$$

Then Constraint (3-2-4) and (3-2-5) will turn to equality constraint:

$$AZ_{j,m,k} = \sum_{m=1}^{m} Z_{j,m,k}^{1}, \forall j \in \mathcal{W}, m \in Q, k \in K \qquad (3\text{-}2\text{-}8)$$

So, $AZ_{j,m,k}$ will be sum of product $k \in K$ in all level capacity of warehouse $j \in \mathcal{W}$ up to capacity $m \in Q$.But if binary variable$t_{j,m}$ get value zero $AZ_{j,m,k}$ will be zero by Constraint (3-2-6).

At this step, following constraints let $t_{j,m}$ equals to maximum opened level that means only maximum capacity level (m) of warehouse ($j \in \mathcal{W}$) get value 1 and other $t_{j,m}$s get zero.

$$t_{j,m} \leq y_{j,m} \forall j \in \mathcal{W}, m \in Q \qquad (3\text{-}2\text{-}9)$$

$$\sum_{m=0}^{m=q} t_{j,m} = 1 \; \forall j \in \mathcal{W} \qquad (3\text{-}2\text{-}10)$$

$$t_{j,m} \geq t_{j,m-1} + My_{j,m} - M \forall j \in \mathcal{W}, m \in Q \qquad (3\text{-}2\text{-}11)$$

Constraint (3-2-9) guarantees that if $y_{j,m} = 0$, auxiliary $t_{j,m}$ cannot be 1. Constraint (3-2-10) assures that only one capacity of related warehouse can get value 1.For instance, if a warehouse $j \in \mathcal{W}$ is built with second capacity level then we will have$t_{j,2} = 1$. In case

that a warehouse $j \in \mathcal{W}$ should not be built then we have $t_{j,0} = 1$ that satisfy constraint (3-2-10). Finally, constraint (3-2-11) causes warehouse $j \in \mathcal{W}$ be operational with maximum available capacity level.

Using above modification in non-linear model, we propose the following linear model for our problem.

Minimize:

$$\sum_{j\in\mathcal{W}} \sum_{m\in\mathcal{Q}} \sum_{k\in K} h_{j,m,k} AZ_{j,m,k} + \sum_{j\in\mathcal{W}} \sum_{m\in\mathcal{Q}} \Delta f_{j,m} y_{j,m} + \sum_{j\in\mathcal{W}} \sum_{i\in\mathcal{C}} \sum_{k\in K} d_{i,k} x_{i,j} c_{i,j} + \sum_{j\in\mathcal{W}} \sum_{m\in\mathcal{Q}} Z^2_{j,k} \hat{c}_{j,k} \quad (3\text{-}2\text{-}12)$$

$$\sum_{j\in\mathcal{W}} x_{i,j} = 1, \forall i \in \mathcal{C} \quad (3\text{-}2\text{-}13)$$

$$x_{i,j} \leq y_{j,1}, \forall i \in \mathcal{C}, j \in \mathcal{W} \quad (3\text{-}2\text{-}14)$$

$$\sum_{k\in K} b_k Z^1_{j,m,k} \leq \Delta S_{j,m} y_{j,m}, \forall j \in \mathcal{W}, m \in \mathcal{Q} \quad (3\text{-}2\text{-}15)$$

$$\sum_{k\in K} b_k Z^1_{j,m-1,k} \geq \Delta S_{j,m-1} y_{j,m}, \forall j \in \mathcal{W}, m \in \mathcal{Q} \quad (3\text{-}2\text{-}16)$$

$$Z^1_{j,k} = \sum_{m\in\mathcal{Q}} Z^1_{j,m,k}, \forall j \in \mathcal{W}, k \in K \quad (3\text{-}2\text{-}17)$$

$$Z^1_{j,k} + Z^2_{j,k} = \sum_{i\in\mathcal{C}} d_{i,k} x_{i,j}, \forall j \in \mathcal{W}, k \in K \quad (3\text{-}2\text{-}18)$$

$$AZ_{j,m,k} \geq \sum_{m=1}^{m} Z_{j,m,k}^{1} + Mt_{j,m} - M, \forall j \in \mathcal{W}, m \in Q, k \in K \quad (3\text{-}2\text{-}19)$$

$$AZ_{j,m,k} \leq \sum_{m=1}^{m} Z_{j,m,k}^{1}, \forall j \in \mathcal{W}, m \in Q, k \in K \quad (3\text{-}2\text{-}20)$$

$$AZ_{j,m,k} \leq Mt_{j,m}, \forall j \in \mathcal{W}, m \in Q, k \in K \quad (3\text{-}2\text{-}21)$$

$$t_{j,m} \leq y_{j,m}, \forall j \in \mathcal{W}, m \in Q \quad (3\text{-}2\text{-}22)$$

$$\sum_{m=0}^{m=q} t_{j,m} = 1 \; \forall j \in \mathcal{W} \quad (3\text{-}2\text{-}23)$$

$$t_{j,m} \geq t_{j,m-1} + My_{j,m} - M, \forall j \in \mathcal{W}, m \in Q \quad (3\text{-}2\text{-}24)$$

$$x_{i,j} \in \{0,1\}, \forall i \in \mathcal{C}, j \in \mathcal{W} \quad (3\text{-}2\text{-}25)$$

$$y_{j,m} \in \{0,1\}, \forall j \in \mathcal{W}, m \in Q, \quad (3\text{-}2\text{-}26)$$

$$t_{j,m} \in \{0,1\}, \forall j \in \mathcal{W}, m \in Q, \quad (3\text{-}2\text{-}27)$$

$$Z_{j,k}^{1}, Z_{j,k}^{2} \in \mathbb{Z}^{+}, \forall j \in \mathcal{W}, k \in K \quad (3\text{-}2\text{-}28)$$

$$Z^1_{j,m,k} \in \mathbb{Z}^+, \forall j \in \mathcal{W}, m \in Q, k \in K \quad (3\text{-}2\text{-}29)$$

$$AZ_{j,m,k} \in \mathbb{Z}^+, \forall j \in \mathcal{W}, m \in Q, k \in K \quad (3\text{-}2\text{-}30)$$

In new model constraint (3-2-13) assigns each customer to only one warehouse, and constraint (3-2-14) implies that customers are assigned only to warehouses that already exist. Constraint sets (3-2-15) and (3-2-16) ensure that the level of storage corresponds to the correct level on the staircase cost function for each warehouse. Number (3-2-17) keeps amount of product $k \in K$ in all level $m \in Q$ for warehouse $j \in \mathcal{W}$ .Constraint (3-2-18) ensures that the demand for all the customers will be met, either by the warehouse or the production facility. Constraints (3-2-19) to (3-2-22) open warehouses with maximum available capacity and constraints (3-2-23) and (3-2-24) assign correct decision binary variable for appropriate capacity level of warehouse$j \in \mathcal{W}$.

3-3 Summary

In this chapter we first presented a model to cover facility location staircase cost function then we linearized it. We transformed the nonlinear model to a linear model by eliminating the nonlinearity term in the objective function. As it is clear in the linear model objective function, the first part assigns appropriate holding cost to warehouse $j \in \mathcal{W}$ products and the second part assign appropriate fix cost value to warehouse $j \in \mathcal{W}$ level $m \in Q$. So there is no non-linear term in objective function. The rest of objective terms are equal to non-linear model.

In the next chapter, we will show the computational result of the non-linear model and linear model by Lingo 8.0 and Lingo 10 software and we compare these two sets of results together.

Chapter 4: Computational Experiments with Lingo 8.0 & Lingo 10.

4-1 Non Linear Model analytical examples

We used Lingo 8.0 for coding and testing the model. We generated random problems with different number of warehouses, capacities, customers, commodities, demands, and different shipping cost.

These random problems have been evaluated on a 1.6 GHz Pentium PC with 1024 MB RAM. We find optimal solution for small problems at early stage of problem running. For example, we obtain optimal solution for the first problem Table 4-1 (see the appendix 1) in 5 second.

This optimal solution has been shown in Table 4-1.Warehouse 1 is set to its maximum capacity which is 210000 units and warehouse 2 is set to its maximum capacity which is 170000 units. Customer 2 and Customer 3 are assigned to warehouse 1 and Customer 2 is assigned to warehouse 2. All demands are being fully satisfied by warehouses ($Z^1_{j,k}$) and no order is being released by any warehouse to satisfy customer demands. ($Z^2_{j,k} = 0$).

Warehouse	*Selected warehouse Capacity*	$Z^1_{j,k}$	$Z^2_{j,k}$	*Assigned customer*
w_1	$m_3 = 210000$	$Z^1_{1,1} = 9000$ $Z^1_{1,2} = 6000$	$Z^2_{1,1} = 0$ $Z^2_{1,2} = 0$	C_2 C_3
w_2	$m_3 = 170000$	$Z^1_{2,1} = 6000$ $Z^1_{2,2} = 4500$	$Z^2_{2,1} = 0$ $Z^2_{2,2} = 0$	C_1

Table 4-1 *Lingo solution for appendix 1 problem*

However, when we increase the number of warehouses, capacities, customers and products gradually we rarely get a feasible solution for two reasons. First of all, the non-linearity term causes each solution fall in a local optimum. Secondly, as the size of the problem increases gradually, we rarely reach a feasible solution by the end of a pre-specified time (maximum 3 hours).

Problem	*W*	*m*	*C*	*K*	*Iterations*	*Processing time*	*Objective value*	*State*
1	2	3	3	2	16509	00:00:05	163800	Optimal
2	2	3	5	3	53197	03:00:00	3.39675e+006	Feasible
3	3	4	6	5	39444608	03:00:00	3.18685e+007	Feasible
4	3	5	8	5	40083603	03:00:03	4.93807e+007	Feasible
5	4	5	12	6	189618630	03:00:01	504648	Feasible
6	4	6	15	8	6744323	03:00:01	3.2792e+006	Feasible
7	5	5	12	8	11072710	03:33:14	2.14585e+008	Feasible
8	5	5	15	8	6097317	03:38:01	4.1133e+006	Feasible
9	6	4	15	6	19628275	03:00:01	*N/A*	Unknown
10	6	6	20	10	4174448	03:00:01	*N/A*	Unknown

Table 4-2 *different size Non linear sample problems with Lingo 8.0*

Table 4-2 confirms that we are not able to obtain feasible solution when the size of problem increases gradually. In section 4-2, we try to solve same Table 4-2 problems with linear model. We will compare the differences at the end.

4-2 Linear Model analytical examples

We solved problems 1 to 10 in Table 4-2 for linear model experiments by Lingo 10 software. The result of these experiments is shown in Table 4-3. The results obtained suggest that our problem is NP hard and it cannot be solved by branch and bound method in reasonable time.

Problem	*W*	*m*	*C*	*K*	*Iterations*	*Processing time*	*Objective value*	*State*
1	2	3	3	2	172	00:00:05	163800	Optimal
2	2	3	5	3	29694	00:00:08	3.39596e+06	Optimal
3	3	4	6	5	43371835	00:01:04	1.64991e+07	Optimal
4	3	5	8	5	12387	00:00:06	2.84864e+07	Optimal
5	4	5	12	6	687169	00:01:04	500259	Optimal
6	4	6	15	8	502615	00:01:27	3.05511e+06	Optimal
7	5	5	12	8	4303	00:00:12	2.06428e+08	Optimal
8	5	5	15	8	63830616	01:38:01	1.12045e+06	Optimal
9	6	4	15	6	4009909	03:00:01	1.82949e+07	Feasible
10	6	6	20	10	18453071	03:00:01	8.57393e+06	Feasible

Table 4-3 *different size linear sample problems with Lingo 10.0*

As it shown in Table 4-2, for problems 1 to 8 we obtain the optimal solution. As the size of problems increase, solution obtained stay at a feasible state and for larger problems there would be no solution at all. Linear model facilitates the problem solving by providing better solution as it shown in Table 4-3. More problems get optimal solution but when the size of problem increases the chance of getting optimal solution decreases as well. For this reason, we used a Meta heuristic algorithm approach for solving larger problems.

4-3 Summary

In this chapter we solved some problems for both nonlinear and linear models with Lingo 8 and Lingo 10. As it shown in Table 4-2 the results obtained by the linear model when the size of problems get large are reasonably better than the results we obtain from the non-linear model. Another important issue is about processing time of problems when the size of problems increases. We acquire better solution in a shorter time frame with the linear model than the nonlinear model.

However, in larger size problems linear model barely reaches a feasible solution. For covering large size problems, we are going to develop a Meta heuristic for our problem. Among different kind of Meta heuristic methods such as simulated annealing, genetic algorithm and Tabu Search, we choose Tabu Search because it uses flexible memory and responsive exploration in guiding the solution process to move from one trial solution to another. By responsive exploration, it determines a search direction in the solution space based on the properties of the current solution and the search history and converges to optimal or near optimal solution at the end.

In next Chapter, we will use a Meta heuristic method the Tabu Search, in an attempt to find better feasible solutions for larger problems within an acceptable processing time. In next chapter we will develop Tabu Search method for our model. Then, we simulate the model with Tabu Search algorithm to compare the results with Lingo 8.0 and Lingo 10

solutions. For having better comparison we will compare the Tabu Search final solution with both non-linear and linear model results.

Chapter 5: TABU SEARCH FRAME WORK

The proposed model is considered to be an NP-Hard problem with reference to the model with a multi commodity (k), single holding cost ($h_{j,k}$) for all $j \in W$ and one potential available capacity for each warehouse is NP-Hard (Bektas and Bulgak, 2008). Thus, we develop a Meta Heuristic method for solving our model at this stage to reach better solutions. Tabu Search (TS) is a popular optimization technique used in a variety of optimization problems (Glover and Laguna 1997). The beneficial advantage of Tabu Search is escaping from local optimality especially in combinatorial problems where for reaching this goal, a move that leads to the next considered solution can be accepted even if the cost of this solution is worse than the current solution. (Ah Kioon *et al.* 2008)

As the literature defines, Tabu Search generalizes the basic local search procedure which is terminated when an improved solution in the neighborhood of the current solution cannot be found. Precisely, Fred Glover (1997) proposed new approach, which he called Tabu Search, to allow local search methods to overcome local optima. The principle of Tabu Search is to pursue a local search whenever it encounters a local optimum by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called Tabu lists that record the recent history of the search. Tabu lists containing attributes can be more effective for some domains, although they raise a new problem. When a single attribute is marked as Tabu, this typically results in more than one solution being Tabu. Some of these solutions that must now be avoided could be of excellent quality and might not have been visited. To mitigate this problem, "aspiration criteria" are introduced: these override a solution's Tabu state, thereby

including the otherwise-excluded solution in the allowed set. A commonly used aspiration criterion is to allow solutions which are better than the currently-known best solution (Glover 1997).

In this manner, we set the best non-improving solution as our current solution when it is not in Tabu list or it satisfies the aspiration criterion. With Tabu Search we can escape from the local optima and explore the larger subset of solution space. Therefore for advancing our procedure, we must specify an initial solution that is chosen from a set of feasible solutions with the best objective value, the way that Tabu moves, the time that Tabu lasts, and the aspiration criteria which dictates how to overrule a Tabu.

Although, we can start with any solution in feasible region but the best way is founding a good initial solution which converges to best solution at lowest computation time.
For finding an initial solution we got an idea from Uster and Keskin (2007). We employ following steps to have a good initial solution:

- Initialization
- Diversification
- Duplication
- Feasibility
- Customers assignment and Commodity placement
- Selecting Best Solution

For specific number of t_{max} we generate two kind of initial solutions (primary and reverse initial solution). Then we check the feasibility for these generated solutions. After feasibility check, we select the solution which returns lowest cost and store it as best initial solution. This method will continue till $t > t_{max}$ is satisfied.

At end, we will announce best initial solution which has been found as our permanent initial solution. Figure 5-1 illustrates the flowchart of initial solution phase.

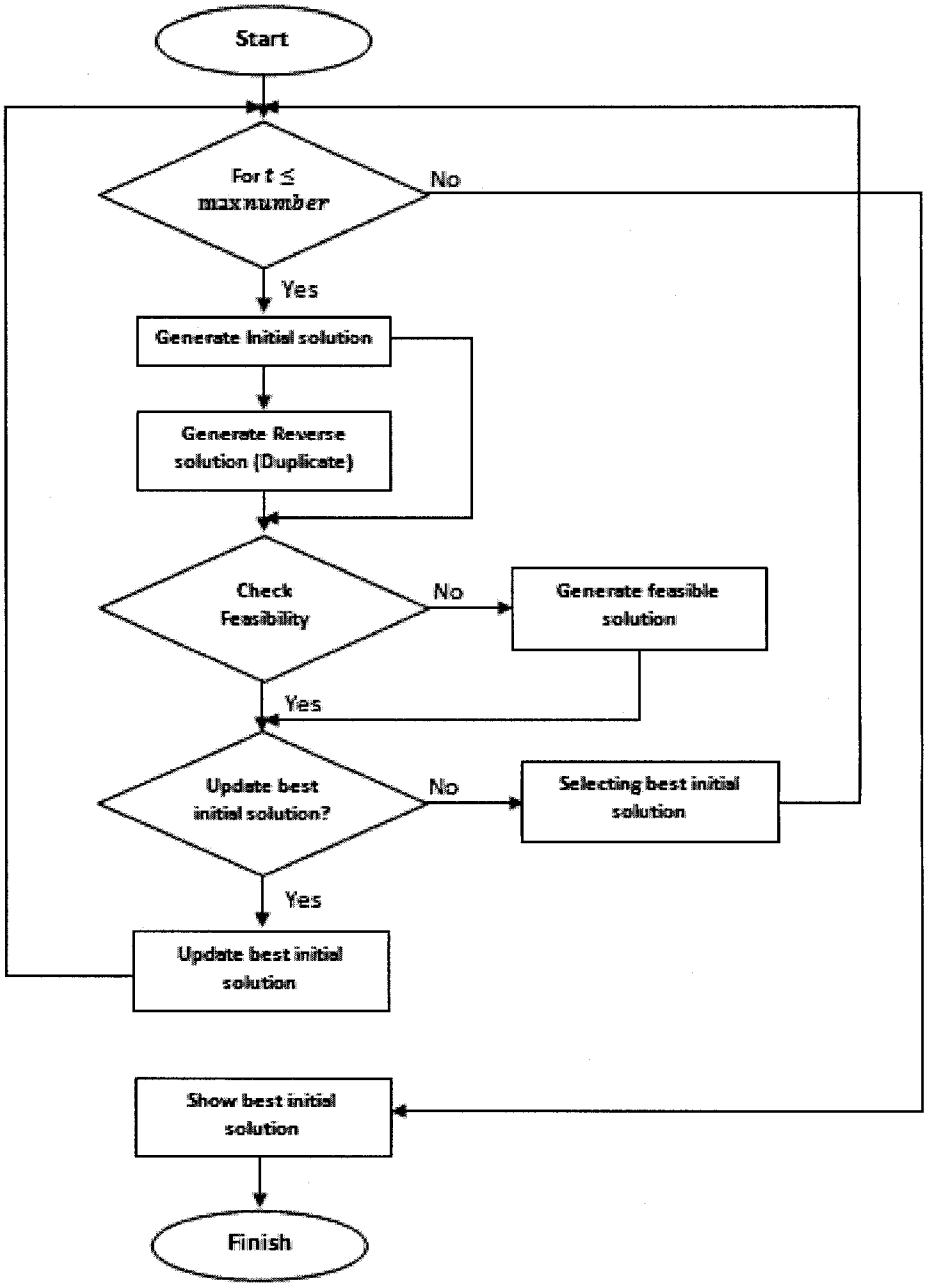


Figure 5-1 *Initial solution flowchart*

Following section's explanation makes initial solution phase comprehensible.

5-1 Initialization

Initial solution is a binary vector y that consists of the warehouses and the proper capacities. For instance, suppose we have 3 possible warehouse locations that are able to create up to four different capacities. Figure 5-2 shows an example of an initial solution.

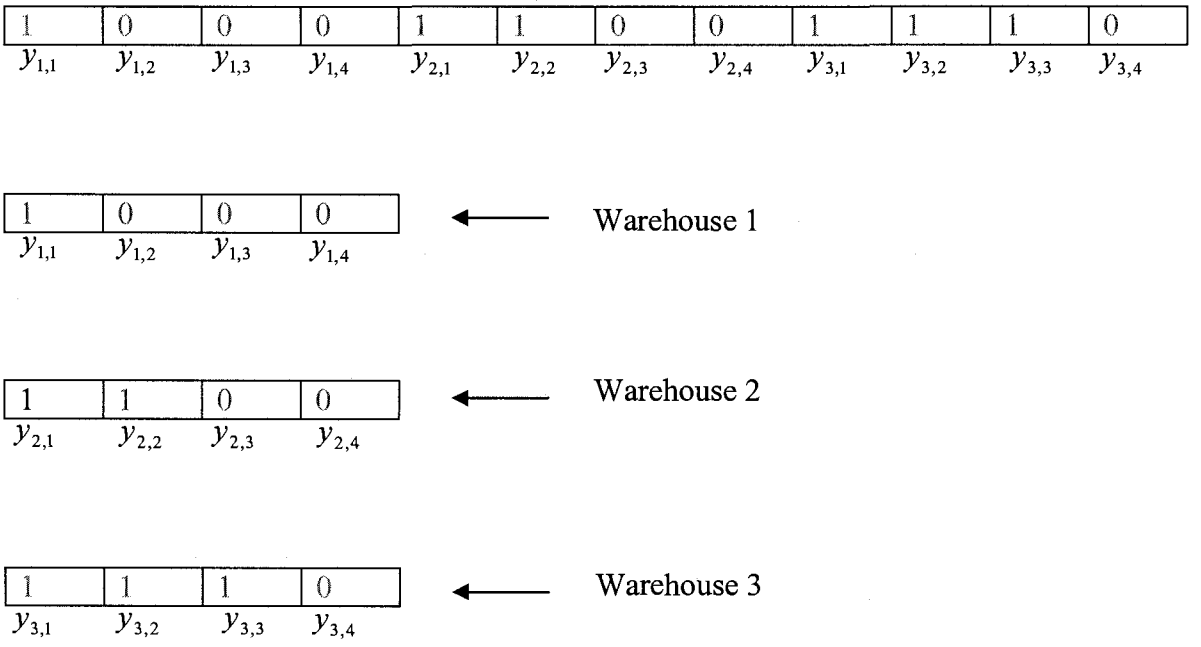


Figure 5-2 *an initial solution example*

As it shown in the figure 5-2, the first warehouse is set to its first capacity level, the second warehouse is opened with second capacity level and third warehouse uses its third capacity level as initial solution.

For generating new solutions we randomly generate initial solution with $j \times m$ elements with binary value (0, 1) in diversification step. We repeat this procedure for t times to cover more area of our feasible region.

5-2 Diversification

Let $y_{j,m}$ be the elements of an initial binary solution (Y) as it illustrated in figure 5-3.

0	0	0	0	0	0	0	0	0	0	0	0
$y_{1,1}$	$y_{1,2}$	$y_{1,3}$	$y_{1,4}$	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$	$y_{2,4}$	$y_{3,1}$	$y_{3,2}$	$y_{3,3}$	$y_{3,4}$

Figure 5-3 *binary solution for warehouse capacities*

The initial solution vector (Y) denotes an n-vector ($n = j \times m$) which each component of Y receives value 0 or 1, we randomly set these elements to zero and one. Algorithm 1 shows how we are able to generate these mentioned solutions. Algorithm 5-1 shows the way of generating these initial solutions:

Algorithm 5-1 generating Initial solutions

1: *for* $t=1$ *to* t_{max}

2: *for* $j=1$ *to* W_{max}

3: *for* $m=1$ *to* Q_{max}

4: $y_{[j][m]} \leftarrow Random(0,1)$

5: *Next for*

6: *Next for*

For example, above algorithm generates below solution (figure 5-4):

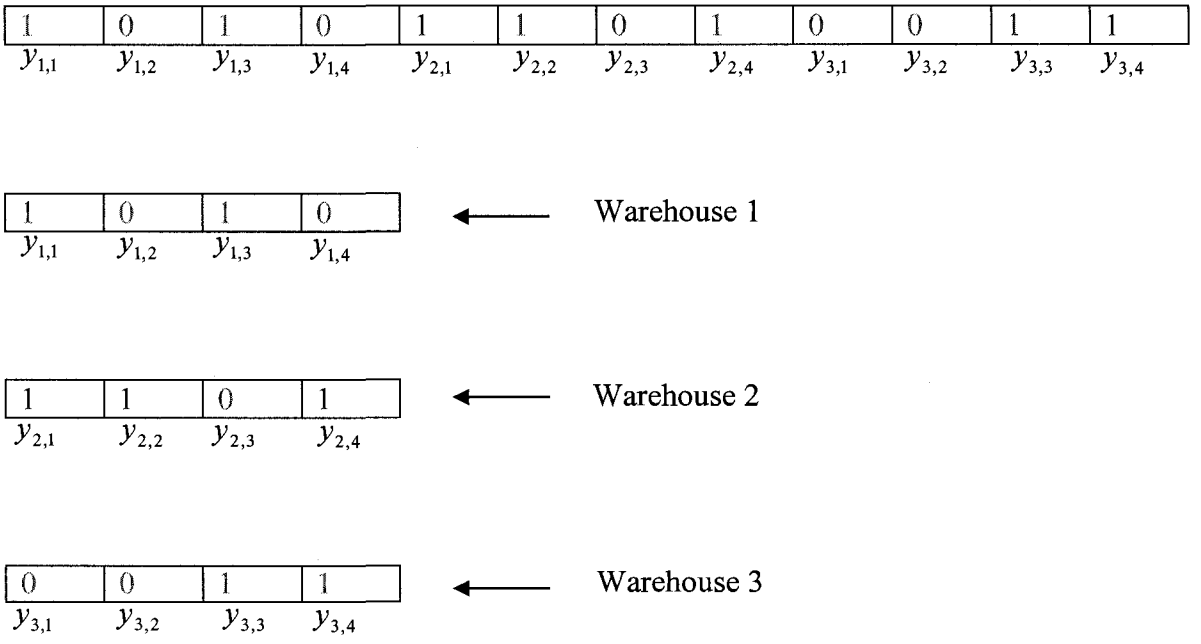


Figure 5-4 *Random Initial solution for warehouse capacities*

5-3 Duplication (Reverse solution)

In reference to Glove 1997 we used Algorithm 5-2 for generating another solution by inverting each element of generated solution in algorithm 1. Algorithm 5-2 shows the way of creating another solution from generated solutions:

Algorithm 5-2 generating Reverse solutions

1: *for* $t=1$ *to* $t_{\max}$

2: *for* $j=1$ *to* $W_{\max}$

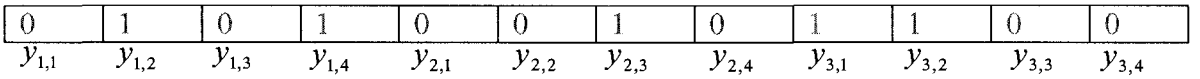
3: *for* $m=1$ *to* $Q_{\max}$

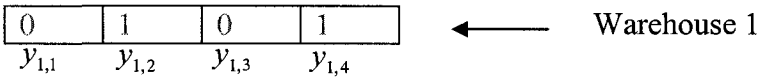
4: $y'_{[j][m]} = 1 - y_{[j][m]}$

5: *Next for*

6: *Next for*

Figure 5-5 shows reversed solution of figure 5-4 case which is created by algorithm 5-2.





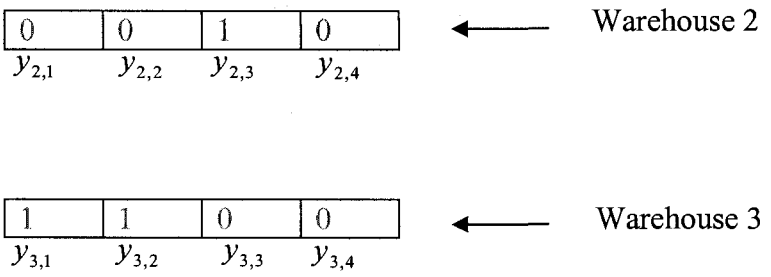


Figure 5-5 *Reversed Initial solution for warehouse capacities*

5-4 Feasibility

The trial solutions in the population generated by the diversification and duplication steps would be infeasible in most cases due to violation of two constraints (3) and (4). For instance, if the capacity of any warehouse j at level m is set to zero, then a higher capacity level, e.g., $m+1$ cannot be set to 1.

Therefore, we change these non feasible solutions into feasible one using algorithm (3) given below. In the case of infeasibility ($y_{j,m} < y_{j,m+1}$), the algorithm changes the value of $y_{j,m}$ to one or the value of $y_{j,m+1}$ to zero by generating a random variable (r). Then algorithm returns to the initial element ($y_{j,1}$). This procedure repeats until a feasible solution generated.

Algorithm 3 generating feasible solution

1: *for* $j=1$ *to* $W_{\max}$

2: $m \leftarrow 0$

3: *While* $m < Q_{\max} - 1$

4: *if* $y_{j,m} < y_{j,m+1}$

5: $r \leftarrow 0 \le Rand \le 1$

6: *if* $r < .5$

7: $y_{j,m} \leftarrow 1$

8: *else*

9: $y_{j,m+1} \leftarrow 0$

10: $m \leftarrow 0$

11: *end if*

12: else

13: $m \leftarrow$ m+1

14: *end while*

15: *Next for*

5-5 Assignment of Customers and Commodity placement:

After finding a feasible solution (Y^I), we will assign customers to available warehouses base on lowest shipping cost. Priority of assigning goes to customer with most product demand. For handling this issue, we apply a similar heuristic that is used by Bektas and Bulgak (2008) for our problem. We define customer assignment heuristic steps:

We will consider assignment decision by following formula:

$$V(i) \in \arg\min_{j \in Y^I} \left\{ \sum_{k \in K} d_{i,k} c_{i,j} + \hat{c}_{j,k} \left(\sum_{k \in K} d_{i,k} \times b_k - S_{j,m} \right)^+ \right\},$$

Where $\left(\sum_{k \in K} d_{i,k} \times b_k - S_{j,m} \right)^+ = \max\left(0, \left(\sum_{k \in K} d_{i,k} \times b_k - S_{j,m} \right)^+ \right)$, $V(i)$ denotes the set of warehouses that customer i can be assigned to, and $S_{j,m}$ would be the highest level of each warehouse that already is on hand. Summing up, each customer i is assigned to warehouse $j \in Y^I$ such that the total cost of shipping between customer and warehouse ($\sum_{k \in K} d_{i,k} c_{i,j}$), and the distribution cost of excess demand if exist ($\hat{c}_{j,k} \left(\sum_{k \in K} d_{i,k} \times b_k - S_{j,m} \right)^+$) minimized. Obviously, the capacity of warehouse $j \in Y^I$ is decreased by the amount $\left(\sum_{k \in K} d_{i,k} \times b_k \right)$ whenever customer i is assigned to it. The assignment decision is terminated after each customer has been assigned to a warehouse. The resulting solution is shown by $\bar{x}_{i,j}$.

After each customer assigned to warehouse successfully, we try to determine how much of commodities should store at each warehouse $j \in Y^I$, precisely we wish to determine values of the $Z^1_{j,k}$ as well as amount of the excess commodities which each warehouse should order directly to plant ($Z^2_{j,k}$). It is possible that at previous step, not all warehouses $j \in Y^I$ have customers assigned to them. If this happen, we will have new feasible solution and we change the initial feasible solution to new one as follows:

$$Y^I \leftarrow Y^I = \{j, m \in Y^I \mid \exists i \in C \ s.t \ V(i) = j\}$$

In other word, new Y^I is the set of open warehouses with each element having at least one assigned customer. For each $j \in Y^I$ the commodity placement can be represented by followed integer programming formulation:

$$(CP_j) \quad Minimize \sum_{k \in K} h_{j,m,k} Z^1_{j,k} + \sum_{k \in K} \hat{c}_{j,k} Z^2_{j,k} \qquad 5\text{-}5\text{-}1$$

Subject to:

$$\sum_{k \in K} b_k Z^1_{j,k} \le S_{j,m} \qquad 5\text{-}5\text{-}2$$

$$Z^1_{j,k} + Z^2_{j,k} = \sum_{i \in C: V(i) = j} d_{i,k} \quad \forall k \in K \qquad 5\text{-}5\text{-}3$$

$$Z^1_{j,k} \ge 0, Z^2_{j,k} \ge 0, \forall j \in Y^I, k \in K \qquad 5\text{-}5\text{-}4$$

The values for $Z^1_{j,k}$ and $Z^2_{j,k}$ are calculated only for warehouses $j \in Y^I$, also the right hand side of constraint 2-3 is defined only for customers that are assigned to these warehouses.

For solving (CP_j) we use similar algorithm which has used by Bulgak and Bektas, 2008.

Algorithm 4 Heuristic to solve (CP_j)

1: Sort commodities in an increasing order of $\frac{h_{j,m,k}}{\hat{c}_{j,k}}$. Let $\{k_1, k_2, ..., k_{|K|}\}$ denote this ordering

2: $pcap = S_{j,m}$

3: $t = 1$

4: while $pcap \geq 1$ *do*

5: $\overline{Z}^1_{j,k_t} = \min\{\sum_{i \in c} d_{i,k_t} \overline{x}_{i,j}, \left[\frac{pcap}{b_{k_t}}\right]\}$

6: $pcap = pcap - \overline{Z}^1_{j,k_t} \times b_{k_t}$

7: $t \leftarrow t + 1$

8: *end while*

9: for all $k \in K$ do

10: $\overline{Z}^2_{j,k_t} = \sum_{i \in C} d_{i,k_t} \overline{x}_{i,j} - \overline{Z}^1_{j,k_t}$

11: *end for*

Commodities are sort by their $\frac{h_{j,m,k}}{\hat{c}_{j,k}}$ value. After that, the commodities by this given order are placed in the warehouse up to capacity $S_{j,m}$. As soon as the capacity $S_{j,m}$ is met, all reminded commodities are supplied from the plant directly which is correspond to the values of the $\overline{Z}^2_{j,k_t}$. Based on these algorithms, final objective value for each feasible solution is derived from following formula.

$$\phi_f = \sum_{j \in Y^I} f_{j,m} y_{j,m} + \sum_{j \in Y^I} \sum_{i \in C} \sum_{\substack{k \in K \\ V(i)=j}} d_{i,k} c_{i,j} + (CP_j)$$

As it clear we only need to find a feasible solution (Y^I), afterward we calculate the other variables amount by two mentioned algorithms and we calculate the objective value for each instance feasible solution.

5-6 Selecting best initial solution

After generating feasible solutions, the problem is solved for these solutions (Initial and Reverse) by the heuristic methods described above. Afterwards, we select one solution among these solutions which returns lower objective function value. We set this solution as the best solution. According to the procedure above, for t times and we compare each best solution in every period. If the best solution in the next iteration is better than the previous one, we update the best solution, otherwise while loop continues till reach to t number.

5-7 Summary

In initial phase we randomly opened and closed potential warehouses for determined location with different capacities. Then, we assigned customers to this available warehouse set and allocated customer demands to each warehouse by two different heuristic algorithms. Furthermore, objective value for these solutions are compared to each and best objective value and solution is selected as best initial solution.

We coded above algorithm in visual C++ 6.0 which passes best initial solution to Tabu Search program. In next chapter, we will develop Tabu Search for our model, this Meta heuristic method works on potential warehouse locations and warehouse capacities.

Chapter 6: Tabu Search Implementation

In this chapter we implement the Tabu Search algorithm as discussed in chapter 5 for the proposed model. We explained how Tabu Search works and how it is able to find solution by calculating the $x_{i,j}, Z^1_{j,k}$ and $Z^2_{j,k}$. After implementation we will compare the Tabu Search results with both the non-linear and the linear model results for the same set of problems. In this way we are able to show how close Tabu result is to optimal or near optimal solutions.

The Tabu Search scheme for our problem is described as follows: we start with initial solution (Y^I) which we obtained in pervious section and we keep its objective function value by calculating the $x_{i,j}, Z^1_{j,k}$ and $Z^2_{j,k}$. We also set this objective value as Best Answer ($\phi(Y^{Best})$) and Current Answer ($\phi(Y^C)$).Then, we generate a certain sets of neighborhoods first by opening new warehouses with different capacity levels and second by increasing or decreasing capacity level of our current solution. If the best of these moves is not Tabu and is better than overall solutions or, the best is Tabu but satisfies the aspiration criterion we pick that move and consider it as best solution (Y^{Best});otherwise, we pick the best move that is not Tabu and put as our current solution (Y^c). Figure 6-1 illustrates the flowchart of the above explanation.

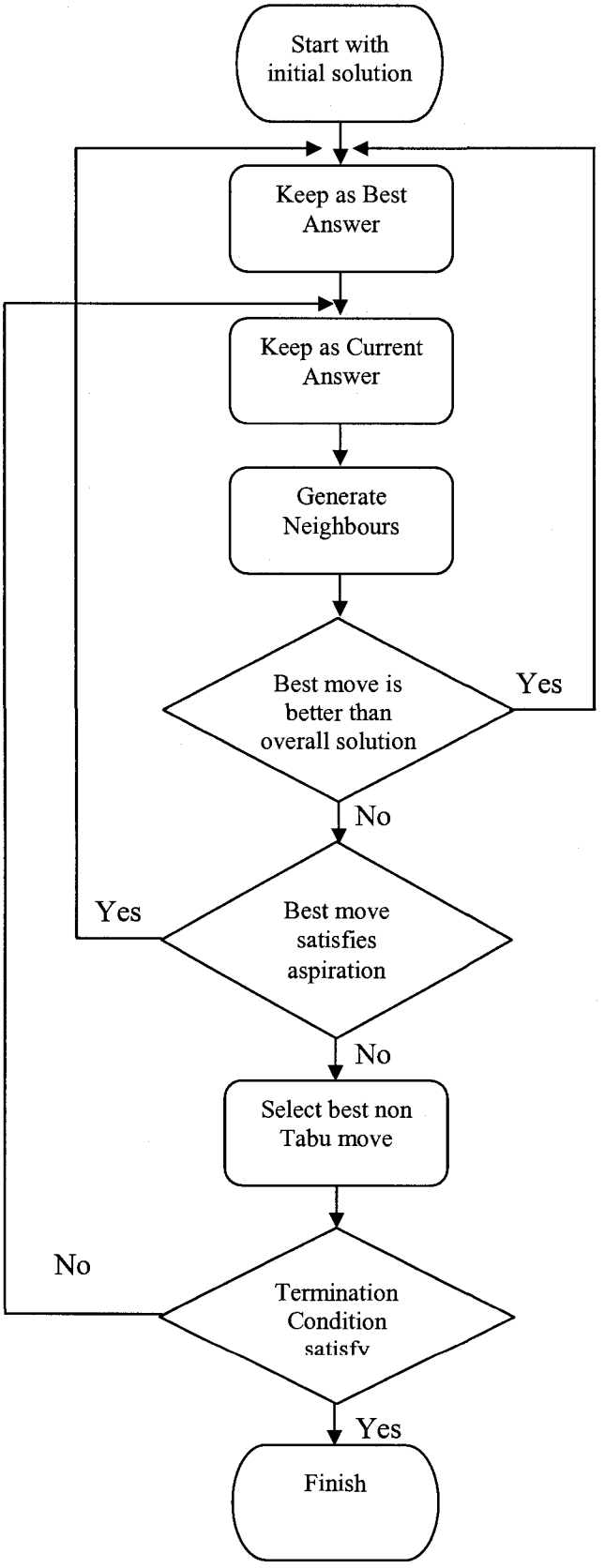


Figure 6-1 *Tabu Search flowchart*

In reference to the scheme, first, we try to generate some neighborhood from the initial solution (Current Solution) through closing or opening warehouse capacity.

Let A= $\{ j \in W, m \in Q : y_{j,m}=1 \}$ and $\psi = W / A$, in this case we are able to create some neighborhood by Add and Drop move. Add move consist of moves where a single component ($y_{j,m}$) is opened when it's already close and Drop move consist of moves where a single component ($y_{j,m}$) is closed when it's already open. Second, we generate some neighborhoods with different warehouse capacity levels.

Figure (6-2) shows generating new neighborhood from current solution. We generate new solutions by opening/closing capacities from current solution. Suppose we have three potential warehouse locations that can be set up to five capacities, our current solution shows first warehouse is opened with its third capacity, second warehouse is set to its second capacity and third warehouse is set to its first capacity.

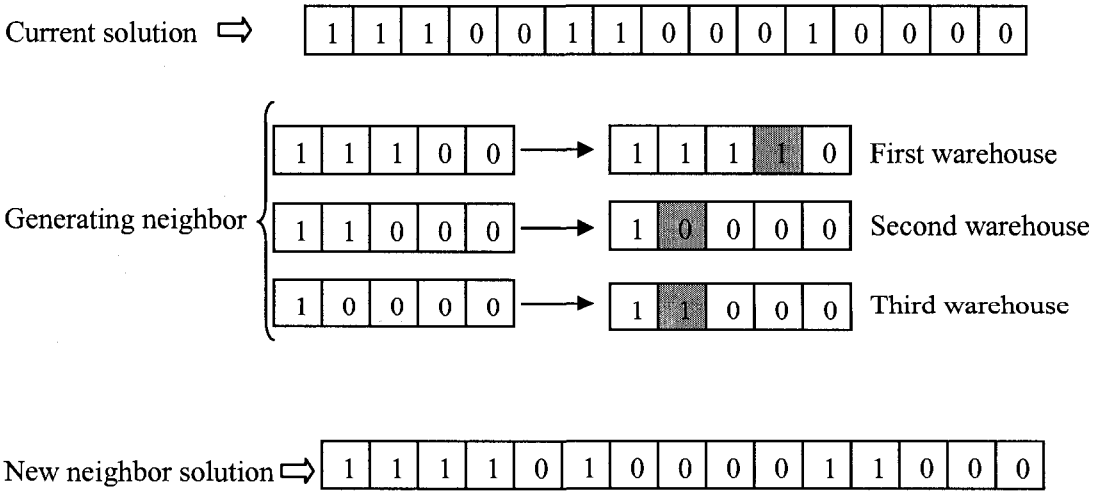


Figure 6-2 *Generating neighbor by Add and Drop Move*

Consequently, we are able to generate different neighborhoods from current solution through dropping or adding capacity to current warehouse level capacity.

Second alternative, we create new neighborhoods with different size capacity level by setting each warehouse to new capacity level. Figure (6-3) shows possibility of generating new warehouse capacity. Suppose we have three potential warehouse locations that can be set up to five capacity levels, new neighborhoods can be generated with setting each warehouse to new different capacity level. Figure (6-3) shows these possible moves.

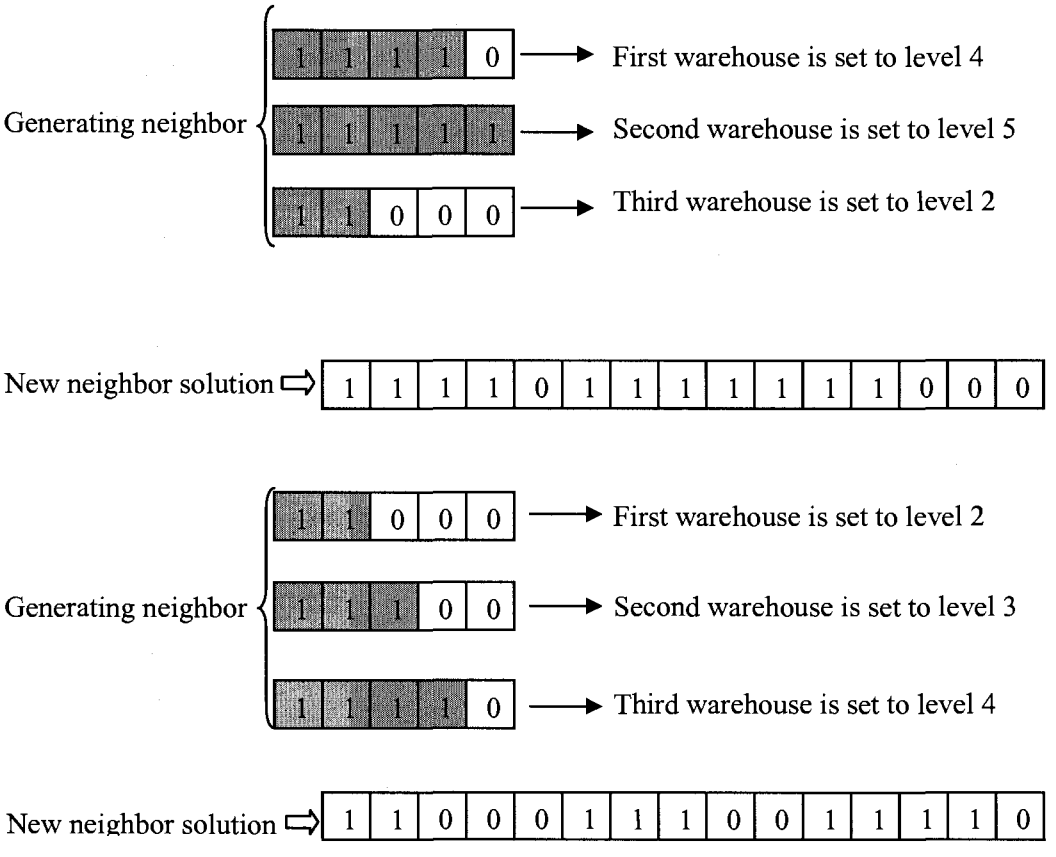


Figure 6-3 *Generating neighbor by creating new level capacity*

To prevent cycling and re-visiting previously visited solutions, Tabu move restrictions are employed. In our implementation, we classify a solution obtained by Add / Drop or Generating new capacities, as a Tabu if it corresponds to closing a capacity or open new capacity which was opened or closed in an accepted solution in the course of the procedure.

This Tabu move restriction is solved by employing Tabu tenure which is the number of iterations that an open capacity or closed one of a specific warehouse remains a Tabu. For simplicity we set Tabu tenure for a newly opened capacity or recently closed one for a fixed number of iterations. In our case we define two Tabu lists. First, if new capacity opens in new move we put this capacity in recently added capacity list which cannot drop for a fix number of iterations, second if opened capacity drops in new move we put this capacity in recently dropped list which cannot add for a fix number of iterations.

Therefore, the Tabu Search algorithm uses two tenure ADD and DROP lists where $TADD = (T_{1,1}....T_{j,m})$ and $TDROP = (T_{1,1}....T_{j,m})$.

$T_{j,m}$, in each list shows the recently opened/closed capacity of warehouse j, for example, if $T_{j,m} > 0$ for some $j \in W$ in one of above list then relevant warehouse is Tabu and can not be dropped if it belongs to $TADD$ list or cannot be added if it belongs to $TDROP$

list. Any warehouse capacity with a corresponding $T_{j,m}$ equal to zero in *TADD* and *TDROP* Tabu lists is non-Tabu and can be add/drop.

In all iterations, when a candidate solution results in opening or closing capacity m of warehouse j, relevant $T_{j,m}$ is assigned to the appropriate Tabu tenure and all other positive entries in the Tabu lists are decreased by one.

6-1 Aspiration Criterion:

We define the aspiration criterion as solution involving Tabu move that has better objective value than the best known answer, then the Tabu status is disregarded. Otherwise, if the aspiration criterion is not satisfied, we continue to the next iteration with the best non-Tabu solution.

An aspiration criterion is used to overrule the Tabu restrictions; therefore we can consider the attractive unvisited solutions as well. Although one solution is a Tabu move, but it is accepted as legitimate solution whenever it satisfies the aspiration criterion.

For Tabu algorithm input data, we define the maximum number of iterations, max number of non-improving iterations and the Tabu tenure (Keskin and Üster, 2007). At the beginning, no warehouse capacity is a Tabu; therefore add and drop Tabu lists consist of zeros.

We search the both Add/Drop neighborhood and new warehouse capacity neighborhood of the initial solution (Y^I) per iteration and pick the best solution in the neighborhood (Y^c). Afterward, we check the Tabu status.

If the current solution (Y^c) does not contain a Tabu move, we accept this solution as the new initial solution. We also check if this solution is better from the best overall solution (Y^{Best}) that we have so far.

If it is, we update Y^{Best} and reset the number of non-improving solutions to zero; else we add one to non-improving solutions. We require also updating the Tabu list, so we decrease all positive entries by one and setting the value for newly closed, opened or both to Tabu tenure.

If the current solution contains a Tabu move, essentially the aspiration criterion will be checked. If the aspiration is satisfied, we accept the solution as best overall solution and set non-improving solutions number to zero. We also update the Tabu list as well.

When the aspiration criterion is not satisfied, we pick the best non-Tabu solution as new initial solution. Again, we update the Tabu list and increase the number of non-improving solutions by one. Before moving to the next iteration, we check to see if the number of non-improving solutions is smaller than its maximum or not. If not, we terminate the Tabu Search and report the overall best solution as a result of the research. Otherwise the

procedure continues in this fashion until the preset total number of iterations or a preset number of successive non-improving iterations are met.

6-2 Computational Results:

Appendix 2 shows Tabu Search pseudo code for our capacitated location model. We coded Tabu Search algorithm in visual C++ 6.0 and run same problems of Table 4-2 for comparison between nonlinear model and Tabu results. We set non-improving solutions to 1000 iterations, maximum number of iterations to 2250, and Tabu tenure to number of warehouses (j) multiply in maximum capacity (m).

Problem	*W*	*m*	*C*	*K*	*Tabu Search processing time*	*Tabu Search Objective value*	*Lingo 8.0 Processing time*	*Lingo 8.0 Objective value*
1	2	3	3	2	00:00:02	163800	00:00:05	163800
2	2	3	5	3	00:00:04	3.39595e+06	00:00:08	3.39675e+06
3	3	4	6	5	00:00:10	1.86236e+07	03:00:00	3.18685e+07
4	3	5	8	5	00:00:11	3.09071e+07	03:00:03	4.93807e+07
5	4	5	12	6	00:00:08	504648	03:00:01	504648
6	4	6	15	8	00:00:02	3.0551e+06	03:00:01	3.2792e+06
7	5	5	12	8	00:01:45	2.13883e+08	03:33:14	2.14585e+08
8	5	5	15	8	00:03:05	1.1649e+06	03:38:01	4.1133e+06
9	6	4	15	6	00:00:06	1.70125e+007	03:00:01	*N/A*
10	6	6	20	10	00:00:10	8.56848e+006	03:00:01	*N/A*

Table 6-1 *nonlinear model and Tabu Search result comparison*

As it shown in Table (6-1), Tabu Search result is much better than nonlinear model solutions in all cases. When the number of warehouses, capacities, products and customers are increased Lingo software will not be able to enter to feasible state in non-linear model. Problems 9 and 10 show this phenomenon.

Table (6-2), compares linear model and Tabu Search results for same problems of Table (4-2).

Problem	*W*	*m*	*C*	*K*	*Tabu Search Objective value*	*Lingo 10.0 Processing time*	*Lingo 10.0 Objective value*	*State*
1	2	3	3	2	163800	00:00:05	163800	Optimal
2	2	3	5	3	3.39595e+06	00:00:08	3.39596e+06	Optimal
3	3	4	6	5	1.86236e+07	00:01:04	1.64991e+07	Optimal
4	3	5	8	5	3.09071e+07	00:00:06	2.84864e+07	Optimal
5	4	5	12	6	504648	00:01:04	500259	Optimal
6	4	6	15	8	3.0551e+06	00:01:27	3.0551e+06	Optimal
7	5	5	12	8	2.13883e+08	00:00:12	2.06428e+08	Optimal
8	5	5	15	8	1.1649e+06	01:38:01	1.12045e+06	Optimal
9	6	4	15	6	1.70125e+07	03:00:01	1.82949e+07	Feasible
10	6	6	20	10	8.56848e+06	03:00:01	8.57393e+06	Feasible

Table 6-2 *linear model and Tabu Search result comparison*

In most cases, linear model reaches to optimal solution but when the size of problems get increased Lingo 10.0 stay at feasible state and cannot reach to optimal. A good point in Tabu solutions is, given solutions are much near to optimal solution. In all cases, Tabu

solution is near to optimal solutions, and in larger problems (problem 9 and 10) we acquire better solution than linear model.

At this time, we try to solve larger problems of linear model and Tabu Search for having better comparison between solutions of linear model and Tabu Search. Table (6-3) shows more problems with significantly larger number of warehouses, capacities, customers and products.

For considering the proposed model carefully, we run all problems with different amount of demands, shipping costs, penalty costs, fix open costs and capacity sizes to assure that the problems have been solved over a large range of data.

We set maximum iteration for Tabu Search method to maximum 1500 iteration and maximum non-improvement iteration to 1400 iteration and Tabu tenure to number of warehouses (j) multiply in maximum capacity (m) as before.

Problem	*W*	*m*	*C*	*K*	*Initial solution by Tabu Search*	*Final solution by Tabu Search*	*Lingo 10 objective function solution*	*Lingo 10 Status*
11	10	5	20	10	2.8807e+06	2.1529e+06	2.19469e+06	Feasible
12	10	5	35	10	4.5432e+06	1.8447e+06	2.04029e+06	Feasible
13	10	5	40	15	1.3846e+08	1.2291e+08	*N/A*	Unknown
14	12	6	20	20	5.3461e+07	2.7977e+07	*N/A*	Unknown
15	12	6	30	15	3.4757e+07	9.4804e+06	*N/A*	Unknown
16	12	6	35	20	5.8750e+07	4.2540e+07	*N/A*	Unknown
17	15	6	25	10	2.2213e+07	4.0371e+06	*N/A*	Unknown
18	15	6	30	15	2.8428e+07	5.0215e+06	*N/A*	Unknown
19	15	6	35	20	2.7159e+07	1.8473e+07	*N/A*	Unknown
20	15	6	40	20	3.2417e+08	2.634278e+08	*N/A*	Unknown

Table 6-3 *larger problems with 10 potential warehouse locations*

Table 6-3 shows the result of this comparison. When the size of problems gets larger as we expected, we cannot reach feasible state by Lingo software after preset time for solving models (3 hours). The very few first problems reach feasible state but these objective functions are not better than Tabu solutions.

For our Tabu Search, the process starts off with best initial solution that already found by initial phase and continues with Tabu Search algorithm.

Table 6-3 demonstrates that Tabu Search have had enormous improvement in opening warehouses with different capacities and assigning customers to these opened warehouses compare to initial solution value.

The Tabu Search solving process time for last ten problems has illustrated in Table 6-4 for comparison to Lingo processing time.

Problem number	Tabu Search solving time	Lingo 10 preset time
11	00:13:36	03:00:00
12	00:16:13	03:00:00
13	00:17:40	03:00:00
14	00:21:20	03:00:00
15	00:22:00	03:00:00
16	00:29:15	03:00:00
17	00:28:00	03:00:00
18	00:22:00	03:00:00
19	00:28:00	03:00:00
20	00:36:00	03:00:00

Table 6-4 *Tabu Search process time in comparison with Lingo 10*

The processing time of Tabu Search is reasonably faster than Lingo 10 and every node of problem would be searched faster than the branch and bound method performed by Lingo.

Figure 6-4 shows the difference between linear model and Tabu Search objective function.

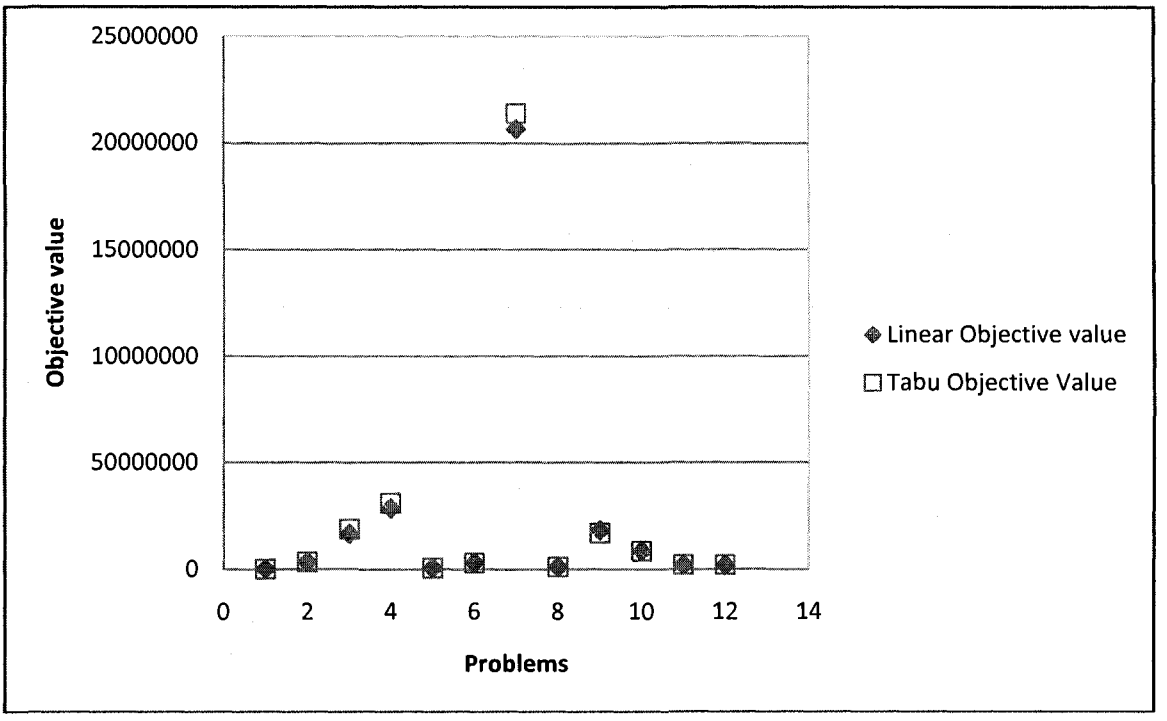


Figure 6-4 *Lingo and Tabu Search objective value comparison*

We plot first twelve problems to compare the objective functions from both linear model and Tabu Search. As it clear both solutions are so close together that the difference is ignorable.

In Table (6-3) for the problems 13 to 20, since we could not reach to any solution by Lingo software we try to illustrate the difference between the Tabu Search solution and the Lingo objective bound. By this method, we can conclude how close Tabu Search solution is to the optimal bound. We consider problem 11 to 20 where Lingo fail to find optimal or feasible solution. The objective bound of these problems have shown in table 6-5.

Problem number	Lingo objective bound	Lingo 10 preset time
11	1.97578e+06	03:00:00
12	1.16727e+06	03:00:00
13	1.18159e+08	03:00:00
14	2.37865e+07	03:00:00
15	7.7977e+06	03:00:00
16	3.66703e+07	03:00:00
17	2.67573e+06	03:00:00
18	3.18389e+06	03:00:00
19	1.4547e+07	03:00:00
20	2.40929e+08	03:00:00

Table 6-5 *Lingo 10 objective bound for problem 11 to 20.*

As it shown in figure 6-5, the gap between objective bound of Lingo for problem 11 to 20 and Tabu Search objective function is negligible. At worst case, there is only a 6% difference between the Tabu Search final answer and the Lingo objective bound. However we should bear in mind the objective bound of Lingo may be tighter even after preset time.

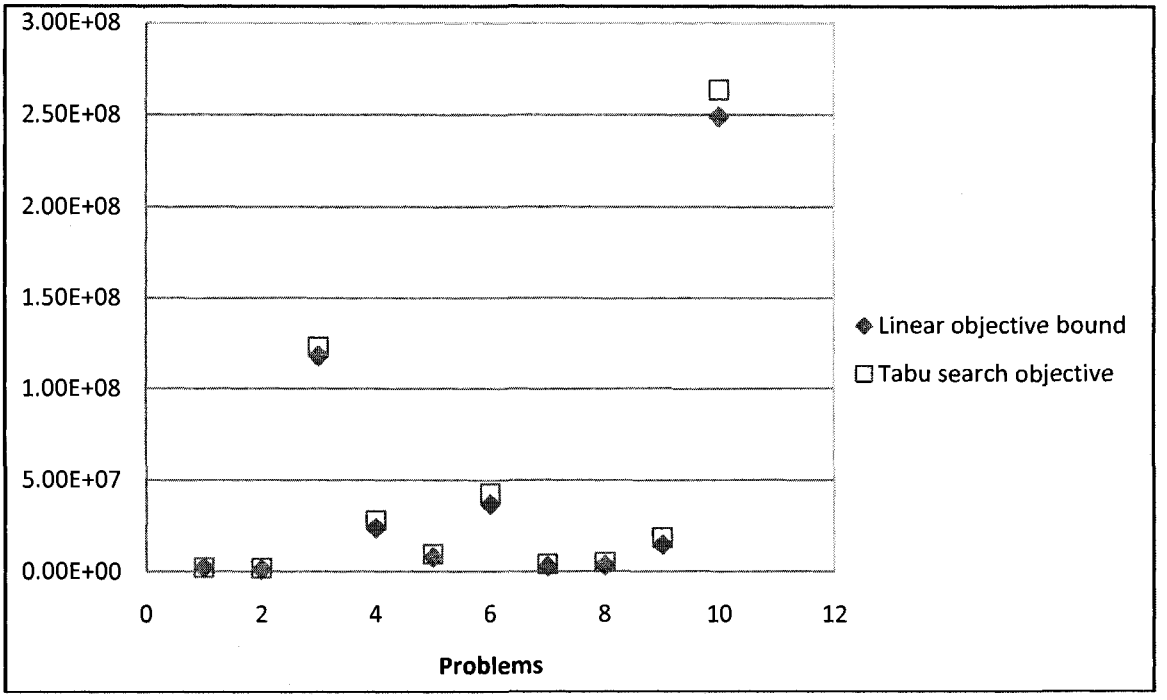


Figure 6-8 *Lingo objective bound and Tabu Search objective value comparison*

The near to optimal solution by proposed Tabu Search give us this opportunity to find a reasonable solution in a very short time when other search methods fail to find at least one feasible solution for larger problems.

6- 3 Summary

In this chapter we explained the way our Tabu Search goes for better solution in search space and we provided how Tabu Search will improve solution by explaining Tabu aspiration and Tabu tenure. Given solution by Tabu Search method for exact problems shown in Table 6-2 and 6-3 illustrates how close these solutions are to optimal point. Also by Tabu Search algorithm we acquire near to optimal solution in reasonable time comparing to linear model processing time.

Chapter 7: Conclusions and Future Research

In this research, we considered the problem of locating capacitated warehouses in a supply chain setting with staircase cost functions. A non linear integer programming formulation is presented. Based on the mathematical techniques from the literature, we transferred the nonlinear model to a linear model. We coded the nonlinear model in Lingo 8.0 and linear model in Lingo 10 software. The linear model results are found to be much better in comparison with the nonlinear model and we could reach on optimal solution for small and medium size problems within a shorter time and in an efficient way.

However for larger size problems, because of the NP hard structure of such problems, we are not able to reach a feasible solution within a reasonable time frame. For this reason, we developed a Tabu Search algorithm. The Proposed algorithm showed good quality solutions compared to those from the nonlinear model, and near optimal solutions compared to those from the linear model for medium size problems. However, when the size of the problems increases, we cannot reach a feasible solution by the branch and bound method employed by Lingo. By comparing the Tabu Search solutions with the Lingo objective bound, we can conclude that the Tabu Search results are acceptable and we can obtain good solutions for larger size problems using the Tabu Search algorithm.

For future research, we aim to develop hybrid meta-heuristic algorithms. In this way, one meta-heuristic method (such as simulated annealing, Genetic algorithm) works on

customer binary variable which assigns customers to available warehouses. Accordingly, we can get either optimal solution or improved near optimal solutions. Thus, the gap between the Tabu Search result and optimal solution will be reduced, if we use hybrid meta-heuristic methods.

Secondly, we aim to consider product demand to be stochastic as opposed to deterministic which would be a much more realistic consideration. By choosing a stochastic customer demand the way of solving problem gets more challenging. Further research will improve the customer assignment to available warehouses by their product types and demand.

References

1. Bektas, T. and Bulgak, A.A. Lagrangean-based solution approaches for the generalized problem of locating capacitated warehouses, *International Transactions in Operational Research,* 15:67-85, 2008.
2. Cortinhal, M.J. and Captivo, M.E. Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, 151:333-351, 2003.
3. Defersha, F.M. and Chen, M. A linear programming embedded genetic algorithm for an integrated cell formation and lot sizing considering product quality, *European Journal of Operational Research*, 187:46-69, 2008.
4. Delmaire, H. and Diaz, J.A. and Fernandez, E. and Ortega, M. Reactive GRASP and Tabu Search based heuristics for the Single Source Capacitated Plant Location Problem, *INFOR*; 37:194, 1999.
5. Gendron, B. and Potvin, J. and Soriano, P. A Tabu Search with Slope Scaling for the Muliticommodity Capacitated Location Problem with Balancing Requirements. *Annals of Operation Research*, 122:193-217, 2003.
6. Ghiani, G. and Guerriero, F. and Musmanno, R. The capacitated plant location problem with multiple facilities in the same site. *Computers & Operation Research*, 29:1903-1912, 2002.
7. Glover, F. and Laguna, M. Tabu Search. *Kluwer Academic Publishers*: Dordrecht, the Netherlands, 1997.
8. Hindi, K.S. and Pieńkkosz, K. Efficient solution of large scale, single-source, capacitated plant location problems. *European Journal of Operation Research*, 50:268-274, 1999.
9. Holmberg, K. Solving the staircase cost facility location with decomposition and piecewise linearization. *European Journal of Operation Research*, 75:41-61, 1994.

10. Holmberg, K. and Ling, J. A Lagrangean heuristic for the facility location problem with staircase costs. *European Journal of Operation Research*, 97:63-74, 1997.
11. Keskin, B.B. and Üster, H. Meta-heuristic approaches with memory and evolution for a multi-production/distribution system design problem. *European Journal of Operation Research*, 182:663-682, 2007.
12. Kioon, S.A. Tabu Search procedure for the design of integrated cellular manufacturing systems with production planning and dynamic system reconfiguration, PhD thesis, Concordia University, 2007.
13. Klose, A. and Drexl, A. Facility location models for distribution system design. *European Journal of Operational Research*, 162:4-29, 2005.
14. Klose, A. and Görtz, S. A branch-and-price algorithm for the capacitated facility location problem. *European Journal of Operational Research*, 179:1109-1125, 2007.
15. Lorena, L. and Senne, E. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31:863-876, 2004.
16. Pirkul, H. and Jayaraman, V. A Multi-commodity, Multi-plant, Capacitated facility location problem: Formulation and Efficient heuristic solution. *Computer & Operations Research,* 25:869-878, 1998.
17. Rolland, E. and Schilling, D.A. and Current, J.R. An efficient Tabu Search procedure for the p-Median problem, *European Journal of Operation Research*, 96:329-342, 1996.
18. Sridharan, R. The capacitated plant location problem. *European Journal of Operation Research*, 87:203-213, 1995.
19. Sun, M. Solving the uncapacitated facility location problem using Tabu Search, *Computers & operation research*; 33:2563-2589, 2006.
20. Wu, L. and Zhang, X. and Zhang, J. Capacitated facility location problem with general setup cost, *Computers & Operations Research*, 33:1226-1241, 2006.

Appendix 1: First Lingo and Tabu Search sample problem

Amount	Holding cost	Amount	Penalty cost	Amount	Opening cost	Amount	Shipping cost	Amount	Capacity size	Amount
6000	$h_{1,1,1}$	2.00	$\hat{c}_{1,1}$	800	$f_{1,1}$	1000	$c_{1,1}$	2	$w_{1,1}$	150000
4500	$h_{1,1,2}$	2.50	$\hat{c}_{1,2}$	900	$f_{1,2}$	2500	$c_{2,1}$	3	$w_{1,2}$	160000
5000	$h_{1,2,1}$	3.00	$\hat{c}_{2,1}$	900	$f_{1,3}$	3000	$c_{3,1}$	2	w_{13}	210000
3500	$h_{1,2,2}$	3.50	$\hat{c}_{2,2}$	1100	$f_{2,1}$	1500	$c_{1,2}$	4	$w_{2,1}$	50000
4000	$h_{1,3,1}$	4.00			$f_{2,2}$	3000	$c_{2,2}$	4	$w_{2,2}$	80000
2500	$h_{1,3,2}$	4.50			$f_{2,3}$	5000	$c_{3,2}$	4	$w_{2,3}$	170000
	$h_{2,1,1}$	1.35								
	$h_{2,1,2}$	1.5								
	$h_{2,2,1}$	1.51								
	$h_{2,2,2}$	1.65								
	$h_{2,3,1}$	1.7								
	$h_{2,3,2}$	1.8								

Demand	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	$d_{3,1}$	$d_{3,2}$						

Appendix 2: Tabu Search pseudo code:

Input: $f_{j,m}, h_{j,m,k}, d_{i,k}, c_{i,j}, \hat{c}_{j,k}, Y^I, \varphi(Y^I)$

Output: $Y^{best}, \varphi(Y^{best}), X_{i,j}, Z^1_{j,m,k}, Z^2_{j,m,k}$

$Y^{best} \leftarrow Y^I$; $\varphi(Y^{best}) \leftarrow \varphi(Y^I)$

$Y^c \leftarrow Y^I$; $\varphi(Y^c) \leftarrow \varphi(Y^I)$

maxIter←q; Tabu Tenure ←(j×m); maxNonImpr← p

IterNo←0; nonImpr←0

while IterNo<maxIter

    Generate t_1 solutions with different warehouse capacities.

    Calculate objective value for t_1 solutions

    Generate t_2 solutions with adding/dropping warehouse capacities.

    Calculate objective value for t_2 solutions

        for $t < t_1 + t_2$ do

            If $\varphi(Y^t) \le \varphi(Y^c)$ then

                $Y^c \leftarrow Y^t, \varphi(Y^c) \leftarrow \varphi(Y^t)$

            end if

        end for

    If TADD[Tabu]=0 and TDROP[Tabu]=0 then

        $Y^I \leftarrow Y^c; \varphi(Y^I) \leftarrow \varphi(Y^c)$

        If $\varphi(Y^c) < \varphi(Y^{Best})$ then

            $\varphi(Y^{Best}) \leftarrow \varphi(Y^c); Y^{Best} \leftarrow Y^c$; nonImpr←0

        else

            nonImpr← nonImpr + 1

        end if

        Update Tabu list T.

    else

        If $\varphi(Y^c) < \varphi(Y^{Best})$ then

            $\varphi(Y^{Best}) \leftarrow \varphi(Y^c); Y^{Best} \leftarrow Y^c$; nonImpr←0

            Update Tabu list T.

        else

            nonImpr← nonImpr + 1

            Let Y^c be the best non-Tabu solution.

            $Y^I \leftarrow Y^c; \varphi(Y^I) \leftarrow \varphi(Y^c)$

            Update the Tabu list T.

        end if

    end if

    if nonImpr > MaxNomImpr

        Terminate the Tabu Search.

    end if

    iterNo← iterNo +1

end while

Appendix 3: Nonlinear Lingo code:

```
Sets:
Warehouse/1..2/:j;
Capacity/1..4/:m,q; !I had to define one index more than real one to
cover the index 0;
Product/1..2/:k;
Customer/1..3/:i;
HoldingCost(Warehouse,Capacity,Product):h;
FixingCost(Warehouse,Capacity):f;
StorageSize(Warehouse,Capacity):S;
ShippingCost(Customer,Warehouse):c;
Demand(Customer,Product):d;
TotalStorage(Warehouse,Capacity,Product):z;
ExtraCharge(Warehouse,Product):E;
Size(product):b;
CustomerVariable(Customer,Warehouse):x;
WarehouseVariable(Warehouse,Capacity):y;
WarehouseStock(Warehouse,Product):P;
RequestStock(Warehouse,Product):R;
link(Warehouse,Customer,Product):L;
Endsets

Data:


h= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','Hold');
d= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','Demand');
f= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','OpenCost');
E= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','Penalty');
b= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','PSize');
S= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','CSize');
c= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','ShipCost');

@OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','Z_j_m_k')=z;
```

```
@OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','P_j_k')=P;
@OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','R_j_k')=R;
@OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','Y_j_m')=y;
@OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Lingo Sample
Problems\problem 1\NLP.xls','X_i_j')=x;




End Data

!Constraint(9);
@for(CustomerVariable(i,j):@BIN (x(i,j)));

!Constraint(10);
@for(WarehouseVariable(j,m)|m#GT#1:@BIN (y(j,m)));


!Constraint(13);
@for(TotalStorage(j,m,k):@GIN(z(j,m,k)));

!Constraint(14);
@for(RequestStock(j,k):@GIN(R(j,k)));

!objective function;

min=@sum(HoldingCost(j,m,k)|m#GE#2:h(j,m,k)*z(j,m,k))+
@sum(FixingCost(j,m)|m#GE#2:(f(j,m)-f(j,m-
1))*y(j,m))+@sum(HoldingCost(j,m,k)|m#GE#2:((h(j,m,k)-h(j,m-
1,k))*(@sum(Capacity(q)|q#LE#m-1:z(j,q,k))))*y(j,m)) +
@sum(link(j,i,k):d(i,k)*x(i,j)*c(i,j))+
@sum(WarehouseStock(j,k):R(j,k)*E(j,k)) ;



!Constraint(1);
@for(Customer(i):
            @sum(Warehouse(j):x(i,j))=1);

!Constraint(2);
@for(CustomerVariable(i,j):
                  x(i,j)<=y(j,2));


!Constraint(3);
@for(StorageSize(j,m)|m#GE#2:
            @sum(product(k):b(k)*z(j,m,k))<=(s(j,m)-s(j,m-1))*y(j,m));
```

```
!Constraint(4);
@for(StorageSize(j,m)|m#GT#2:
            @sum(product(k):b(k)*z(j,m-1,k))>=(s(j,m-1)-s(j,m-
2))*y(j,m));


!Constraint(6);
@for(WarehouseStock(j,k):
            P(j,k)+R(j,k)=@sum(Customer(i):d(i,k)*x(i,j)));

!Constraint(7);
@for(WarehouseStock(j,k):
            @sum(capacity(m)|m#GT#1:z(j,m,k))=p(j,k));


!Constraint(8);
@for(TotalStorage(j,m,k):
                  z(j,m,k)>=0);
@for(TotalStorage(j,m,k)|m#EQ#1:
                  z(j,m,k)=0);

!Constraint(9);
@for(WarehouseStock(j,k):
                    p(j,k)>=0);
!Constraint(10);
@for(RequestStock(j,k):
                     R(j,k)>=0);
```

Appendix 4: Linear Lingo code:

```
Sets:
Warehouse/1..15/:j;
Capacity/1..7/:m,q; !I had to define one index more than real one to
cover the index 0;
Product/1..10/:k;
Customer/1..25/:i;
HoldingCost(Warehouse,Capacity,Product):h;
FixingCost(Warehouse,Capacity):f;
StorageSize(Warehouse,Capacity):S;
ShippingCost(Customer,Warehouse):c;
Demand(Customer,Product):d;
TotalStorage(Warehouse,Capacity,Product):z,AZ;
ExtraCharge(Warehouse,Product):E;
Size(product):b;
CustomerVariable(Customer,Warehouse):x;
WarehouseVariable(Warehouse,Capacity):y,t;
WarehouseStock(Warehouse,Product):P;
ProblemsStock(Warehouse,Product):R;
link(Warehouse,Customer,Product):L;
Endsets

Data:


h= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','Hold');
d= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','Demand');
f= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','OpenCost');
E= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','Penalty');
b= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','PSize');
S= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','CSize');
c= @OLE('C:\Documents and Settings\Iman Niroomand\My Documents\My
university Research\Thesis\DifferentModels\Larger Tabu Problems
C++\Lingo 12\problem12.xls','ShipCost');

End Data

!objective function;

min=@sum(HoldingCost(j,m,k)|m#GE#1:h(j,m,k)*AZ(j,m,k))+
@sum(FixingCost(j,m)|m#GE#2:(f(j,m)-f(j,m-
```

```
1))*y(j,m))+@sum(link(j,i,k):d(i,k)*x(i,j)*c(i,j))+
@sum(WarehouseStock(j,k):R(j,k)*E(j,k)) ;

!Constraint(1);
@for(Customer(i):
            @sum(Warehouse(j):x(i,j))=1);

!Constraint(2);
@for(CustomerVariable(i,j):
                  x(i,j)<=y(j,2));


!Constraint(3);
@for(StorageSize(j,m)|m#GE#2:
            @sum(product(k):b(k)*z(j,m,k))<=(s(j,m)-s(j,m-1))*y(j,m));



!Constraint(4);
@for(StorageSize(j,m)|m#GT#2:
            @sum(product(k):b(k)*z(j,m-1,k))>=(s(j,m-1)-s(j,m-
2))*y(j,m));


!Constraint(5);
@for(WarehouseStock(j,k):
            P(j,k)+R(j,k)=@sum(Customer(i):d(i,k)*x(i,j)));

!Constraint(6);
@for(WarehouseStock(j,k):
            @sum(capacity(m)|m#GT#1:z(j,m,k))=p(j,k));


!Constraint(7);
@for(TotalStorage(j,m,k):
                  z(j,m,k)>=0);
!Constraint(8);
@for(TotalStorage(j,m,k)|m#EQ#1:
                  z(j,m,k)=0);

!Constraint(9);
@for(WarehouseStock(j,k):
                    p(j,k)>=0);
!Constraint(10);
@for(ProblemsStock(j,k):
                     R(j,k)>=0);
!Constraint(11);
@for(TotalStorage(j,m,k)|m#GE#2:
            AZ(j,m,k)>=@sum(Capacity(q)|q#GE#2 #AND#
q#LE#m:z(j,q,k))+10000000000*t(j,m)-10000000000);
!Constraint(12);
@for(TotalStorage(j,m,k)|m#GE#2:
            AZ(j,m,k)<=@sum(Capacity(q)|q#GE#2#AND# q#LE#m:z(j,q,k)));
!Constraint(13);
@for(TotalStorage(j,m,k)|m#GE#2:
            AZ(j,m,k)<=10000000000*t(j,m));
```

```
!Constraint(14);
@for(StorageSize(j,m)|m#GE#1:
                        t(j,m)<=y(j,m));
!Constraint(15);
@for(Warehouse(j):
      @sum(Capacity(m)|m#GE#1:t(j,m))=1);
!Constraint(16);
@for(StorageSize(j,m)|m#GE#2:
                        t(j,m)>=t(j,m-1)+10000000000*y(j,m)-
10000000000);


!Constraint(17);
@for(CustomerVariable(i,j):@BIN (x(i,j)));

!Constraint(18);
@for(WarehouseVariable(j,m)|m#GT#1:@BIN (y(j,m)));


!Constraint(19);

@for(WarehouseVariable(j,m):@BIN (t(j,m)));


!Constraint(20);
@for(TotalStorage(j,m,k):@GIN(z(j,m,k)));

!Constraint(21);
@for(ProblemsStock(j,k):@GIN(R(j,k)));


!Constraint(22);
@for(TotalStorage(j,m,k)|m#EQ#1:
                  AZ(j,m,k)=0
```