A Simulation-Based Optimization System for Green Building Design

Weimin Wang

A Thesis

in

The Department

of

Building, Civil, and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

February, 2005

Your file Votre référence
ISBN: 0-494-04054-8
Our file Notre référence
ISBN: 0-494-04054-8

ABSTRACT

A Simulation-Based Optimization System for Green Building Design

Weimin Wang, Ph.D.
Concordia University, 2005

Green building is a recent design philosophy that requires the consideration of resources depletion and waste emissions during its whole life cycle. Simulation-based optimization can assist designers to achieve a better building design by overcoming the drawbacks of trial-and-error with simulation alone. This dissertation presents the design and implementation of a simulation-based optimization system for the conceptual design of green buildings.

In the optimization model, variables are mostly envelope-related design parameters such as orientation, building shape, wall type, and wall layer. The concept of structured variable is used to describe the hierarchical relationship between variables. Life-cycle cost and life-cycle environmental impact are two major objective functions that respectively evaluate the economical and environmental performance of a building. The impact categories considered in this research include resource depletion, global warming, and acidification. They are unified together with the indicator "expanded cumulative exergy consumption", which is calculated as the sum of the cumulative exergy consumption due to resource inputs, and the abatement exergy consumption due to waste emissions.

The system consists of four components: the input and output, the optimizer, the simulation programs, and the data files. The genetic algorithm is implemented in the optimizer to solve both single- and multi- objective optimization problems. The simulation programs are developed based on the ASHRAE toolkit for building load calculations in order to evaluate objective functions and functional constraints. The system is developed with the object-oriented technology. An object-oriented framework, which is a reusable software architecture represented by a set of classes, is proposed in this research to facilitate the reuse of code and software design. This framework can act as a basis to solve many other simulation-based optimization problems.

A case study is used to demonstrate the application of the system. In this case study, a multi-objective genetic algorithm is employed to optimize a single-story office building in terms of the life-cycle cost and life-cycle environmental impact. The case study resulted in multiple Pareto solutions which can help designers to understand the trade-off relationship between reducing environmental impacts and increasing costs due to green design strategies.

Acknowledgement

First, I would like to thank my supervisors, Dr. Hugues Rivard and Dr. Radu Zmeureanu, for guiding me to carry out this research. Thanks for their encouragement and criticism which motivated me to strive for the best. Their enthusiasm and strictness in research will continue to be my learning examples.

I would like to thank all my Ph.D. committee members for their comments and suggestions on this work.

I am grateful to Dr. Karen Liu from the National Research Council of Canada for providing the experimental data on green roof temperatures. I appreciate Dr. Hua Ge and Dr. Dominique Derome for their discussions on the envelope types. I am indebted to Mr. Sylvain Bélanger for his useful input in solving several computing problems.

Sincere thanks need to go to all my friends and colleagues, who have enriched my life at Concordia University in the past four years.

I thank my parents for their selfless love and support. I owe them so much that I can never pay back.

I greatly acknowledge the financial support from the EJLB foundations in Canada.

Table of Contents

List of Figures

List of Tables

Nomenclature

a	length, m
A	area, m2
AbatEx	abatement exergy, MJ
ACH	air change per hour, hr-1
b	width, m
CExC	cumulative exergy consumption, MJ
d	normalized Euclidean distance
dr	demand rate, \$/kW
dt	demand threshold value, kW
e	exergy, MJ/kg
E	energy cost, \$
EE	environmental impact in the operation stage, MJ
EN	embodied energy, MJ
EPD	equipment power density, W/m^2
f	production coefficient, MJ/kg
F	view factor
g	acceleration of gravity, m/s^2
h	convective heat transfer coefficient, W/(m^{2}K)
	building height, m
i	discount rate
I	investment cost, \$
IC	initial construction cost, \$
LCC	life-cycle cost, \$
LCEI	life-cycle environmental impact, MJ
LPD	lighting power density, W/m^2
m	mass, kg
M	non-fuel operating and maintenance costs, \$

n	study period of life cycle analysis for a building, yr
N	population size
OC	operating cost due to energy consumption, $
OE	environmental impact in the pre-operation stage, MJ
ON	annual on-site operating energy consumption, MJ/yr
p	pressure, Pa
PD	peak demand, kW
PI	performance index
q	heat flux, W/m^2
Q	thermal load, W
r	energy price escalation rate
R	repair and replacement cost, $
s	specific entropy, J/(kg·K)
S	floor area, m^2
T	temperature, K
u	specific internal energy, J/kg
X	outside CTF coefficient;
Y	cross CTF coefficient
z	height, m
Z	inside CTF coefficient
v	velocity, m/s
V	volume, m^3

Subscripts

0	reference state
a	indoor air
αsol	solar radiation
conv	convective
g	ground
k	non-fuel resource
ki	conduction heat flux on inside surface

ko	conduction heat flux on outside surface
LWR	longwave radiation exchange with the environment
LWX	radiation exchange between internal surfaces
m	objective function
MRT	fictitious surface
o	outdoor air
os	operation stage
p	present value
ps	pre-operation stage
radDist	radiation distributed on inside surfaces
si	inside surface
so	outside surface
sys	heating or cooling system
t	time
w	waste emission

Greek Letters

α	ratio of the chemical exergy to the energy content of a given fuel
β	wall tilt, degree
η	exergetic efficiency
η'	energetic efficiency
ρ	density, kg/m^3
υ	specific volume, m^3/kg
σ	Stefan-Boltzmann constant (=5.67*10^{-8} W/m^{2}K^4)
ϵ	emittance
δ	time step
Φ	flux CTF coefficient

Chapter 1
Introduction

1.1 Background

Climate change is one of the major global environmental challenges facing the whole world today. It has been established by the International Panel on Climate Change (IPCC) that man-made greenhouse gas (GHG) emissions are at the root of the climate change (Watson and the Core Writing Team 2001). The IPCC reports that the concentration of three primary GHGs, carbon dioxide (CO_2), methane (CH_4), and nitrous oxide (N_2O), in the atmosphere increased by about 31%, 151%, and 17%, respectively, in the past two hundred and fifty years.

The increases of GHG concentrations have enhanced the heat-trapping capability of the earth's atmosphere. This effect tends to alter atmospheric and oceanic temperatures, weather patterns, and the entire hydrological cycle. The report from the IPCC (Watson and the Core Writing Team 2001) indicates that the global average surface temperature increased by about 0.6 °C over the 20th century and that the global mean sea level rose about 150 mm during the same period. This trend will continue at an increasing rate if no effective measures are taken from now. Table 1.1 shows the alarming increases of CO_2 concentration, global average temperature, and mean sea level over the next century.

Table 1.1 Trend of global CO_2 concentration, temperature and sea level change

Year	**CO_2 concentration increase (ppm)**	**Global average temperature change (°C)**	**Global sea level rise (cm)**
1990	354	0	0
2000	367	0.2	2
2025	405-460	0.4-1.1	3-14
2050	463-623	0.8-2.6	5-32
2100	540-970	1.4-5.8	9-88

Data source: IPCC (Watson and the Core Writing Team 2001)

Besides climate change, there are also other regional or local environmental challenges such as acidification, urban air pollution, and the loss of biological diversity. Climate change and regional environmental issues may exacerbate each other, involving complex interactions. For example, climate change could aggravate the regional air pollution and delay the recovery of the ozone layer (Watson and the Core Writing Team 2001). Both climate change and regional environmental problems are due to waste emissions from human activities, in particular, the burning of fossil fuels.

For Canada, the anthropogenic GHG emissions on a per capita net basis are relatively high compared with those of other nations. Canada has roughly 0.5% of the world's population but produces about 2.2% of the total global GHG emissions (Government of Canada 2001). In 2001, Canadians contributed about 720 million tones (MT) CO_2 equivalent to the atmosphere, an increase of almost 19% over 607 MT recorded in the year 1990 (Environment Canada 2003). About 80% of Canada's total GHG emissions are due to energy-related human activities, while the other 20% emissions come mainly from non-energy sources such as industrial processes and waste disposal.

If all current federal and provincial policies on energy and environment are held constant, the projected GHG emissions by the year 2010 in Canada will increase to 764 MT. This number is 199 MT above the target established in the Kyoto Protocol, which stipulates that by the period 2008-2012, Canada will reduce its GHG emissions to 6% below its 1990 level (Analysis and Modeling Group 1999).

There is a close relationship between buildings and environment. According to the U.S. Department of Energy (2004a), buildings account for 39% of the total primary energy consumption and 71% of the electricity consumption. Because buildings are one of the major energy consumers, they produce a large portion of GHG emissions, air pollutants, and solid wastes. For example, in the year of 2002, buildings in the U.S. produced nearly 38 percent of CO_2, 52 percent of SO_2, and 20 percent of NO_x emissions (U.S. Department of Energy 2004a).

Canada has a similar situation as the U.S. in terms of the relationship between buildings and environment. A recent report has shown that in 2002, buildings in Canada consumed about 2529 petajoules (PJ) of secondary energy and produced 139 MT of greenhouse gases (Natural Resources Canada 2004). The distribution of energy consumption and GHG emissions among the five end-use sectors are listed in Table 1.2.

Table 1.2 Secondary energy use and the associated GHG emissions distribution in Canada, 2002

Sector	**Energy (PJ)**	**GHG Emissions (MT)**
Residential	1399	75
Commercial	1130	64
Industrial	3176	163
Transportation	2306	165
Agriculture	206	14

Data source: Natural Resources Canada (2004)

It can be concluded from the above discussions that building construction and operation require enormous amounts of energy and create large amounts of waste emissions. How they are built can affect the ecosystems around us in countless ways.

1.2 Problem Statement

As the environmental impacts of buildings become more apparent, it is important to start incorporating environmental performance as a criterion in building design. Green building is a recent design philosophy that requires the consideration of resources depletion and waste emissions during its whole life cycle (Woolley et al. 1997). The term "green building" is often used interchangeably with "sustainable building", "environment-friendly building" and "energy-efficient building", even though their implications may be a little different (Cole 1999). Whatever the definition is, some basic principles such as energy performance, resource efficiency, low waste emissions, and indoor environmental quality should be considered in green building design (U.S. Green Building Council 2003).

Although the environmental and social benefits of green buildings are widely accepted, cost is a prohibitive factor to integrate sustainable building practices into projects. Because property developers usually make their decisions based on the initial cost, the application of green building is prematurely labeled as "too costly" or "economically unattractive" (Kats et al. 2003). This wrong perception is caused by many reasons, but the most important one is the lack of appropriate economical performance criterion when comparing different alternatives for a green building design. Therefore, it is essential to consider both environmental performance and economical performance in view of the whole life of buildings in order to justify the additional investments associated with green building design strategies.

The successful design of green buildings requires that special attention be paid to the conceptual design phase when many potential design alternatives are generated and roughly evaluated in order to obtain the most promising solution. Decisions made in the phase of conceptual design have considerable impacts on building performance including the environmental aspect. For example, simply making buildings the right shape and the correct orientation can reduce the energy consumption by 30-40% at no extra cost (Cofaigh et al. 1999). Because of the intertwined interrelationships among numerous design parameters, multiple performance criteria, and different life-cycle stages, designers require computer assistance in green building design, especially in the conceptual design phase. Although many building simulation programs have been developed, the usual trial-and-error procedure for performance improvement is time-consuming and ineffective because of the inherent difficulty in searching a large design space. Therefore, a computer tool that can effectively aid designers in the conceptual phase of green building design is needed to improve the performance of new buildings.

As indicated in the technology roadmap for high-performance commercial buildings (U.S. Department of Energy 1999), the fundamental goal of building design is to optimize the building's performance through a whole-building approach. This means that the interactions between building subsystems are considered to find the optimal solutions in terms of comfort, functionality, resource efficiency, and life-cycle cost. This research attempts to make some contributions to achieve the above goal.

1.3 Research Objectives

The objective of this research is to develop a simulation-based optimization system for the conceptual design of green buildings. This system considers both life-cycle cost and life-cycle environmental impact and can obtain the optimal values for many building envelope-related parameters, which are usually determined at the conceptual design phase, and hence have critical influence on building performance. In detail, the research intends to:

- Establish an optimization environment in which single- and multi- objective optimization problems can be solved. For multi-objective optimization, both economical and environmental performances are considered to help designers understand their trade-off relationships.

- Provide high flexibility so that designers can define the number and types of variables, constraints, and objective functions. In other words, the system offers designers the freedom to establish a variety of optimization models.

- Analyze the characteristics of optimization problems that use simulation programs to evaluate objective functions and constraints. Based on the analysis, a computer system is developed to facilitate the reuse and the extension to other similar problems.

- Consider various environmental impact categories that have global, regional, and long-lasting impacts on the environment. In this way, the environmental impacts are no longer only limited to energy consumption, which is insufficient as an indicator for environmental performance because many impacts are not energy-related.
- Employ the life-cycle analysis methodology to expand the scope from building operation only to cover both construction and operation. Thus, initial construction cost and environmental impacts from material acquisition, transportation, and construction are considered in the corresponding objective functions.
- Apply the developed system to optimal building design through case studies. The impacts of different objective functions on the optimal solutions are investigated, from which some general guidelines are drawn to be used by designers.
- Identify the potential difficulties in solving simulation-based optimization problems for green building design and suggest future research directions to address the above difficulties.

1.4 Methodology

To achieve the stated objectives, this research employs the following methodology:

- A literature review is conducted at first to identify the limitations with previous related studies.
- The optimization model is developed based on a clearly defined scope. An indicator is used in the model to evaluate the environmental performance of a building design.
- The system design is carried out after analyzing the general characteristics for simulation-based optimization problems.
- The system is implemented.
- The implemented optimizer is validated by mathematical optimization problems with known optimal solutions.
- A case study is employed to demonstrate the application of the system to solve green building optimization problems.

1.5 Organization of the Dissertation

This dissertation is organized as follows:

- Chapter 2 gives a literature review about the indicators for environmental performance and the methodologies for green building design. A survey about previous related optimization studies is conducted, and their major limitations are identified to justify this research.

- Chapter 3 presents an optimization model including its variables, constraints, and objective functions. To establish the model, the scope for this optimization study is carefully defined to capture the interactions between building components and subsystems while keeping it manageable. A new indicator is proposed to evaluate the environmental performance of a building design. This indicator is used in one of the objective functions in the optimization model.

- Chapter 4 elaborates on the formulation of the simulation-based optimization system. The components of the system, namely, the simulation programs, the optimizer, the data files, and the input and output are discussed in detail.

- Chapter 5 focuses on the computer model of the system. An object-oriented framework is developed to facilitate the reuse of the software design. Some fundamental knowledge of object-oriented design is briefly presented. Then, the framework design and its customization are presented through the major modules of the computer model.

- Chapter 6 discusses the validation of the system. Several mathematical optimization problems are employed to test the implemented algorithms.

- Chapter 7 applies the validated system to optimize a building design for two performance criteria: life-cycle cost and life-cycle environmental impact. The original problem formulation is modified in terms of optimization criteria to investigate the corresponding changes in solutions. The results are presented and analyzed.

- Chapter 8 ends this dissertation with some conclusions and suggestions for future work.

Chapter 2
Literature Review

Many studies have been done in green building design. This chapter reviews previous studies that are related to this research. Because life-cycle assessment is the basis for evaluating the environmental performance of green buildings, its general framework and indicators are briefly presented in the first section. Then, methods for green building design are reviewed. Previous building design optimization studies for environmental performance are discussed further in the third section.

2.1 Green Building Performance Assessment

The performance of green buildings can be evaluated from a broad range of aspects. In GBTool (Cole and Larsson 2002), for example, seven performance issues are considered: resource consumption, loadings, indoor environmental quality, quality of service, economics, pre-operations management, and commuting transportation. Similar issues can be observed in other performance assessment frameworks such as LEED (U.S. Green Building Council 2003) and BREEAM (British Research Establishment 2003). Since environmental performance is the essential issue that makes green buildings distinctive from traditional building design, this section focuses on the assessment of environmental performance for buildings.

2.1.1 Life-Cycle Assessment

Life-cycle assessment (LCA) is a system analysis method used to understand and evaluate the environmental aspects and potential impacts associated with a product, process, or service, across all stages of its life cycle from raw material acquisition to final disposition (Guinee 2002). Compared with other environmental management approaches such as risk assessment and material flow analysis, LCA is able to track and reveal the shifts of environmental impacts between different media and life-cycle stages. The importance of the LCA lies in its key features: a system-wide perspective, a multi-medium outlook, and the use of a functional unit accounting system to normalize the data (Barnthouse et al. 1998). Figure 2.1 shows the life-cycle process of buildings, covering from natural resource extraction, through material production, construction, and operation, to demolition. Maintenance and renovation are usually required during the operation stage, and transportation is an activity associated with most other stages.

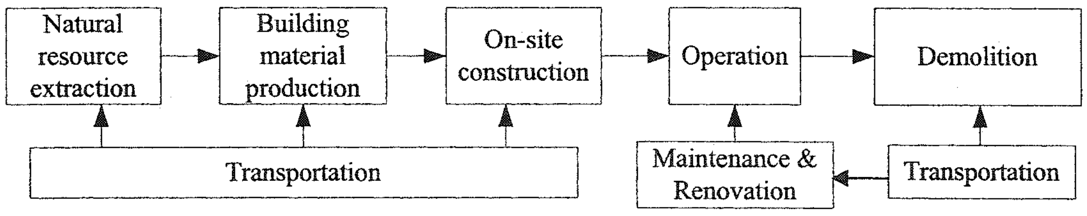


Figure 2.1 Building life-cycle process

An LCA may be divided into four major phases (Guinee 2002): (1) goal definition and scoping; (2) inventory analysis; (3) impact assessment; and (4) interpretation. Each phase is briefly presented below.

In the goal definition and scoping phase, the purpose of LCA and the system boundary in terms of time and space are defined. In the context of building design optimization, for example, LCA could be used by designers to compare different design alternatives with respect to environmental performance. Therefore, a number of decisions such as the environmental issues to be considered, the life-cycle stages to be covered, and the indicator to be used must be made in this phase.

In the inventory analysis phase, the inputs and outputs of the system under study are compiled and quantified. The outcome of this phase is a list containing the amount of resource consumption (e.g., energy and iron ore) and the quantities of waste emissions (e.g., CO_2 and lead) released to the environment. The development of a life-cycle inventory for a building is usually the most resource-intensive phase, which requires the following information:

- The life-cycle inventory data, including both resource inputs and waste outputs, for each material and product required for the construction of a building.
- The quantity of each required material and product. It is better to use the same unit for life-cycle inventory data and for quantity calculation; otherwise, transformations between different units are involved.
- Resource consumption and waste emissions from the on-site construction stage and the demolition stage. Because the environmental impacts of these two stages are largely influenced by the specific characteristics of a building, many uncertainties are involved in this step.
- Environmental impacts due to building operation. Energy simulation programs are

usually employed to calculate energy consumption, which is the dominant resource input at the operation stage. The amount of each waste emission is derived from energy consumption and the corresponding emission factor, which is the estimated average emission rate for the on-site consumption of a certain fuel.

In the impact assessment phase, the inflows and outflows identified in the inventory analysis phase are processed to evaluate their potential environmental impacts on human health, ecosystem quality, and resource depletion. This phase consists of a number of mandatory steps (i.e., selection of impact categories, classification, and characterization) and optional steps (i.e., normalization, grouping, weighting, and data quality analysis) (Guinee 2002). Some steps are presented here to facilitate later discussions.

- Selection of impact categories. Commonly used impact categories include resource depletion, global warming, ozone depletion, acidification, eutrophication, photochemical smog, human toxicity, ecotoxicity, and solid waste (Udo de Haes 1996).

- Classification. Inventory flows are assigned to different impact categories based on the expected impacts on the environment. For example, CO_2 emissions are classified into global warming.

- Characterization. The inventory flows within each impact category are aggregated into a common unit. Equivalency assessment is a methodology that is often employed in this step to evaluate the emission loadings or resource use and to represent them as impact indicators. For example, different green house gases are converted into the CO_2 equivalent and added together to represent the global warming potential. Characterization methods determine the depth level of impact assessment, which is relevant to the elements in the cause-effect chain and the degree of quantification.

- Normalization. This step calculates normalized indicators relative to reference values in order to better understand the relative importance of all considered impact categories. Normalization is necessary if the subsequent weighting step is performed.
- Weighting. All the environmental impacts are integrated together based on the relative weights assigned to the different impact categories. Subjective influence in this step cannot be eliminated, although some methods such as analytical hierarchy process (Saaty 1990) are helpful to obtain systematic and consistent weights.

In the interpretation phase, the consistency and completeness of results obtained from the impact assessment are checked according to the goal defined in the first phase. The results are reported in a neutral and informative manner. The opportunities to reduce the environmental impacts of the system under study are evaluated.

2.1.2 Environmental Performance Indicators

The previous subsection has shown that a number of impact categories are considered in life-cycle assessment. These different categories can be aggregated together with the weighting step. Although there is a concern about the loss of transparency after weighting

(ISO 1997), a single indicator is useful for building optimization problems because: (1) when impact categories are dealt with separately, it increases the number of objective functions that must be handled by the optimization; and (2) if LCA results are expressed with several impact indicators with different scales and units, it is difficult for designers to compare different design alternatives. Therefore, different aggregated indicators for environmental performance are reviewed in this subsection so that an appropriate one can be selected for this research.

Many environmental performance indicators have been proposed in the past decades, as listed in Table 2.1. According to the methodology used to integrate impact categories, each indicator is classified into one of five groups of weighting methods including proxy, technology, panels, monetization, and distance-to-target (Finnveden 1999). Each group may be further divided into several sub-groups based on the underlying concepts. For each concept, an indicator or a method is provided together with its references, advantages, disadvantages, and previous application level in the building hierarchy (e.g., material, system, or whole building). If the use of an indicator in the field of buildings is not found in the literature, the corresponding cell in the last column of Table 2.1 is labeled as N.A. for not available. The five groups of weighting methods are briefly discussed below.

- The proxy approach uses one or a few reference parameters to measure the total environmental impacts. Representative reference parameters include mass, energy, and exergy. The proxy approach is usually easily understandable if the chosen reference parameter is widely used. A major disadvantage of the proxy approach is that it concentrates on resource consumption, but ignores waste emissions. However, the subsequent technology approach can be combined with this approach to consider both resource consumption and waste emissions.

- The technology approach depends highly on the available technology to reduce environmental burdens. For example, the indicator "ecological footprint" expresses the areas of land and water ecosystems needed to provide the resources and to assimilate the waste of the system being studied (Chambers et al. 2000). The footprints for both resource production and waste assimilation are technology-dependent.

- The panel approach aims at developing weights for different impact categories based on the perceived importance or relevance of the surveyed people. Indicators using the panel approach can vary in the following aspects: (1) the characterization model used to convert the inventory data of a given impact category into a common metric. For example, the fate-effect model is used in the Eco-indicator 99 (Goedkoop and Spriensma 2001), and the equivalency model is used in the UK Ecopoints (Dickie and Howard 2000); (2) the reference values used in the normalization step, which may be the total impacts for the globe, a region, per capita, or others (Pennington et al. 2004); and (3) the implementation of the panel approach in terms of panellists and procedures (Finnveden 1999).

- The monetary approach attaches monetary values to every considered impact category. Thus, diverse impacts can be unified by translating them into the same monetary units. Unlike goods or services tradable in a competitive market, environmental resources (e.g., clean air and quiet surroundings) are public goods that are not explicitly priced in the market. Some specially designed methods such as the hedonic pricing method, the damage cost avoided method, and the contingent valuation method must be employed to estimate the economic values of environmental resources (Haab and McConnell 2002). Of these methods, contingent valuation is the most widely used one because of its ability to estimate non-use values associated with environmental services. However, because the survey-based process involves asking people directly how much they would be willing to pay in order to preserve specific environmental service, many factors such as respondents, survey forms, and question formulations have great impacts on the results (Matthews and Lave 2000).

- The distance-to-target approach is essentially a normalization technique with some kind of target or standard as the normalization coefficient for each impact category. As described by Finnveden (1999), indicators within this group may vary in terms of: (1) the specific formula used to relate the targets or standards to weighting factors; (2) the choice of targets or standards; and (3) whether inventory data or characterized data are used in weighting. Of the above three influential factors, the second one has significant impacts on the LCA results (Lindeijer 1996).

An indicator combining the proxy approach and the technology approach is used in this research to evaluate the environmental performance of a building. This indicator is discussed in detail in the next chapter.

Table 2.1 Overview of environmental performance indicators

Group	Underlying concept	Example of indicator or method	Reference	Considered impacts	Advantages	Disadvantages	Previous application level in buildings
Proxy	Mass	Material input per service unit (MIPS)	Stiller (1999)	Abiotic and biotic raw materials, water, erosion and air.	Easily understandable	▪ Does not distinguish resources with different abundance. ▪ Does not consider emissions.	Material
	Energy	Life-cycle energy	Cole and Kernan (1996) Adalberth (1997)	Energy	Easily applicable and understandable	▪ Does not account for non-energy resources. ▪ Does not consider waste emissions. ▪ Does not distinguish energy sources with different quality.	Material, assembly, system, whole building
	Exergy	Total equivalent resource exergy	Zhang (1995)	Energy, abiotic raw materials, waste emissions	▪ Can distinguish energy quality. ▪ Can account for non-energy resources. ▪ Can consider waste emissions. ▪ Does not use weights.	▪ Does not distinguish renewable and non-renewable energy. ▪ Difficulty in establishing the relationship between exergy and waste emissions.	Energy conversion system
		Sustainability coefficient	Dewulf et al. (2000)	Same as above	▪ Same as above ▪ Distinguish renewable and non-renewable energy.	▪ Does not consider utilization phase. ▪ Need to calculate the exergy loss in production process.	N.A.
		Exergy Loss	Cornelissen and Hirs (2002)	Energy, abiotic raw materials, wood	▪ Can determine the location and magnitude of exergy loss in the production process. ▪ Only exergy of relevant materials and products is needed.	▪ Does not consider the impact of emissions on the environment. ▪ Does not reflect the influence of materials with different cumulative exergy consumption.	Heating system

Group	Underlying concept	Example of indicator or method	Reference	Considered impacts	Advantages	Disadvantages	Previous application level in buildings
Techno logy	Carrying capacity	Ecological Footprint	ECOTEC (2001)	Biotic resource, land, CO_2 emissions	▪ Easily understandable ▪ Applicable to both macro-level (e.g., national) and micro-level (e.g., product).	▪ Does not account for waste emissions other than CO_2. ▪ Does not account for abiotic resources such as mineral ore.	N.A.
Panels	equivalency	UK Ecopoints	Dickie and Howard (2000)	Abiotic and biotic raw materials, waste emissions	▪ Comply with the LCA framework	▪ Require weights ▪ Data is limited to UK.	Whole building
	Damage function	Eco-indicator 99	Goedkoop and Spriensma (2001)	Abiotic and biotic raw materials, waste emissions, land	▪ Comply with the LCA framework ▪ Evaluation is deep in the cause-effect chain.	▪ Require weights ▪ Data is limited to the European Union.	Material
Moneti zation	Willingness to Pay	Environmental priority strategies (EPS)	Steen (1999)	Abiotic and biotic raw materials, waste emissions, land	▪ Evaluation is deep in the cause-effect chain. ▪ Easily to be applied.	▪ Default data is applicable to only OECD countries.	Material
	External cost	ExternalE	European Commission (1997)	Energy	▪ Evaluation is deep in the cause-effect chain. ▪ Can integrate with the cost analysis.	▪ Does not account for resource consumption. ▪ Data is available only for the European Union.	N.A.
	Ecotax	Taxes	Reijnders (2003)	Energy, waste emissions	▪ Can integrate with the cost analysis.	▪ Taxes are political and subjective. ▪ Taxes may be not available for all emissions. ▪ No direct evaluation of impacts on environment.	N.A.
Distanc e to target	Absorption capacity of environment	Swiss Ecopoints	Hertwich et al. (1997)	Waste emissions	▪ Consider the dilution capacity of nature and the current pollution level.	▪ Site-dependent. ▪ Difficulty in determination of the dilution capacity and current pollution level. ▪ Does not account for resource use.	N.A.

2.2 Green Building Design Methods

A basic principle of green building design is to reduce the negative impacts on built environment while taking cost and other performance criteria into account. This principle is implemented in previous studies through four methods: the empirical rule or guideline based manual method, the simulation based trial-and-error method, the knowledge-based method, and the optimization method. These four methods are discussed below.

2.2.1 Empirical Rule or Guideline Based Manual Method

Empirical rules or guidelines are generalized statements, tables or diagrams that can guide decision-making for good design. They can be quickly and easily applied at the initial building formulation stage because no demanding calculation is required. If used correctly, empirical rules or guidelines can provide a shortcut to achieve a design alternative with good performance.

Most design guidelines aim at reducing energy consumption in the operation stage of buildings. They are usually available in handbooks about passive solar building design (Bansal et al. 1994; Steven Winter Associates 1998). Green Building Advisor (2004) is a computer tool that offers qualitative design strategies to help designers identify applicable measures to improve the environmental performance of a building project. There is always a trade-off relationship between the application scope and the practical significance of these design guidelines. Some general rules such as "place the most windows on the south side and fewest on the north side to maximize solar gain and reduce heat loss" (Steven Winter Associates 1998) are obvious and easy to implement in the design process. On the contrary, specific rules indicating numeric values may be valuable in practice, but they can easily become unwieldy if they have too many preconditions. For example, a rule of thumb for direct-gain systems indicates that the south glazing area should be 26% of the floor area in Salt Lake City, U.S. (Lechner 2001). This guideline is valid only if a number of preconditions are satisfied. Some examples of these preconditions are as follows: adequate thermal storage is provided, the building is well insulated, and double-glazing type windows are used. Unfortunately, the essential preconditions tend to be forgotten in practice.

To address the inability of component-oriented rules to reflect interactions between various components of a building, computer-based models can be used to derive guidelines. A life-cycle cost calculation model is employed by Balcomb (1984) to make a study on passive solar heating design for residential buildings. Guidelines are recommended on the basis of correlations between different parameters that are obtained from evaluations of that life-cycle cost model, and these guidelines expressed in tables and formulas can be used easily to determine many envelope-related parameters. Baker and Steemers (2000) developed a manual design strategy suitable for the early design stage. Primary energy consumption is calculated as a function of several key parameters. The results are provided as curves to illustrate the relationship between the end-use energy consumption and the glazing area for a particular location, orientation and building type. For example, the curves in Figure 2.1 are for offices in northern UK with design illuminance and internal gains being 500 lux and 30 W/m^2, respectively. Because many parameters of the underlying models are fixed as defaults, the curves can be applicable only to those cases where those parameters take similar values.

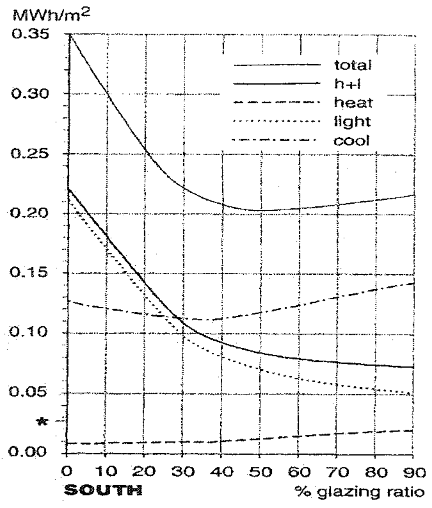


Figure 2.2 LT (Lighting and Thermal) curves for offices
(From Baker and Steemers 2000)

Many design guidelines help in achieving energy efficiency by recommending proper values for physical properties of building components (e.g., the thermal resistance value of walls). However, no suggestion is provided about the way to satisfy that requirement. For example, several insulation material alternatives having the same thermal resistance value may differ greatly in terms of environmental impacts from material production and construction process. With increasing importance attached to life-cycle analysis for buildings in recent years, some guidelines have been established in view of the environmental impacts of materials. Anink et al. (1996) ranked substitutable materials according to their environmental performance. Woolley et al. (1997) compared function-equivalent materials and products with respect to a number of environmental impact categories, based on which a few "best buys" are recommended. Although the suggested ranking or "best buys" are straightforward and can be conveniently used in practice, they might be misleading in the context of the whole building design because of an inappropriate comparison basis. For example, unit volume is the comparison basis for insulation materials in the study of Woolley et al. (1997). In this case, the ranking cannot be used directly to determine which material is better for a building design because thermal conductivity affects the required quantity of insulation materials. A building design using a low-ranked material may be more environment-friendly if that material has much lower thermal conductivity than the recommended best buy.

In many situations, a green building rating system and the prescriptive clauses of energy-related standards can act as design guidelines. Some popular green building rating systems are GBTool (Cole and Larsson 2002), LEED (U.S. Green Building Council 2003), and BREEAM (British Research Establishment 2003). Because these rating systems describe what objectives should be achieved for each criterion, they are useful to evaluate building performance after a building design is completed. However, no details are provided regarding the ways to achieve those objectives, which are actually needed to do green building design. Representative standards are the ASHRAE standard 90.1-1999 (ASHRAE 1999) and the model national energy code for buildings in Canada (National Research Council 1997). Prescriptive clauses of energy-related standards regulate what must be done. Because many features that affect energy use are not considered, the prescriptive approach is usually restrictive and conservative to achieve energy efficiency of buildings.

2.2.2 Simulation Based Trial-and-Error Method

With the increasing complexity of building design, simple empirical rules or guidelines are no longer sufficient on their own. To obtain accurate performance prediction, one must consider the interactions among building systems, the influence of outside weather conditions, the operation schedule of a building, and so on. In this respect, sophisticated building simulations are unique because of their ability to consider the subtle relationships between many performance-related features. According to the performance criterion they can evaluate, simulation programs for buildings operation can be classified into four different categories (Hien et al. 2000): energy analysis and HVAC system sizing, ventilation and indoor air quality, natural and electrical lighting, and acoustics. A website containing a directory of building simulation programs is maintained by the U.S. Department of Energy (2004b). Since energy simulation is used in this research, the following discussions focus on simulation programs for operating energy, embodied energy, and life-cycle energy consumption.

Many programs have been developed to calculate the operating energy consumption of a building. Steady state energy simulation programs based on the degree-day method and the bin method (ASHRAE 2001) are easy to use, but they cannot account for the effects of thermal mass, solar heating, and dynamic interactions among building components, all of which are important in green building design. In contrast to these simple programs, detailed energy simulations such as EnergyPlus, DOE, and BLAST (U.S. Department of Energy 2004b) can give more accurate energy consumption prediction by comprehensively modeling the interactions among building components and the impacts of instant changes in indoor and outdoor environment. Accordingly, the detailed energy simulations require much more detailed input data to evaluate a building design than the steady state energy simulations. Reviews about operating energy simulations are made by

Al-Rabghi and Hittle (2001), Hien et al. (2000), and Hong et al. (2000).

In recent years, some programs have been developed to calculate the embodied energy of a building, where the embodied energy is the energy consumed by all of the processes associated with the production of a building, including raw material acquisition, manufacturing, transportation, construction, maintenance, and reparation. These programs are rarely limited to calculate only embodied energy; instead, they are expanded to analyze a series of environmental impact categories using the life-cycle assessment methodology. Thus, they are called LCA (life-cycle assessment) tools in many cases. Menke et al. (1996) wrote an overview and evaluation of LCA tools, but many of them are not specific to building design. A comprehensive list and discussion of LCA tools for building design can be found at (Annex 31 2001; RMIT University 2004). According to the building typology, available life-cycle assessment programs for building design support can be grouped into product-oriented LCA tools such as BEES (Lippiatt 2002), system-oriented LCA tools such as Athena EIE (ATHENA 2003), and whole building-oriented LCA tools such as Eco-Quantum (IVAM 2004). All these LCA tools do not calculate operating energy consumption.

Some efforts have been made to develop simulation programs that integrate operating energy consumption and embodied environmental impacts together. The early design model developed by Yohanis and Norton (2000) considers operating energy, embodied energy, and capital cost as well. Envest 2 (British Research Establishment 2004) is a commercial software package to estimate the life-cycle environmental impacts of a building. Both the early design model and Envest 2 calculate the operating energy consumption with simplified methods. EQUER (Peuportier 2001) couples an LCA model with an operating energy simulation program to predict life-cycle environmental impacts. EEE is a prototype developed by Baouendi (2003) to calculate life-cycle energy consumption, greenhouse emissions, and life-cycle cost for houses. In this prototype, the operating energy is estimated by HOT2000 (2001), and the embodied energy of building materials in the database are from various sources.

Building energy simulations play an important role in green building design. They can predict the energy-related performance of a building design alternative, and they can also be used to study the impact of design parameters on the considered performance criterion. However, with simulation-based trial-and-error method, the designers must manually explore the design space for a better building design. After a design alternative is defined, simulation programs are applied to evaluate its performance. If the results are unsatisfactory, some design parameters are modified, and simulation programs are used again to get the performance for the modified alternative. This procedure is repeated until a satisfactory design is obtained. Because of the large design space, designers may explore only a few design alternatives after several trials. Many potentially better design alternatives are missed. Therefore, the simulation-based trial-and-error method is inefficient to explore the design space. In addition, since simulation programs require extensive data input, entering and modifying input data can be a time-consuming and error-prone process. Moreover, many required data are often unavailable until later design stage. This means that simulation programs are typically used when the design has become fixed and is difficult to alter. To address the above problems, simplified calculations are adopted in some programs such as Envest 2; default values for parameters known at detailed design stage are used for many other programs such as Energy-10 (U.S. National Renewable Energy Laboratory 2004) and Building Design Advisor (Papamichael 1999). However, simplified calculations decrease the accuracy of energy estimation, and default input values may not coincide with the final design.

2.2.3 Knowledge-Based Method

Knowledge-based systems (KBS) are computer programs that use knowledge represented in rules and facts to solve problems in a specific domain. Many expert systems have been prototyped in the field of HVAC systems and energy-efficient building design (Carnejo and Hittle 1989; Mayer et al. 1991; Qun 1999). Two approaches can be identified in previous KBS applications.

First, heuristics extracted from human experience are used to provide assistance to designers. Essentially, this method is an automated version of the first methodology (i.e., empirical rule or guideline based manual method) for green building design. A major problem associated with this method is that heuristics usually have narrow application scopes (Shaviv et al. 1996). Therefore, the KBS developed according to this approach is limited to small-scale qualitative-type problems such as HVAC system type selection for small buildings (Shams et al. 1994).

The second approach integrates heuristic knowledge and procedural simulation programs. This approach has been implemented in a number of knowledge-based systems (Athienitis and Akhniotis 1992; Robin et al. 1993; Shaviv et al. 1996). Heuristics guide the design process whenever they are adequate and applicable to the current design situation; otherwise, procedural simulation models are used. In a representative study by Shaviv et al. (1996), the heuristics are derived directly from a large number of simulation evaluations. Design parameters are grouped according to their impacts on energy performance. The heuristics work by setting reasonable default values or by suggesting a series of simulation runs. Compared with the first approach, simulation integrated KBS can effectively assist designers in energy-efficient building design because suggestions are provided in both qualitative and quantitative forms.

The main challenge for rule-based expert systems lies in the knowledge acquisition from human experts, literatures, or simulation runs. A recent artificial intelligence technique called case-based reasoning (CBR) overcomes the difficulty of knowledge acquisition by representing specific knowledge embedded in individual cases. Unlike rule-based KBS that starts reasoning from scratch each time, CBR starts by retrieving a similar previous case. Simulation is an integrated component in some CBR systems in order to verify the retrieved or adapted cases. For example, in a CBR system for building envelope design (Iliescu 2000), the energy-efficiency criterion is evaluated with a simulation program to check whether the design alternative agrees with the ASHRAE 90.1 standard.

A knowledge-based system addresses the drawback of intensive data input associated with procedural simulation methods. Given a little initial coarse information, a knowledge-based system can quickly lead to a good design solution, and it can also offer explanations regarding the reasoning process. However, as indicated by Kolodner (1993), heuristic methods may not find optimal solutions because of their weakness in exploration.

2.2.4 Optimization Method

Optimization method aims at finding optimal solutions with respect to some predefined performance criteria. Unlike many knowledge-based systems that set many simulation inputs as default values, optimization methods take the advantage of exploration to determine the optimal value for each variable. In essence, optimization is an automated process incorporating three steps: generation, simulation, and evaluation (Radford and Gero 1987).

There are many difficulties in applying optimization to building design. These difficulties can be seen in terms of the following aspects:

- There are complicated interactions among parameters, and there is more than one performance criterion to be considered. A reasonable system boundary must be defined to make the established optimization model operational without sacrificing too much the interrelationships among building elements.
- The performance criteria of a design alternative are usually evaluated by different commercial simulation programs, which are difficult to couple to an optimization program.
- Both continuous and discrete variables usually coexist in building design optimization models. Moreover, some variables may be interdependent with each other.
- The design space is typically large.

Despite these difficulties, many studies have proposed optimization models for building design. The next section will focus on previous optimization studies with respect to environmental performance.

2.3 Building Design Optimization Studies for Environmental Performance

This section intends to present a broad picture of former research on optimizing building environmental performance. Previous related studies are summarized in the first subsection. Then, limitations of previous studies are discussed.

2.3.1 Overview of Previous Related Optimization Studies

This subsection reviews only optimization studies for the environmental performance at the level of building systems. Component-based optimization studies such as overhang design (Chan et al. 1998) are not covered. Studies aiming at life-cycle cost minimization are included because operating energy cost is incorporated in the objective function. Based on the above-defined scopes, previous studies are reviewed, and they are listed in Table 2.2 in chronological order. Table 2.2 is organized as follows.

- The column titled "objective function" indicates what are the main aspects considered in terms of building energy consumption, e.g., heating, cooling, and lighting. Also, this column indicates whether the studies are single- or multiple-objective optimization (i.e., S and M).

- The column titled "variables" indicates the main parameters optimized and whether they are continuous or discrete variables (i.e., C and D).

- The columns titled "simulation program" and "optimization program" indicate whether commercially available software is used in the studies (i.e., Yes and No). When this information is not provided in the literature, "unknown" is indicated.

- The column of optimization technique states the underlying optimization methods used.

- Each optimization model is classified into one of three levels of generality: poor, medium, and good. If the research focuses on an individual building with little flexibility, the generality level is considered poor. If the optimization model is applicable only to buildings with a predefined number and type of variables, the level is considered medium. If designers are offered some freedom to change the optimization model, the generality level is regarded as good.

Table 2.2 Summary of previous optimization studies for environmental performance

Author	Year	Country	Objective Function		Variables		Simulation Program	Optimization program	Optimization Technique	Generality
Gupta	1970	Australia	Degree of thermal discomfort	S	Mainly envelope related	C	No	No	Direct search method	Medium
Wilson and Templeman	1976	U.K.	Life-cycle cost (heating)	S	Heating system, insulation	C	No	No	Geometric programming	Poor
Radford and Gero	1987	Australia	Thermal load, daylighting, construction cost, usable area	M	Mainly envelope related	D	No	No	Dynamic programming	Medium
Bouchlaghem & Letherman	1990	U.K.	Degree of thermal discomfort	S	Mainly envelope related	C	unknown	No	Direct search method	Medium
Johnson et al.	1990	Canada	Life-cycle cost (lighting + thermal)	S	Window	C	Yes	No	Feasible direction	Poor
Miller	1992	U.S.	Life-cycle cost (lighting + thermal)	S	Window, insulation, mechanical system type	C	Yes	No	Direct search, gradient-based method	Poor
Al-Hommoud	1994	U.S.	Operating energy (heating +cooling)	S	Mainly envelope related	C	Yes	No	Direct search method	Medium
El-Khawas	1997	U.S.	Life-cycle cost (lighting + thermal)	S	Wall/roof/window type, orientation, window fraction	C D	No	Yes	Branch and bound method	Good

Author	Year	Country	Objective Function		Variables		Simulation Program	Optimization program	Optimization Technique	Generality
Gustaffson	1998	Sweden	Life-cycle cost (heating)	S	Heating system, insulation, window type	C D	No	Yes	Mixed integer linear programming	Medium
Peippo et al.	1999	Finland	Life-cycle cost, operating energy (heating +lighting)	S	Envelope related, lighting, active solar related	C D	No	No	Direct search method	Medium
Hauglustaine and Azar	2001	Belgium	Regulation compliance, cost, energy	M	Envelope related	C	Unknown	No	Genetic algorithm	Good
Khajehpour	2001	Canada	Capital cost, operating cost, income revenue	M	Structure-related, envelope-related	C D	No	No	Genetic algorithm	Medium
Rudbeck et al.	2001	Denmark	Life-cycle cost	S	Mainly envelope related	C D	Unknown	Yes	Unknown	Medium
Saporito et al.	2001	U.K.	Operating energy (heating)	S	Mainly envelope related	C D	Yes	No	Lattice method	Medium
Wetter	2001	U.S.	Operating energy (heating+cooling+ lighting)	S	Orientation, window width, shading device transmittance	C	Yes	No	Direct search method	Good
Caldas and Norford	2002	U.S.	Operating energy (heating+cooling+ lighting)	S	Window place and size	D	Yes	No	Genetic algorithm	Medium
Coley and Schukat	2002	Germany	Annual energy consumption (heating+cooling)	S	Orientation, wall construction, window location and shading	C D	Yes	No	Genetic algorithm	Medium

Author	Year	Country	Objective Function		Variables		Simulation Program	Optimization program	Optimization Technique	Generality
Wright et al.	2002	U.K.	Operating energy cost, thermal comfort	M	HVAC system size and operation	C D	No	No	Genetic algorithm	Medium
Nielsen	2002	Denmark	Life-cycle cost	S	Envelope construction, heating and lighting system design	C D	No	No	Simulated annealing	Good
Mahdavi and Mahattanatawe	2003	Austria	Integrated performance for passive design	M	Envelope related	C	No	No	Direct search method	Medium
Nassif et al.	2003	Canada	Operating energy consumption, thermal comfort	M	HVAC system control	C	No	No	Genetic algorithm	Poor
Lu et al.	2005	Singapore	Operating energy consumption	S	HVAC system	C D	No	No	Genetic algorithm	Medium

Most previous optimization models are established as single-objective optimization problems with degree of discomfort, operating energy consumption, or life-cycle cost as the performance criterion.

- Thermal discomfort calculated as the mean absolute deviation from the comfort level is used as the objective function by Bouchlaghem and Letherman (1990) to optimize the building envelope. Such a study neglects the environmental impacts and cost associated with the additional use of materials by passive solar design strategies.

- Operating energy consumption is the optimization criterion in many studies (Al-Homoud 1997; Coley and Schukat 2002; Saporito 2001; Wetter 2001). It should be noted that the meaning of "operating energy" varies significantly with these studies. Only heating energy consumption is covered in the study of Saporito (2001) while both heating and cooling energy are covered in (Al-Homoud 1997; Coley and Schukat 2002). The study of Wetter (2001) enlarged the scope further to include lighting energy consumption in the optimization model. The energy end-use implied in the objective function has direct impacts on the selection of variables.

- Life-cycle cost is dealt as a performance criterion in several studies (El-Khawas 1997; Miller 1992; Nielsen 2002; Peippo et al. 1999). Compared with the other two objective function types, life-cycle cost has a distinctive advantage of balancing the trade-off relationship between initial construction and operating energy consumption in terms of cost. For example, more insulation usually decreases operating energy consumption cost but increases initial construction cost.

In practice, multiple criteria are usually considered simultaneously in the decision-making process of building design. Therefore, a proper multi-objective optimization model can offer designers more convenience than a single objective model. Radford and Gero (1987) applied dynamic programming to multi-criteria design optimization with the following four performance criteria: thermal load, daylight availability, construction cost, and usable area. Hauglustaine and Azar (2001) developed a tool that can optimize building envelope in the early design stage. As many as ten criteria related to regulation compliance, energy consumption, and cost are considered in that tool. The weighted sum method is used to solve the formulated multi-criteria problem. Users are required to enter expected values for each criterion to facilitate normalization. This cannot be easily done because a priori knowledge about the design is needed to input expected values for each criterion. It is also difficult for designers to understand the performance space because only one optimal solution is generated for each run. Mahdavi and Mahattanatawe (2003) also used the weighted sum method for multi-criteria optimization of passive solar building design with thermal comfort and daylighting quality as the two considered issues. Khajehpour (2001) applied a multi-objective genetic algorithm to the conceptual design of high-rise office buildings. Three objective functions considered in that study include capital cost minimization, annual operating cost minimization, and annual income revenue maximization. Because Khajehpour put emphasis on structural design, a simplified method is used to calculate energy consumption, and many important parameters on environmental performance such as insulation are not treated as variables. Wright et al. (2002) applied a multi-objective genetic algorithm to building thermal optimization with emphasis on mechanical system design, where operating energy cost and occupant thermal comfort are the two performance criteria. Nassif et al. (2003) also used a multi-objective genetic algorithm to optimize HVAC system control with operating energy consumption and thermal comfort as the two performance criteria.

2.3.2 Limitations of Previous Optimization Studies

Although the above optimization studies are significant to explore effective ways for better building design, several limitations may undermine their application in practice. They are discussed below.

Incomplete environmental performance criterion. Embodied energy and environmental impacts are neglected in all previous studies. This may be due to the following three reasons: (1) it is difficult to obtain consistent and accurate data about environmental impacts for all building materials and components; (2) building simulation programs for life-cycle environmental impact evaluation are not easily accessible; and (3) embodied energy and environmental impacts are not directly related to building construction costs, which is usually the major concern of most clients and designers. However, the optimal solution derived from operating energy consumption optimization may not be the desired one if the timescale is extended to cover the whole life-cycle of buildings. This is because the environmental burden can shift from the operating stage to upstream stages such as material production and building construction. As indicated by Yohanis and Norton (2002), increasing operating energy efficiency makes embodied energy consideration more important. The energy initially embodied in a building could be as much as 67% of its operating energy over a 25-year period for a generic single-story office building. Furthermore, energy consumption is no longer a complete criterion to evaluate environmental performance because many environmental impacts associated with material production are not energy-related. This makes it necessary to incorporate other impact categories such as natural resource depletion and global warming into the objective function.

Difficulty in making cost-effective decisions accounting for environmental performance. Most previous studies deal with either economical or environmental performance (Al-Homoud 1997; Coley and Schukat 2002; Miller 1992; Wetter 2001). In a few studies that have considered both economical and environmental performance criteria, two approaches were adopted: (1) one criterion is handled as a constraint (Nielsen 2002; Peippo et al. 1999); and (2) the weighted sum technique is used (Hauglustaine and Azar 2001). Both approaches require designers to provide a priori information such as boundary value for the constraint and weights for the performance criteria. With little knowledge about the performance space of the problem in advance, designers may find it difficult to set appropriate values for those required inputs. Furthermore, only one optimal solution is obtained for each run if the two performance criteria are treated separately or coupled together into one meta-criterion. The designer cannot learn the impact of the marginal change of one criterion on another just from a single optimal solution. Therefore, it is difficult to make cost-effective decisions without knowing the trade-off relationship between economical and environmental performance. Coley and Schukat (2002) realized the limitation of generating a single optimal solution;

therefore, they try to obtain a number of high-quality solutions distributing in the design space as wide as possible. At the end of the optimization, all obtained solutions are filtered to eliminate those alternatives with energy performance lower than a threshold value. Stacked histograms are visually presented to the designer in order to facilitate final selection after considering other important criteria not implemented in the optimization model.

Mismatch between optimization model and design practice in terms of variables. The mismatch between optimization model and design practice can be seen in two ways. In the first case, the variable type is not properly defined. Many parameters such as window types can only take discrete values. However, they are often defined as continuous variables because of the difficulty for numerical optimization methods to deal with discrete variables; for example, Miller (1992) represented the window type by its thermal resistance value. As a result, there may be no windows available in the market having the obtained optimal thermal resistance value. In the second case, some variables in the model are not directly design-oriented, but are intermediate results from other design calculations. For example, Al-Homoud (1997) gave an optimal value for time lag of walls, for which designers may find it difficult to map to a corresponding design solution. Mahdavi and Mahattanatawe (2003) realized the inconvenience brought to designers by the above problem. They carried out the optimization of enclosure design in two steps. The optimal values for physical properties of the enclosure are obtained first. Then, the actual building enclosure is determined by mapping the property values to the database of enclosure constructions.

Rigid system development. Generally, building optimization problems may vary in terms of the following aspects: (1) the number and types of variables; (2) the number of constraints; (3) the optimization model types (e.g., multi-objective or single objective, constrained or unconstrained); (4) building simulation programs used to evaluate objective functions or constraints; and (5) optimization algorithms. Previous studies have been limited to a fixed context in terms of the variables, the optimization models, and the simulation programs. Little attention has been paid to the generalized problem of simulation-based optimization for green building design. Thus, it is hard to extend and reuse the developed applications in other similar problems with different optimization environments. This may be the major reason that former developed optimization models are rarely used by designers in practice. In recognition of the disadvantages of the rigid system development, Wetter (2004) has developed a generic optimization tool, GenOpt, that can facilitate customization of the optimization environment by users. However, multi-objective optimization is not supported in the current version of GenOpt. This has made it inconvenient for green building design which requires multiple performance criteria be considered and the trade-off relationships between different criteria be investigated.

2.4 Summary

Optimization holds much potential for green building design because of its prominent advantages compared with other methods. The major advantages of optimization can be summarized as:

- It can find the optimal or near optimal solution for a predefined problem. Thus, optimization usually produces more effective design than experience or heuristics based methods.

- The searching process for optimal solution is automatically carried out by optimization algorithms. Therefore, it is more efficient than the conventional trial-and-error design procedure.

- It is possible to employ detailed simulation programs in the early design stage as long as those parameters available at the later design stage are dealt as variables.

Despite the above advantages, optimization is not a common design practice. This is mostly due to the following two reasons. First, many difficulties exist in establishing a model and solving the optimization problem. Second, designers find it cumbersome to use previously developed systems or programs for building optimization. This thesis intends to propose a solution to address the limitations of previous related studies. The findings are presented in the following chapters.

Chapter 3
Optimization Model Development

This chapter presents the underlying mathematical model for this life-cycle optimization research. Since a well-defined system boundary must be established to carry out an optimization study, the scope of this research is defined first. The second section introduces the expanded cumulative exergy consumption, which will be used as an objective function to express the life-cycle environmental impact. The optimization model is finally described with its variables, objective functions, and constraints.

3.1 Scope Definition

To define the research scope for green building optimization, the first thing is to identify the important factors that affect the environmental performance of a building. Baker and Steemers (2000) indicate that building design, service system design, and occupant behavior are three major factors upon which the energy performance depends. Their roles on energy performance are illustrated in Figure 3.1. This figure shows that by changing design parameters such as plan, orientation, and façades, a low-quality building design could consume energy about 2.5 times more than a high-quality design. The range can be extended to about 5 times if the service system parameters such as lighting efficiency and boiler efficiency are varied. The occupant factors can further bring the total range to about 10 times.

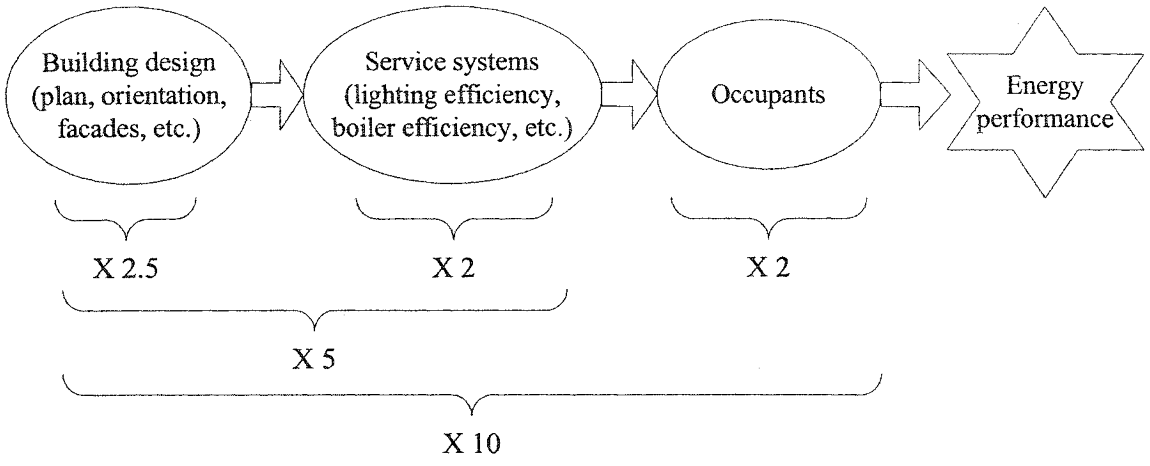

Building design (plan, orientation, facades, etc.)
Service systems (lighting efficiency, boiler efficiency, etc.)
Occupants
Energy performance
X 2.5
X 2
X 2
X 5
X 10

Figure 3.1 Factors and their roles on building energy performance
(Adapted from Baker and Steemers 2000)

Although Figure 3.1 was developed for energy performance, the same principle still applies if the consideration is expanded from energy consumption to environmental performance. Of the three factors, the building design is extremely important for the environmental performance of a building because (Baker and Steemers 2000): (1) building design related parameters are the least likely to be changed after its construction; and (2) strategies implemented in building design have impacts on service system performance and occupant behavior. Regarding building design, the envelope, which separates indoor from outdoor, plays the most important role to achieve the environmental performance. Therefore, this research selects building design, in particular, building envelope design, as the optimization scope.

Since LCA (life-cycle assessment) is used to evaluate the environmental performance of a building design, it is also necessary to define a clear scope for the LCA in this study. Of all the life-cycle stages shown in Figure 2.1, this study considers natural resource acquisition, building material production, on-site construction, operation, and transportation associated with the above stages. Demolition, maintenance and renovation are not included because of the following reasons:

- Some studies (Adalberth 1997; Chris et al. 2003; Cole and Kernan 1996; Seppo and Arpad 2003) revealed that material acquisition, production, on-site construction, and operation have a dominant proportion (more than 95%) of the life-cycle environmental impacts of a building.

- Demolition, maintenance and renovation cause much more uncertainties on LCA than other stages because: (1) the environmental impacts of demolition depend on the technology available at the end of a building's life that can lasts more than 50 years; and (2) the frequencies of maintenance and renovation are related to property management and functionality changes, which are user-dependent.

- Data about the environmental impacts due to the demolition, maintenance and renovation are unavailable for many building materials or assemblies. Although some LCA tools like Athena EIE (ATHENA 2003) have made some efforts in this respect, lack of reliable data is still a serious problem.

LCA assumes that waste emission and its environmental impact has linear relationship without threshold values (Barnthouse et al. 1998). This assumption determines that the global, continental, and long-lasting impact categories (e.g., global warming) can be dealt with by LCA with acceptable theoretical accuracy, but there is increasing loss of accuracy as more local and transient impact categories (e.g., ecotoxicity) are considered in the aggregated LCA indicators. The impact categories that can influence the environment on a large scale and in the long-term include resource depletion, global warming, acidification, and ozone depletion (Barnthouse et al. 1998). Because the optimization focuses on the building envelope, ozone depletion gases due to refrigerant leaks are not considered. Thus, the impact categories considered in this optimization study include resource depletion, global warming, and acidification, while other impacts characterized as being local and transient such as ecotoxicity and photochemical smog are not considered. Each category incorporates a number of inventory items. For this research, resource depletion focuses on fossil fuels and minerals; global warming considers three major greenhouse gases (CO_2, CH_4, N_2O); acidification considers two major acidic gases (SO_x and NO_x).

It can be noticed that the considered impact categories have different units and magnitudes. For example, the energy consumption and SO_x emissions of a building over its life may have dimensions of 10^7 MJ and 10 kg, respectively. A number of ways have been suggested in the literature to integrate various impact categories together to produce one-number indicator as shown in Table 2.1. In this research, the expanded cumulative exergy consumption is used as the indicator to evaluate the life-cycle environmental impact.

3.2 Expanded Cumulative Exergy Consumption

This section first gives a brief introduction about the concept of exergy, which will be used in one of the objective functions in this study. Then, available methods that incorporate exergy into the life-cycle assessment to measure resource depletion and waste emissions are presented.

3.2.1 Exergy

Exergy is "the amount of work obtainable when some matter is brought to the state of thermodynamic equilibrium with the common components of the natural surroundings by means of reversible processes, involving interaction only with the above-mentioned components of nature" (Szargut et al. 1988). Unlike energy, exergy is always destroyed because of the irreversible nature of the process. Exergy is an extensive property whose value is fixed by the state of the system once the environment has been specified. Therefore, the evaluation of exergy depends on both the state of a system under study and the conditions of the reference environment. Several reference environment models have been summarized and explained by Rosen and Dincer (1997). For example, in the reference-substance model proposed by Szargut et al. (1988), reference species is selected for each element. These reference species are defined as common components of the natural environment which is divided into atmospheric air, seawater, and the earth's crust for consideration. The exergy value can be expressed as the sum of two parts: thermomechanical exergy and chemical exergy, and the former can be further divided into three components: physical exergy, kinetic exergy and potential exergy. The specific exergy (e) can be calculated as follows (Moran and Shapiro 1994):

$$e = \underbrace{(u-u_0)+p_0(\upsilon-\upsilon_0)-T_0(s-s_0)}_{\text{physical}} + \underbrace{v^2/2}_{\text{kinetic}} + \underbrace{gz}_{\text{potential}} + \underbrace{e^{CH}}_{\text{chemical}} \qquad (3\text{-}1)$$

where, u, υ, and s represents the specific internal energy, volume, and entropy, respectively. The subscript 0 stands for the reference state. The items $v^2/2$ and gz are specific kinetic energy and gravitational potential energy, respectively.

Compared with energy, exergy has the advantage of being able to evaluate the quality of different energy sources. For example, the quality index of electricity and district heating are 100% and 30%, respectively (Wall 1977). The comparison of energy and exergy is summarized in Table 3.1 (Dincer and Cengel 2001; Szargut et al. 1988).

Table 3.1 Comparison of energy and exergy

Energy	**Exergy**
• Satisfies the law of conservation • Is a property parameter that depends only on the state of the matter under consideration • Is a measure of quantity only	• Does not satisfy the law of conservation • Is a property parameter that depends on two states: (i) the matter under consideration, and (ii) the matter of the reference environment • Is a measure of quantity and quality • Measures the ability to do work in the future • A measure of the potential output from a product to the environment

3.2.2 Cumulative Exergy Consumption (CExC) as a Measure of Resource Depletion

Resource depletion can cause the gradual degradation of natural resources, e.g., the low-concentration of mineral ores and the exhaustion of economically exploitable oil reserves. Because more energy is needed to concentrate the lower-grade ores and to extract deeper-covered fuels, more exergy is consumed in the production process of metals and fuels. This will consequently cause an increase in the total exergy consumption required by any other products with either metals or fuels as inputs.

Cumulative exergy consumption (CExC) expresses the sum of the exergy of all natural resources that are consumed in all the steps of a production process (Szargut et al. 1988). Unlike cumulative energy consumption, CExC takes into account the exergy consumption of not only energy sources but also the non-fuel resources extracted from the environment. Three methods are suggested to calculate cumulative exergy consumption (Szargut et al. 1988): (i) the method of process analysis; (ii) the method of balances of cumulative exergy consumption; and (iii) the method based on values of cumulative energy consumption. They are briefly explained below because of the importance of CExC for this research.

Method of Process Analysis

The idea underlying the method of process analysis is to trace the production processes of a product and divide them into different levels (Szargut et al. 1988). The first level comprises only the final step of the chain of production processes. The immediate consumption of fuels, non-fuel resources, and intermediate products are evaluated at this level. The second level incorporates the evaluation of extraction, transportation and storage of the natural materials used in the first level, the immediate consumptions of fuels and non-fuel resources for the intermediate products of the first level. In the third level, the production of machines for level 1 and the extraction, transportation and storage of the natural materials used in level 2 are considered. Additional levels may be required for complex products. More levels means that more processes are traced and that the accuracy of estimating the CExC may be improved accordingly. However, according to Szargut et al. (1988), the values derived from the first two levels account for 90-95 percent of the entire cumulative exergy consumption.

Method of Balances of CExC

The method of balances of CExC needs to establish a group of equations. Each equation expresses the balance of CExC with respect to a kind of natural resource and a relevant product. For example, the balance of CExC for the *k-th* natural resource and the final product *j* can be expressed as Equation 3-2 (Szargut et al. 1988). The left side of this equation indicates the outputs of a process where the *k-th* natural resource goes, whereas the right side indicates the inputs where the *k-th* natural resource comes.

$$\mathrm{CExC_{kj}} + \sum_{i}(\mathrm{CExC_{ki}} \cdot \mathrm{f_{ij}}) = \sum_{i}(\mathrm{CExC_{ki}} \cdot \mathrm{a_{ij}}) + \mathrm{B_{kj}} \qquad (3\text{-}2)$$

where,

$CExC_{kj}$ and $CExC_{ki}$: cumulative exergy consumption of the *k-th* natural resource per unit of the final product *j* and the intermediate product *i*;

f_{ij} and a_{ij}: production and consumption of the *i-th* intermediate product associated with unit product *j*;

B_{kj}: immediate exergy consumption of the *k-th* natural resource for unit product *j*.

Similarly, an equation can be established for each intermediate product *i* and the *k-th* natural resource. All these equations are linked together and solved to obtain the value of $\mathrm{CExC_{kj}}$. The same methodology is followed to derive the CExC of other natural resources. Finally, the CExC of the final product *j* ($\mathrm{CExC_j}$) can be obtained with the following formula:

$$\mathrm{CExC}_j = \sum_k \mathrm{CExC}_{kj} \tag{3-3}$$

Method Based on Values of Cumulative Energy Consumption

The basic idea of the method based on values of cumulative energy consumption is to derive the CExC from the value of cumulative energy consumption. This can be reasonably done because of two reasons:

- First, the ratio between fuel exergy (chemical exergy) and fuel energy is known. For example, the ratio between chemical exergy and gross calorific value for bituminous coal, lignite, fuel oil, and gasoline are 1.03, 1.04, 0.99 and 0.99, respectively (Szargut et al. 1988). Moreover, there is minor difference in the ratio between exergy and energy for different fuels. Therefore, based on the cumulative energy consumption of a product, the cumulative exergy from fuel consumption can be obtained with acceptable accuracy even if the energy mix is unknown.

- Second, for most products, the dominant portion of cumulative exergy consumption comes from fuel exergy consumption. As shown in Table 3.2, the cumulative exergy consumption of non-fuel materials is mostly due to the fuel exergy used in the extraction or purification process, while the exergy from non-fuel resources (e.g. mineral ores) is small (about 5-12% of CExC). Hence, the exergy consumption of non-fuel resources can be reasonably estimated.

Table 3.2 Cumulative exergy consumption for some common materials

Material	**CExC (MJ/kg)**		
	From non-fuel resources	**From fuel**	**Total**
Steel (from ore)	4.9	41.0	45.9
Copper	7.5	59.5	67
Aluminium (from Bauxite in the ground)	2.0	30.5	32.5
Cement from raw materials	0.35	5.83	6.18

Data source: Szargut et al. (1988)

Of the three available methods for the calculation of CExC, the first method is time-consuming because it requires detailed process analysis for each product; the second method has a major drawback of data unavailability because the coefficients such as f_{ij} and a_{ij} in the formula 3-2 are unknown for many products; the last method is simple to use because the embodied energy for building materials can be conveniently obtained from many sources such as databooks and life-cycle assessment tools. Therefore, the method based on cumulative energy consumption is employed in this research to calculate CExC. An improvement regarding the application of the chosen method in this study is to calculate, not to roughly estimate the exergy consumption of non-fuel resources. The calculation is based on the chemical exergy and the quantity of each non-fuel resource, as will be presented in detail in Section 3.4.

3.2.3 Abatement Exergy Consumption (AbatEx) as a Measure of Waste Emissions

Exergy can not only be used to measure resource consumption; it can also measure waste emissions. Because exergy can evaluate the degree of disequilibrium between a substance and its environment, a rational and meaningful relationship can be established between the environmental impact potentials and the exergy of waste emissions (Ayres et al. 1998, Rosen and Dincer 1999). Three methods using exergy as a measurement of waste emissions are discussed below.

Direct Measurement Method

The direct measurement method expresses the harmful effects of waste emissions directly in terms of their exergy. However, as indicated by Szargut et al. (1988), such an assumption seems to be oversimplifying. First, regarding waste emissions, substance with more exergy does not necessarily have more serious environmental impacts. Second, the exergy of waste emissions cannot be added with cumulative exergy consumption from resource inputs because their implications are different: the exergy of natural resources is useful, but the exergy of waste emissions is usually not useful. Thus, with the direct measurement method, two separate indicators are still required to evaluate natural resource depletion and waste emissions.

Ecological Cost Coefficient Method

The ecological cost coefficient method was developed by Szargut et al. (1988). This method investigates the effects of waste emissions on the natural resources and man-made products. Regarding the effects of waste emissions, a couple of examples are the corrosion of buildings and the death of trees caused by acid rains due to SO_x emissions. Natural resource coefficient and man-made product coefficient are the two kinds of coefficients required to evaluate the ecological costs due to waste emissions. The major obstacle behind this method lies in the difficulty of determining the above coefficients. With so many man-made products, resources, and waste emissions, it is almost impossible to get all the required coefficients.

Abatement Exergy Method

In the abatement exergy method, the impact of emissions is evaluated by the exergy required to remove or isolate the emissions from the environment. For example, the abatement exergy of NO_x was estimated to be 16 MJ/kg, based on the technology involving the chemical reaction between NO_x and NH_3 (Cornelissen 1997). Although the value of the abatement exergy for one type of waste emission depends on the technology used, the abatement exergy method is employed in this research because it has the following advantages:

- This method can be easily applied once the abatement exergy is known for each waste emission.
- Abatement exergy data for some waste emissions are available in the literature. For example, the abatement exergy for CO_2 and SO_x are 5.86 and 57 MJ/kg, respectively (Dewulf et al. 2001).
- Abatement exergy can be summed directly with cumulative exergy consumption because abatement exergy essentially represents the cumulative exergy consumption for a process to remove or isolate an emission from the environment.

3.2.4 Expanded CExC as a Measure of Resource Depletion and Waste Emissions

The expanded CExC is calculated as the sum of cumulative exergy consumption (CExC) and abatement exergy consumption. It considers both resource inputs and waste emissions to the environment over all life-cycle stages. It can be regarded as a unifying indicator to evaluate life-cycle environmental impacts. The term "expanded cumulative exergy consumption" is used mainly due to the following two considerations: (1) the coverage is expanded from the general CExC, estimated for resource depletion, to include the abatement exergy consumption, estimated for waste emissions; and (2) abatement exergy is essentially the cumulative exergy consumption associated with the operations to remove or isolate waste emissions. Although the term is new, the idea behind expanded CExC has been observed in a few previous studies. Zhang (1995) combined the resource exergy consumption and the exergy needed for recovering the total equivalent CO_2, to screen several energy conversion systems. They used the term "total equivalent resource exergy" instead of expanded cumulative exergy consumption. Dewulf et al. (2001) applied the idea of expanded CExC to assess the sustainability of four waste gas treatment systems. The main advantages of using expanded CExC for life-cycle optimization of buildings with respect to environmental performance can be summarized as:

- It incorporates fuel and non-fuel resources together to characterize the resource depletion.

- It combines resource depletion and waste emissions together; therefore, the life-cycle environmental impacts can be condensed into one single objective function. Moreover, it can avoid the subjectivity associated with weights setting in some previous integration methods as shown in Table 2.1.

- It facilitates the future enlargement of the scope to include the design of mechanical systems. Exergy-related indicators such as exergetic efficiency are important to investigate and assess the use of low valued and environmentally friendly energy sources for the mechanical system of a building (Moran 1982; Annex 37 2003). Since the same concept of exergy is used for the design of buildings and mechanical systems, it is possible to use a single indicator (e.g., exergetic efficiency) to evaluate the environmental performance of a whole building design.

The expanded CExC will be used as one of the objective functions in the optimization model discussed below.

3.3 Design Variables

A variable is a design parameter that can be changed during the optimization process to find its most favorable value for the defined objective functions. Major considerations in selecting variables from the list of design parameters are discussed first. Based on these considerations, all variables in the optimization model are presented. The relationships among variables are discussed at the end of this section because some variables are interrelated.

3.3.1 Major Considerations in Variable Selection

Two types of variables are used to define a building design: discrete and continuous. The design parameters that can have discrete values only must be defined as discrete variables, whereas the parameters that can take continuous values may be defined as either continuous or discrete variables. For example, the type of windows must be defined as a discrete variable with a list of available window products; orientation may be a continuous variable ranged in between 0 and 90 degrees, or it may be a discrete variable with a pre-set list of values such as 0, 15, or 30 degree.

For a simulation-based optimization study, the following factors need to be considered in the selection of variables:

- The predefined research scope. Since building envelope is the major scope of this optimization research, most variables are envelope-related parameters; a few others are structure-related and overhang-related.

- The impact of design parameters on the performance criteria. It is important to set those sensitive parameters that can significantly influence the considered performance as variables. The sensitive parameters can be determined from experience, guidelines, and sensitivity analysis.

- The ease of use of optimal variable values in design practice. It is preferable to select those parameters that are directly design-oriented as variables; otherwise, designers may find it difficult to figure out the corresponding design alternative for given optimal values. If the thermal resistance of walls is a variable, for example, given its optimal value, designers still do not know the corresponding wall configuration since there are many ways to achieve the same thermal resistance. However, the above problem can be addressed by using each layer as a variable.

- The degree of a designer's control on parameters. Usually, only the parameters that can be controlled by designers are potential variables, while those parameters beyond the control of designers are fixed for an optimization environment. For example, the discount rate is a parameter indicating the rate of return on the next investment opportunity available for a developer. Although it affects the life-cycle cost, it is not treated as a variable since its value mostly depends on the national financial policy, not on the designers.

- Simulation programs. Variables must be from those parameters that can be modeled in the simulation programs used to evaluate the performance criteria. For example, if the EnergyPlus program (U.S. Department of Energy 2003) is used to estimate energy consumption, the orientation can be handled as a variable. However, if a program based on the degree-day method is used, it is impossible to set orientation as a variable because that method does not consider the impact of orientation on energy consumption.

- Optimization algorithms. Both the number and the type of variables may be affected by the chosen optimization algorithm. For example, a gradient-based optimization algorithm cannot easily handle discrete variables, thus limiting the scope to those design parameters with continuous values.

With the above factors considered, the variables in the optimization model are defined and described in the following subsection.

3.3.2 Variable Description

It is assumed in this research that the building has only one floor with known area and height. This assumption is thus made because the simulation program to be discussed in the next chapter is applicable for one single zone with similar thermal conditions and loads. The variables considered in this study are categorized into several groups: building geometry-related variables, structure-related variables, envelope configuration-related variables, and overhang-related variables. Each variable is presented with a brief description, and the variable name in parenthesis is used later.

Geometry-related variables

Geometry-related variables are those parameters that can change a building in terms of orientation, shape, length, and width. The following variables are used:

- Building orientation (*orientation*): the angle between the true north and the building north. The true north is the direction indicated by a compass while the building north is the direction relative to one side of a building, as indicated by the designer. Orientation is measured in degrees with clockwise being positive. It is a variable that can be either continuous or discrete.

- Building shape (*shape*): a discrete variable that defines the shape of the building plan. This research considers two alternatives: rectangular shape and L-shape.

- Aspect ratio (r_0): the ratio of a to b, where a and b are the length and width of the rectangle as shown in Figure 3.2. Aspect ratio is used for both of the two considered building shapes. It is a variable that can be either continuous or discrete.

- Length ratio (r_1): the ratio of a_1 to a for side3 (Figure 3.2). Length ratio is used only for an L-shape building. This variable can be either continuous or discrete.

- Width ratio (r_2): the ratio of b_1 to b for side2 (Figure 3.2). Width ratio is also used only for an L-shape building and can be either continuous or discrete.

- Tilt of wall for each side i ($wallTilt_i$, i=1,2,3,4): the angle in degrees between the outward normal of the ground and of the wall surface, as shown in Figure 3.3. This variable can be either continuous or discrete.

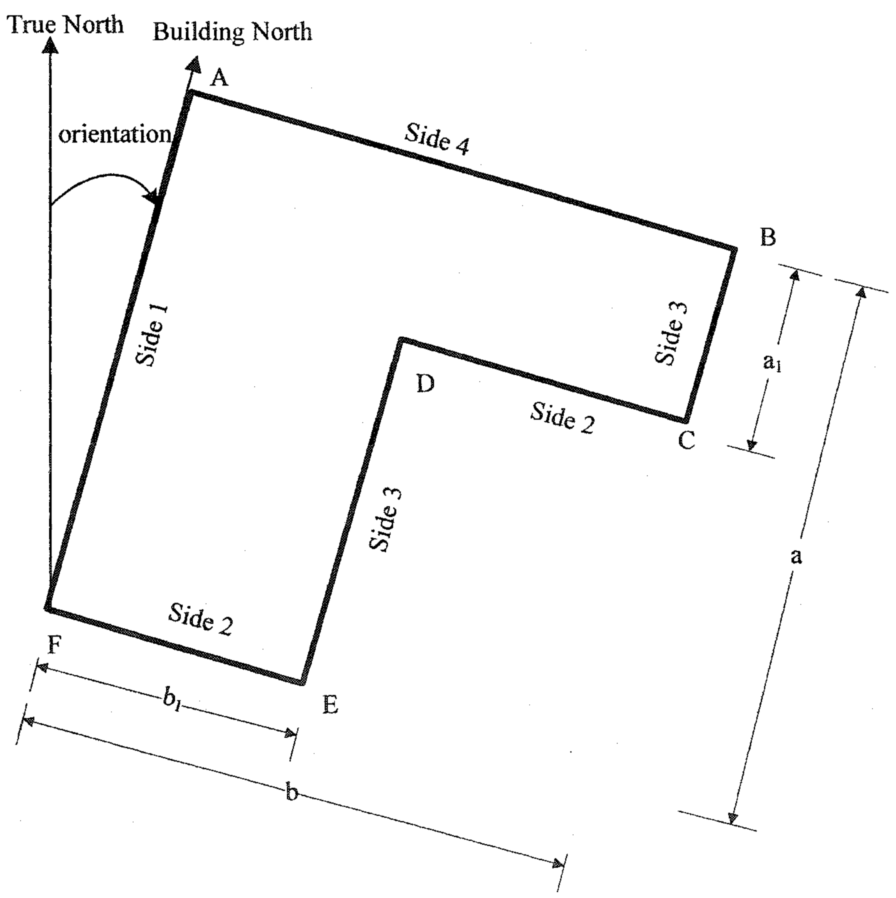


Figure 3.2 Illustration of the shape of building plan and related variables

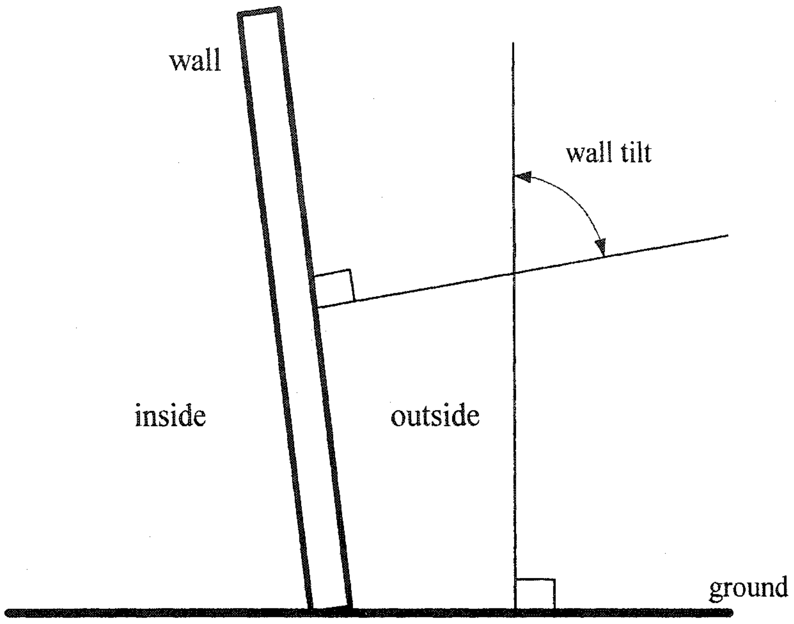


Figure 3.3 Illustration of wall tilt

Structure-related variables

Structural design is not in the scope of this research. However, the structure has an important impact on the building envelope design because of their close relationships. For example, the structural system determines the applicable wall types, and the wall may be an integrated part of the structural system. Therefore, this study considers the following structure-related variables: the structural system and the structural layers of opaque building envelope elements such as walls and roofs. The structural layers will be discussed as envelope configuration-related variables.

- Structural system (*strSystem*): different alternatives for the building structural system (e.g., steel frame vs. concrete frame). The purpose of this variable is to ensure the compatibility between walls, roofs, floors and overhangs.

__Envelope configuration-related variables__

The building envelope system can be divided into opaque walls, floors, roofs, and windows. A wall can be decomposed into a number of successive layers such as cladding, insulation, and sheathing. Both the sequence and material types of wall layers depend on the wall type. In addition to dealing with each layer as a discrete variable, it would be advantageous to deal with the wall type as another discrete variable to consider alternative wall constructions simultaneously. The same principle applies to roof type and roof layers, floor type and floor layers. This research includes the following envelope configuration-related variables:

- Window design switch (*winSwitch*): a binary variable that determines whether the window type can vary with different facades. If the switch is on, the same window type must be used for all four facades. If the switch is off, different window types can be used on the four façades.

- Window type for each side *i* ($winType_i$, i=1,2,3,4): a discrete variable defining the different window types (e.g., uncoated single glazing and low-e double glazing) available for each façade. Because window frames and mullions are not considered in this study, window types differ with glazing only.

- Window fraction for each side *i* ($winRatio_i$, i=1,2,3,4): the ratio of window area to the wall area for each façade. This variable can be either continuous or discrete.

- Wall type for each considered structural system ($wallType_k$): a discrete variable defining the different wall types (e.g., masonry cavity wall and steel-frame wall) suitable for a given structural system design, where, *k* represents the *k-th* structural system design.

- Each layer of each wall type (*wallLayer*$_{ij}$): a discrete variable that is used to define the possible alternatives (e.g., 50 mm expanded polystyrene and 15 mm mineral wool) for a given layer (e.g., insulation) of each wall type, where, *i* and *j* represents the *i-th* layer and the *j-th* wall type, respectively.

- Roof type for each considered structural system (*roofType*$_k$): a discrete variable defining the different roof types (e.g., conventional compact roof type and inverse roof type) suitable for a given structural system design, where, *k* represents the *k-th* structural system design.

- Each layer of each roof type (*roofLayer*$_{ij}$): a discrete variable that is used to define the possible alternatives (e.g., concrete roof deck and steel deck) for a given layer (e.g., roof deck) of each roof type, where, *i* and *j* represents the *i-th* layer and the *j-th* roof type, respectively.

- Floor type for each considered structural system (*floorType*$_k$): a discrete variable defining the different floor types (e.g., hard floor and resilient floor) suitable for a given structural system design, where, *k* represents the *k-th* structural system design.

- Each layer of each floor type (*floorLayer*$_{ij}$): a discrete variable that is used to define the possible alternatives (e.g., tile and wood) for a given layers (e.g., finish) for each floor type, where, *i* and *j* represents the *i-th* layer and the *j-th* floor type, respectively.

<u>Overhang-related variables</u>

An overhang is a passive solar design device above windows to reduce the impact of direct solar radiation on a building in summer. The overhang design is closely related to the window below it. An overhang design can be determined with the following parameters: depth, height, left extension, right extension, and thickness, all of which are illustrated in Figure 3.4. Of the above five parameters, left extension and right extension are related to the window layouts. In this study, continuous windows and overhangs are considered on each façade. In other words, the overhang width is the same as the window width, which is set equal to the length of the corresponding wall. Hence, both left extension and right extension are zero. The thickness is regarded as a fixed design parameter because (1) its values can vary within only a small range for a given overhang type; and (2) it does not have a major impact on the sun-shading effect. Thus, the following overhang-related variables are defined:

- Overhang type on each side i ($overhangType_i$, i=1,2,3,4): a discrete variable that indicates the possible overhang types (e.g., aluminum overhang and concrete overhang) on a façade. No overhang is also regarded as an alternative. The structural system affects the applicable overhang types. For example, wood overhangs are suitable for a building with wood structure, but concrete overhangs are not suitable in that case. In addition, the use of overhang may be favorable for some facades but not for others because of the close relationship between the orientation and the sun-shading effect of overhangs.

- Overhang depth on each side *i* (*overhangDepth*$_i$, i=1,2,3,4): the distance in meters between the wall and the outer edge of the overhang, as shown in Figure 3.4. This variable can be either continuous or discrete.
- Overhang height on each side *i* (*overhangHeight*$_i$, i=1,2,3,4): the distance in meters from the top of the window to the overhang, as shown in Figure 3.4. This variable can be either continuous or discrete.

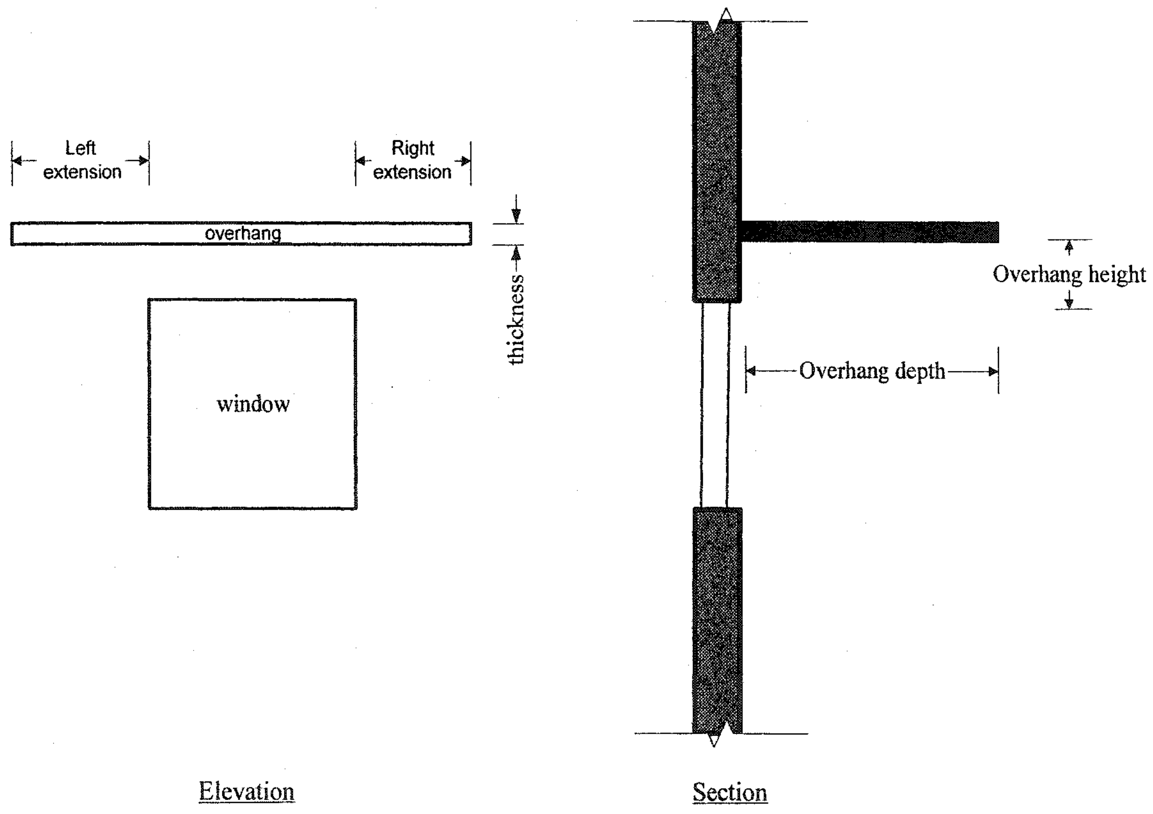


Figure 3.4 Illustration of overhang related design parameters

3.3.3 Structured Variable

Not all the variables described in the previous subsection are used to determine a building design. Some variables such as *orientation* and *winRatio* on each side are always active because their values are necessary to define each building design alternative. However, some variables may be inactive in certain situations. The status of being active or not depends on some other variables. For example, both wall types and wall layers are defined as discrete variables. Different wall types have different compositions in terms of the sequence, the number, or the functionalities of wall layers such as cladding and insulation. Suppose that the variable *wallType* has several discrete values corresponding to several possible wall types. Since each wall type contains a group of layers, several groups of wall layers will appear in the variable list accordingly. In any case, only one wall type and thereby one group of wall layers is used in a building design, while all other groups of wall layers are inactive. Which group of wall layers is used depends on the value of *wallType*. In other words, the variable *wallType* acts as a controller to determine which wall layers are active and which are inactive. In this case, *wallType* is called a structured variable in this research.

A structured variable has the following characteristics: (1) it controls several other variables because the structured variable is at a higher level and the variables under control are at a lower level; (2) it can take discrete values only; (3) it may be controlled by another structured variable, which leads to the existence of multi-level structured variables. For example, *wallType* is a structured variable controlled by *strSystem*.

Structured variables are usually used in one of the following two situations:

- If the hierarchical relationship exists between the entities represented by variables, the variables in the higher-level hierarchies are treated as structured variables. For example, the wall type and wall layers are in different levels of the building hierarchy, so the wall type is a structured variable.

- If the combination of variable values could result in an incompatible design, structured variables may be used to address the above problem. For example, suppose that open web steel joist (OWSJ) with steel deck system and cast-in-place (CIP) concrete flat plate system are two possible types for both roof and floor. Without the structured variable *strSystem*, an incompatible design in terms of structure could be produced with OWSJ floor and CIP roof. This example indicates that the structured variable *strSystem* is necessary in this study.

Since the use of structured variables allows the coexistence of active variables and inactive variable, an enlarged design space can be explored at the early stage of building design. For example, without structured variables, wall type can no longer be dealt with as a variable. This means that only one wall type can be specified for an optimization problem. As a result, several similar optimization problems must be solved if there are more than one wall types available.

3.3.4 Summary of Variables

In summary, all variables in the optimization model are listed in Table 3.3. The characters 'C', 'D', and 'S' in the column "variable type" represent continuous variable, discrete variable, and structured variable, respectively. For each sub-level variable, the column "dependency" indicates the structured variable that has direct control on it.

Because the alternatives (i.e., discrete values) of the structural system has not been defined, its sub-level variables are shown only once in Table 3.3. However, it should be noted that the sub-level variables such as *wallType* and *roofType* can repeat for each alternative of the structural system. The same principle applies to *wallLayer*, *roofLayer*, and *floorLayer*. Chapter 7 gives an example to show how to set values for the variables in Table 3.3.

Table 3.3 Variable list in the optimization model

Group	Variable Name		Variable Type	Dependency
Geometry-related	orientation		C, D	
	shape		S	
	shape=Rectangular	r_0	C, D	shape
	shape= 'L'	r_0	C, D	shape
		r_1	C, D	shape
		r_2	C, D	shape
	wallTilt1		C, D	
	wallTilt2		C, D	
	wallTilt3		C, D	
	wallTilt4		C, D	
Structure-related	strSystem		S	
Envelope configuration-related	winSwitch		S	
	winSwitch=on	winType	D	winSwitch
	winSwitch=off	winType_1	D	winSwitch
		winType_2	D	winSwitch
		winType_3	D	winSwitch
		winType_4	D	winSwitch
	winRatio_1		C, D	
	winRatio_2		C, D	
	winRatio_3		C, D	
	winRatio_4		C, D	
	wallType		S	strSystem
	wallLayer		D	wallType
	roofType		S	strSystem
	roofLayer		D	roofType
	floorType		S	strSystem
	floorLayer		D	floorType
Overhang-related	overhangType_1		D	strSystem
	overhangType_2		D	strSystem
	overhangType_3		D	strSystem
	overhangType_4		D	strSystem
	overhangDepth_1		C, D	
	overhangHeight_1		C, D	
	overhangDepth_2		C, D	
	overhangHeight_2		C, D	
	overhangDepth_3		C, D	
	overhangHeight_3		C, D	
	overhangDepth_4		C, D	
	overhangHeight_4		C, D	

3.4 Objective Functions

An objective function is a quantified performance criterion. The two criteria considered in this research are environmental performance and economical performance. These two performance criteria are evaluated respectively with the following two major objective functions: life-cycle environmental impact and life-cycle cost.

When considering both environmental and economical performance of a building design, the monetary integration approach presented in Chapter 2 could be employed to incorporate the environmental issues into the life-cycle cost via environmental taxes or the estimated damage cost due to waste emissions. Although the monetary approach has the advantage of translating the complicated environmental impacts into a single familiar dimension: money, it has several limitations: (1) unlike the conventional life-cycle cost that uses the discount rate to consider the time value of money, environmental impacts should not be simply discounted (Gluch and Baumann 2004); (2) environmental taxes do not reflect the reality of environmental impacts because taxes are usually stipulated with many other considerations such as politics and the economic policy; and (3) designers cannot understand the conflicting nature and the trade-off relationship between the environmental and the economical performance if a single objective function is used.

Since using the life-cycle cost approach to account for environmental issues have the above limitations, this study prefers to deal with life-cycle environmental impact and life-cycle cost separately.

3.4.1 Life-Cycle Environmental Impact

As mentioned in the second section, expanded cumulative exergy consumption is used as the indicator to evaluate the environmental impact of a building. Let **X** denote the vector of variables used in the optimization problem. The general formula to calculate life-cycle environmental impact (LCEI) is expressed as:

$$\begin{aligned} \mathrm{LCEI(X)} &= \mathrm{EE(X)} + \mathrm{OE(X)} \\ &= [\mathrm{CExC_{PS}(X)} + \mathrm{AbatEx_{PS}(X)}] + [\mathrm{CExC_{OS}(X)} + \mathrm{AbatEx_{OS}(X)}] \end{aligned} \tag{3-4}$$

where,

EE: environmental impact (MJ) in the pre-operation stage, also called embodied environmental impact;

OE: environmental impact (MJ) in the operation stage, also called operating environmental impact;

CExC: cumulative exergy consumption (MJ);

AbatEx: abatement exergy consumption (MJ);

PS: subscript representing the pre-operation stage including natural resource extraction, building material production, on-site construction, and transportation associated with the above stages;

OS: subscript representing the operation stage.

Using the method based on values of cumulative energy consumption, as discussed in section 3.2, the cumulative exergy consumption in the pre-operation stage ($\mathrm{CExC_{ps}}$) and in the operation stage ($\mathrm{CExC_{os}}$) can be calculated, respectively, with the following two formulas:

$$CExC_{PS} = \sum_{j}(\alpha_j \cdot EN_j) + \sum_{k}(e_k \cdot m_k) \tag{3-5}$$

$$CExC_{OS} = n \cdot \sum_{j}(\alpha_j \cdot \frac{ON_j}{\eta_j}) \tag{3-6}$$

where,

EN_j: embodied energy (MJ) due to the use of fuel j, consumed in the pre-operation stage of a building;

ON_j: annual on-site building operating energy consumption (MJ/yr) of fuel j;

m_k: mass (kg) of non-fuel resource k used for building construction;

e_k: chemical exergy (MJ/kg) of non-fuel resource k;

n: study period (yr) of life-cycle analysis;

α_j: ratio between the chemical exergy and the energy content of fuel j;

η_j: overall exergetic efficiency of the production and delivery for fuel j, used to convert from on-site operating energy sources to primary energy sources.

The abatement exergy consumption (AbatEx) is calculated as the product of mass of waste emissions and its unit abatement exergy. The following formula can be used to calculate the abatement exergy consumption for both the pre-operation stage and the operation stage.

$$\text{AbatEx} = \sum_{w} e_w \cdot m_w \tag{3-7}$$

where,

e_w: unit abatement exergy (MJ/kg) of the waste emission w;

m_w: mass (kg) of the waste emission w.

In the above formulas (3-5 to 3-7), n is a user defined parameter. The values for e_k, α_j, η_j, and e_w can be found in various references (Dewulf et al. 2001; Szargut et al. 1988; Zhang 1995). For example, the chemical exergy of gypsum is 50 kJ/kg (Szargut et al. 1988), and the overall efficiency of production and delivery for natural gas is 0.88 (Zhang 1995). For each building element such as a wall and a roof, the embodied energy EN_j due to the use of fuel j, the mass of non-fuel resource m_k, and the mass of waste emission m_w are calculated as the product between the area of the building element and their specific values per unit construction area. The specific values for embodied energy, mass of non-fuel resources, and mass of waste emissions can be obtained from life-cycle assessment programs. For example, the embodied energy for a 100 m^2 area of a 39*89@400 wood-stud wall construction in Montreal can be obtained with the Athena EIE life-cycle assessment program (ATHENA 2003), and its values are 578, 558, 20, 565, 496, 116, 23, 79, 115 MJ, respectively for hydro-electricity, diesel, gasoline, natural gas, wood, coal, heavy fuel oil, nuclear, and feedstock fuels. The annual on-site operating energy consumption ON_j includes the energy for heating, cooling, lighting, and equipment. Since this study considers electricity as the only energy source in the operation stage, the subscript j for annual on-site operating energy consumption is removed in the following discussions. Thus, *ON* is represented as:

$$\mathrm{ON} = \mathrm{ON}_{\mathrm{heating}} + \mathrm{ON}_{\mathrm{cooling}} + \mathrm{ON}_{\mathrm{lighting\,\&\,equipment}} \tag{3-8}$$

The energy consumption for heating and cooling are derived from the space conditioning load, which is defined as the required rate of heat addition or extraction to maintain a specified space temperature profile. The energy consumption for lighting and equipment is derived by multiplying the corresponding power density and the floor area, which are user inputs. In mathematical forms, the three items in Formula 3-8 can be expressed as:

$$\mathrm{ON}_{\mathrm{heating}} = 0.108 * \sum_{\mathrm{month}=1}^{m_{\mathrm{heating}}} \sum_{\mathrm{hour}=1}^{24} \frac{\langle Q_{\mathrm{sys}} \rangle}{\mathrm{PI}_{\mathrm{heating}}} \tag{3-9}$$

$$\mathrm{ON}_{\mathrm{cooling}} = 0.108 * \sum_{\mathrm{month}=1}^{m_{\mathrm{cooling}}} \sum_{\mathrm{hour}=1}^{24} \frac{\langle -Q_{\mathrm{sys}} \rangle}{\mathrm{PI}_{\mathrm{cooling}}} \tag{3-10}$$

$$\mathrm{ON}_{\mathrm{lighting\,\&\,equipment}} = 0.108 * S * \sum_{\mathrm{month}=1}^{(m_{\mathrm{heating}} + m_{\mathrm{cooling}})} \sum_{\mathrm{hour}=1}^{24} (\mathrm{LPD} + \mathrm{EPD}) \tag{3-11}$$

where,

$m_{heating}$, $m_{cooling}$: number of months in the heating season and the cooling season, respectively;

$PI_{heating}$, $PI_{cooling}$: performance index for heating and cooling, respectively. The performance index might be the coefficient of performance (COP) for heat pumps or the efficiency ratio for other systems;

Q_{sys}: load (W) for heating or cooling;

S: total floor area (m^2);

LPD: lighting power density (W/m^2);

EPD: equipment power density (W/m^2).

The constant value of 0.108 takes into account the number of days per month and the conversion from W·h to MJ; that is, $0.108 = 30 * 3.6/1000$. Because of the conventions used in the calculations, the load takes negative values for cooling and positive values for heating. Cooling hours and heating hours may coexist in a given day. Since the cooling hours in the heating season and the heating hours in the cooling season are not counted, the bracket operator < > is used in Formulas 3-9 and 3-10. The bracket operator returns the value of the operand if the operand is positive; otherwise, it returns zero.

The heat balance method (ASHRAE 2001) is used to calculate the heating and cooling load Q_{sys}. According to the principle of energy conservation, a group of equations are established for exterior zone surfaces, interior zone surfaces, and the air in the zone. These equations are solved simultaneously at every hour for a specified day. In addition, the same calculation is iterated for that specified day until the load Q_{sys} is converged.

Based on the zone air heat balance, Equation 3-12 is used to calculate the heating and cooling load Q_{sys}. The equation accounts for the infiltration, the convection from interior surfaces and internal sources, and the HVAC system.

$$Q_{sys} = \sum_i \left[A_i \cdot h_{ci} \cdot (T_a - T_{si,i})\right] - Q_{conv,S} - \rho_{moistAir} \cdot (\frac{ACH \cdot V}{3600}) \cdot (h_o - h_a) \qquad (3\text{-}12)$$

where,

A: area (m^2) of the inside surface;

h_{ci}: convective heat transfer coefficient (W/m^{2}K) on the inside surface;

T: temperature (K);

$Q_{conv,S}$: total convection heat (W) from internal sources, i.e., people, lights, and equipment;

$\rho_{moistAir}$: density (kg/m^3) of outdoor moist air;

ACH: air change per hour (hr^{-1}), which is used to estimate the natural air infiltration;

V: zone volume (m^3);

h: specific enthalpy (J/kg) of air;

a, o: subscripts representing indoor air and outdoor air, respectively;

si: subscript representing inside surface;

i: subscript representing the building elements (e.g., walls and roofs) enclosing the zone.

In Equation 3-12, the inside surface area A and the zone volume V are automatically derived from the building definition; the indoor air temperature T_a and the infiltration rate ACH are user inputs; the total convection heat flux from the internal sources $Q_{conv,S}$ is calculated directly from the internal heat gains and their proportions of convection components; the density $\rho_{moistAir}$ and the enthalpy h_o of outdoor air are estimated in terms of the outdoor air conditions (i.e., dry-bulb and wet-bulb temperatures); the enthalpy of indoor air h_a is calculated in terms of T_a and the humidity ratio of infiltration; the interior convection heat transfer coefficient h_{ci} is calculated in this study using ASHRAE constants (Pedersen et al. 2000); the inside surface temperature T_{si} is solved by combining the equations for the outside surface heat balance and the inside surface heat balance, as discussed below.

The heat balance on the outside surface of the *i-th* building element is expressed in Equation 3-13 and accounts for the solar radiation ($q_{\alpha sol}$), the longwave radiation exchange with the environment (q_{LWR}), the exterior convection ($q_{conv,so}$), and the heat conduction to the interior (q_{ko}). The heat balance on the inside surface of the *i-th* building element is expressed in Equation 3-14 and accounts for the radiative gains from internal sources and transmitted solar radiation ($q_{radDist}$), the radiation exchange between internal surfaces (q_{LWX}), the interior convection ($q_{conv,si}$), and the heat conduction from the exterior (q_{ki}). All the items in Equations 3-13 and 3-14 take the unit of W/m^2.

$$q_{\alpha sol,i} + q_{LWR,i} + q_{conv,so,i} - q_{ko,i} = 0 \tag{3-13}$$

$$q_{radDist,i} + q_{LWX,i} + q_{conv,si,i} + q_{ki,i} = 0 \tag{3-14}$$

The solar radiation flux $q_{\alpha sol}$ is the absorbed solar radiation for opaque surface, and it is the outward flowing fraction of the absorbed solar radiation for windows. The absorbed solar radiation is the product between the absorptance of the outside surface and the total solar irradiance, which is the sum of the solar beam radiation, the diffuse sky radiation, and the solar radiation reflected from the ground. The net long wavelength radiation flux q_{LWR} includes the radiation exchange with the sky, the ground surface, and other surrounding surfaces. It can be calculated with Equation 3-15 below. The radiative flux $q_{radDist}$ is calculated by distributing the total radiation to the inside surface in a userprescribed manner. The total radiation includes the solar transmitted through windows, the longwave and shortwave radiation from internal sources. If the inside surface is glass, the calculation of $q_{radDist}$ should exclude the outward flowing fraction of the distributed shortwave radiation. The net longwave radiation flux exchange q_{LWX} is calculated using the model of mean radiant temperature with balance (Pedersen et al. 2000). This model assumes that the radiant interchange occurs between each surface with a fictitious surface. The net radiant flux for each surface can be expressed as Equation 3-16.

$$q_{LWR,i} = \sigma\varepsilon_{so,i}[F_{o,i}(T_o^4 - T_{so,i}^4) + F_{sky,i}(T_{sky}^4 - T_{so,i}^4) + F_{g,i}(T_g^4 - T_{so,i}^4)] \tag{3-15}$$

$$q_{LWX,i} = \sigma F_{MRT,i}(T_{si,i}^4 - T_{MRT,i}^4) \tag{3-16}$$

where,

σ: Stefan-Boltzmann constant (=5.67*10^{-8} W/m^{2}K^4);

ϵ: long-wave emittance;

T: temperature (K);

F: view factor or gray interchange factor between the inside surface and the fictitious surface;

o, sky, g, MRT: subscripts representing the outdoor air, the sky, the ground, and the fictitious surface, respectively;

so, si: subscripts representing the outside surface and the inside surface, respectively.

The convective heat flux $q_{conv,so}$ on the outside surface is calculated with Equation 3-17, where h_{co} is the exterior convection heat transfer coefficient in W/m^{2}K and calculated with the DOE-2 model (Pedersen et al. 2000). The convective heat flux $q_{conv,si}$ on the inside surface is calculated with Equation 3-18, where h_{ci} is the interior convection heat transfer coefficient in W/m^{2}K and uses the ASHRAE constants according to the tilt angle of each surface (Pedersen et al. 2000).

$$q_{conv,so,i} = h_{co,i}(T_{out} - T_{so,i}) \tag{3-17}$$

$$q_{conv,si,i} = h_{ci,i}(T_a - T_{si,i}) \tag{3-18}$$

The conductive heat fluxes q_{ko} and q_{ki} in Equations 3-13 and 3-14 are calculated using conduction transfer functions (CTFs), which relate conductive heat fluxes to the current and past surface temperatures and the past heat fluxes at discrete time series (ASHRAE 2001). Equations 3-19 and 3-20 are the general form of conduction transfer functions for the outside heat flux and inside heat flux, respectively.

$$q_{ko,i}(t) = -Y_{i,0}T_{si,i,t} - \sum_{j=1}^{nz}(Y_{i,j}T_{si,i,t-j\delta}) + X_{i,0}T_{so,i,t} + \sum_{j=1}^{nz}(X_{i,j}T_{so,i,t-j\delta}) + \sum_{j=1}^{nq}(\Phi_{i,j}q_{ko,i,t-j\delta}) \tag{3-19}$$

$$q_{ki,i}(t) = -Z_{i,0}T_{si,i,t} - \sum_{j=1}^{nz}(Z_{i,j}T_{si,i,t-j\delta}) + Y_{i,0}T_{so,i,t} + \sum_{j=1}^{nz}(Y_{i,j}T_{so,i,t-j\delta}) + \sum_{j=1}^{nq}(\Phi_{i,j}q_{ki,i,t-j\delta}) \tag{3-20}$$

where,

X: outside CTF coefficient;

Y: cross CTF coefficient;

Z: inside CTF coefficient;

Φ: flux CTF coefficient;

q_{ko}, q_{ki}: conduction heat flux (W/m^2) on outside surface and inside surface, respectively;

t, j: subscripts representing time;

δ: time step;

i: subscript representing a building element such as a wall and a roof.

The CTF coefficients (X, Y, Z, and Φ) and the summation limits (nz and nq) in the above two equations are dependent on the construction of building elements. Their values are calculated once for each building element. With these CTF coefficients, transient heat conduction can be modeled simply and solved easily with a linear equation.

3.4.2 Life-Cycle Cost

Life-cycle cost (LCC) analysis is a method to assess the total cost over the study period for a given building or system. Equation 3-21 (Ruegg and Marshall 1990) shows that LCC includes the present value of investment costs, energy costs, non-fuel operating and maintenance costs, repair and replacement costs, and residual values.

$$LCC = I_p + E_p + M_p + R_p - S_p \tag{3-21}$$

where,

I: investment costs (\$);

E: Energy costs (\$);

M: non-fuel operating and maintenance costs (\$);

R: repair and replacement costs (\$);

S: residual values (\$);

p: subscript representing present value.

There are substantial data required to carry out a detailed LCC analysis. Some simplifications are made according to the following principles: (1) the cost items that are unaffected by design variables are excluded in the calculation of LCC because they have equal values for all design alternatives compared; (2) the cost items beyond the scope of this research are excluded in the calculation; and (3) costs and values related to comfort and productivity are excluded because they are difficult to acquire and quantify. Based on these principles, the following simplifications are employed to calculate the LCC:

- Only construction cost is considered in investment costs. Costs for design, land acquisition and other pre-construction activities are not included because they are regarded to have the same values for all design alternatives. Furthermore, the considered construction costs are limited to the research scope, including exterior walls, roofs, floors, windows, and overhangs.

- Non-fuel operating costs such as water costs are excluded because this research does not consider water efficiency.

- Maintenance, replacement, and repair costs are not considered because of several reasons: (i) maintenance, replacement, and repair are out of the scope defined for the life-cycle assessment in this study; and (ii) it is difficult to acquire the replacement and repair cost with various envelope components.

- Residual values are not considered because (i) they are difficult to be estimated; and (ii) they do not have large differences for the design alternatives of a small simple building after taking into account the disposal costs and the long study period.

Therefore, based on the above simplifications, the LCC is formulated as Equation 3-22. Although the simplified LCC does not include all the cost items as required by strict LCC analysis, the term "LCC" is still used in this study because this is a convention in optimization studies focusing on energy performance of building design (El-Khawas 1997; Miller 1992; Nielsen 2002; Peippo et al. 1999).

$$\mathrm{LCC}(X) = \mathrm{IC}(X) + \mathrm{OC}(X) \tag{3-22}$$

where,

X: the vector of design variables considered in this study;

IC: the initial construction cost (\$) of a building including exterior walls, windows, the roof, the floor, and overhangs;

OC: the life-cycle operating cost (\$) due to energy consumption.

The life-cycle operating cost is the present worth of energy costs over the study period, accounting for the impact of inflation and energy price escalation. The following equation (National Research Council 1997) is used to calculate the life-cycle operating cost:

$$\mathrm{OC(X)} = \mathrm{AC(X)} \cdot \frac{1+r}{i-r} \cdot \left[1 - \left(\frac{1+i}{1+r}\right)^{-n}\right] \tag{3-23}$$

where,

AC: annual energy cost ($) in the first year;

i: discount rate, including inflation;

r: energy price escalation rate, including inflation;

n: study period (yr).

The annual operating cost consists of demand cost and energy consumption cost. For each billing period (i.e., a month), the demand cost is calculated based on the maximum power demand (kW) and the applicable demand rate ($/kW); the energy consumption cost is calculated based on the amount of energy consumption (kWh) and the applicable energy consumption rate ($/kWh). Since demand cost and energy consumption cost are calculated in the same way, their calculations are illustrated here using demand cost as an example. Suppose that the demand rate structure is $dr_1, dr_2, \ldots, dr_k$ ($/kW) with the corresponding demand threshold values as $dt_1, dt_2, \ldots, dt_k$ (kW), where dt_k is infinity. This rate structure implies that for a given peak demand *PD* (kW), dr_1 applies to the part less than dt_1, dt_2 applies to the part between dt_1 and dt_2, and so on. Thus, the demand cost *DC* ($) can be calculated with the following formula:

$$\mathrm{DC} = \sum_{i=1}^{j-1}\left[(\mathrm{dt}_i - \mathrm{dt}_{i-1}) \cdot \mathrm{dr}_i\right] + (\mathrm{PD} - \mathrm{dt}_j) \cdot \mathrm{dr}_j \tag{3-24}$$

where, *i* is the index that satisfies $\mathrm{dt}_{i-1} < \mathrm{PD} \le \mathrm{dt}_i$, in which $dt_0=0$.

The initial construction cost *IC* includes the construction cost for exterior walls, windows, roofs, floors, and overhangs. This can be expressed as:

$$\text{IC} = \text{wallCost} + \text{windowCost} + \text{roofCost} + \text{floorCost} + \text{overhangCost} \qquad (3\text{-}25)$$

The cost of each envelope component is obtained by multiplying its area and unit cost. For example, the wall cost is calculated as:

$$\text{wallCost} = \text{wallArea} * \text{wallUnitCost} \qquad (3\text{-}26)$$

The unit cost of walls, roofs, and floors is calculated simply as the sum of unit costs for all constituent layers. The unit cost for each layer construction, windows, and overhangs can be obtained from available cost-estimating guides such as the R.S. Means cost data books (2004a, 2004b). Regarding the area for each envelope component, the floor area is a user defined constant; the areas of other envelope components are calculated as follows.

The surface areas of exterior walls and windows are calculated with the following two equations:

$$\text{wallArea}_i = \text{facadeArea}_i \cdot (1 - \text{winRatio}_i) \qquad (3\text{-}27)$$

$$\text{winArea}_i = \text{facadeArea}_i \cdot \text{winRatio}_i \qquad (3\text{-}28)$$

where,

facadeArea: gross area (m^2) of the façade including opaque wall and windows;

winRatio: window ratio as defined in Section 3.3;

i: subscript representing the indices of façades as shown in Figure 3.2.

The surface area of roof needs to account for the protrusion or contraction due to the tilts of walls. The calculation of roof area depends on the building shape, as can be seen from the following formula:

$$\mathrm{roofArea} = \begin{cases} ab - b\cdot h\cdot \sum\limits_{i=2,4} \dfrac{1}{\tan\beta_i} - a\cdot h\cdot \sum\limits_{i=1,3} \dfrac{1}{\tan\beta_i} + h^2 \cdot \sum\limits_{i,j=1;i\neq j}^{4} \dfrac{1}{\tan\beta_i\cdot\tan\beta_j} & (\text{rectangular shape}) \\ a_1 b + a b_1 - a_1 b_1 - b\cdot h\cdot \sum\limits_{i=2,4} \dfrac{1}{\tan\beta_i} - a\cdot h\cdot \sum\limits_{i=1,3} \dfrac{1}{\tan\beta_i} + h^2 \cdot \sum\limits_{i,j=1;i\neq j}^{4} \dfrac{1}{\tan\beta_i\cdot\tan\beta_j} & (\text{L-shape}) \end{cases}$$

(3-29)

where,

a, b: building length and width (m) (see Figure 3.2);

a_1, b_1: width (m) of the wings of the L-shape building (see Figure 3.2);

h: building height (m);

β: wall tilt (degree) (see Figure 3.3);

i, j: subscripts representing the indices of façades (see Figure 3.2).

The area of overhangs is the total protruded overhang area on the four facades, and it is calculated as:

$$\mathrm{overhangArea} = 2\left(a\cdot \sum_{i=1,3} \mathrm{overhangDepth}_i + b\cdot \sum_{i=2,4} \mathrm{overhangDepth}_i \right) \tag{3-30}$$

where, *a*, *b*, and the subscript *i* have the same meanings as those in Equation 3-29;

overhangDepth is the distance between the wall and the outer edge of the overhang, as defined in Section 3.3. The value of *overhangDepth* is zero if overhang is not used on a façade.

The façade area used in Equations 3-27 and 3-28 is related to the tilts of walls, the building height, the length and the width of the building plan. The area of each façade is calculated as:

$$\text{facadeArea}_i = \begin{cases} \left[a - \frac{h}{2} \cdot \left(\frac{1}{\tan\beta_2} + \frac{1}{\tan\beta_4} \right)\right] \cdot \frac{h}{\sin\beta_i} & (i = 1,3) \\ \left[b - \frac{h}{2} \cdot \left(\frac{1}{\tan\beta_1} + \frac{1}{\tan\beta_3} \right)\right] \cdot \frac{h}{\sin\beta_i} & (i = 2,4) \end{cases} \tag{3-31}$$

The building length *a* and width *b* in Equations 3-29, 3-30, and 3-31 are shape-dependent. They are calculated with the following two equations:

$$a = \begin{cases} \sqrt{S \cdot r_0} & (\text{rectangular shape}) \\ \sqrt{\frac{S \cdot r_0}{r_1 + r_2 - r_1 \cdot r_2}} & (\text{L} - \text{shape}) \end{cases} \tag{3-32}$$

$$b = \begin{cases} \sqrt{S / r_0} & (\text{rectangular shape}) \\ \sqrt{\frac{S}{r_0 \cdot (r_1 + r_2 - r_1 \cdot r_2)}} & (\text{L} - \text{shape}) \end{cases} \tag{3-33}$$

where,

S: floor area (m^2), a user defined constant;

r_0, r_1, r_2: shape related variables as defined in the third section.

3.4.3 Other Objective Functions

In addition to LCEI and LCC, this research also considers several other objective functions, namely, initial cost (IC), operating cost (OC), annual operating energy consumption (AnnualEnergy), and life-cycle energy (LCEnergy). These functions can expand the application scope of this simulation-based optimization system because: (1) they are widely used in practice; and (2) they require less input data than the previous two major objective functions.

The initial cost and operating cost were discussed in the previous subsection. Since electricity is the only fuel type considered in this study, the annual operating energy consumption is equal to *ON* in Equation 3-8. The life-cycle energy is the sum of embodied energy and operating energy, and it is calculated with the following equation:

$$\text{LCEnergy} = \sum_j \text{EN}_j + n \cdot \frac{\text{ON}}{\eta'} \qquad (3\text{-}34)$$

where,

EN_j: embodied energy (MJ) due to the use of fuel j;

n: study period (yr) for life-cycle analysis;

ON: on-site electricity consumption (MJ/yr);

η': overall energetic efficiency of production and delivery of electricity.

3.5 Constraints

Constraints stipulate the conditions that need to be satisfied by relevant variables. According to its relationship with variables, a constraint can relate to a single variable or more than one variable. A constraint related to a single variable may be a box constraint for a continuous variable or a selection constraint for a discrete variable. A constraint involving more than one variable is called a functional constraint, which can be linear or nonlinear, and equivalent or nonequivalent.

- Box constraints indicate intervals for continuous variables with boundary values. For example, if the window ratio on side1 is set as a continuous variable and the lower and upper boundary values are 0.2 and 0.7, respectively, the box constraint corresponding to that variable is defined as:

  $$0.2 \le \text{winRatio}_1 \le 0.7$$

- Selection constraints are applicable to discrete variables. A selection constraint requires that the value of a discrete variable must be chosen from a number of predefined alternatives. For example, if the window type is limited to three available alternatives: uncoated double glazing (*winType1*), Low-e double glazing (*winType2*), and uncoated triple glazing (*winType3*), the corresponding selection constraint is defined as:

  $$\text{winType} \in \{\text{winType1}, \text{winType2}, \text{winType3}\}$$

- Functional constraints establish the relationships for more than one design variable. They are usually set according to technical specifications, design code, and the requirement of developers. This research supports all the functional constraints that can be converted from objective functions. Therefore, the functional constraints could be IC (initial cost), OC (operating cost), LCC (life-cycle cost), AnnualEnergy (annual operating energy consumption), LCEnergy (life-cycle energy), and LCEI (life-cycle environmental impact). The mathematical form of a functional constraint can be easily expressed once the right hand value is specified. For example, given the construction cost budget for a building design, the corresponding functional constraint can be defined as:

$$IC(x) \leq Budget$$

3.6 Summary

This optimization research focuses on the building envelope, which plays a critical role in determining the environmental performance and economical performance of a green building design. The scope of the life-cycle assessment is expanded from the operation stage only as considered in previous studies, to all stages except maintenance, renovation, and demolition in the current study. The impact categories are expanded from energy consumption only in previous studies, to include non-fuel natural resources such as mineral ore, and air emissions with global, continental, and long-lasting impacts on the environment. Expanded cumulative exergy consumption is used to unify all the considered impacts and to facilitate solving the optimization problem.

Variables are envelope-related design parameters. The concept of structured variable is introduced to represent the hierarchical relationship between variables. Constraint types can be box constraint for continuous variables, selection constraint for discrete variables, and functional constraints related to more than one variable. The two major objective functions considered in this study are life-cycle environmental impact and life-cycle cost.

With variables, constraints, and objective functions, an integrated optimization model can be established. Depending on the availability of functional constraints and the number of objective functions, different optimization model types can be developed as follows:

- Single objective optimization without functional constraints
- Single objective optimization with functional constraints
- Multi-objective optimization without functional constraints
- Multi-objective optimization with functional constraints

Since the optimization model may vary with different design situations, it is important to develop a system that could consider all of them. This objective has been kept in mind during the whole process of system analysis, design, and implementation, as will be discussed in the following chapters.

Chapter 4
System Formulation

This chapter presents the components of the simulation-based optimization system GBOptimizer. The four components include the simulation programs, the data files, the optimizer, and the input and output. The overall system framework is presented in the first section to show the interrelationships among those components. Then, each component is presented sequentially.

4.1 System Framework

The workings of simulation-based optimization systems require the collaboration of four components as illustrated in Figure 4.1. The four components play different roles: the input-output communicates with users; the optimizer provides optimization algorithms; the simulation programs evaluate objective functions and functional constraints; and the data files store the data required by the simulation programs. These components are closely connected. Before starting the optimization engine, a user usually needs to refer to data files to customize an optimization problem by defining parameters for the optimizer and the simulation programs through the input-output component. After the optimization engine is initiated, a close interaction exists between the optimizer and the simulation programs. The optimizer transfers variable values to the simulation programs. Using these values, the simulation programs evaluate objective functions and functional constraints and then return the results to the optimizer. During the simulation process, the simulation programs may frequently access data files to define the entity represented by a variable.

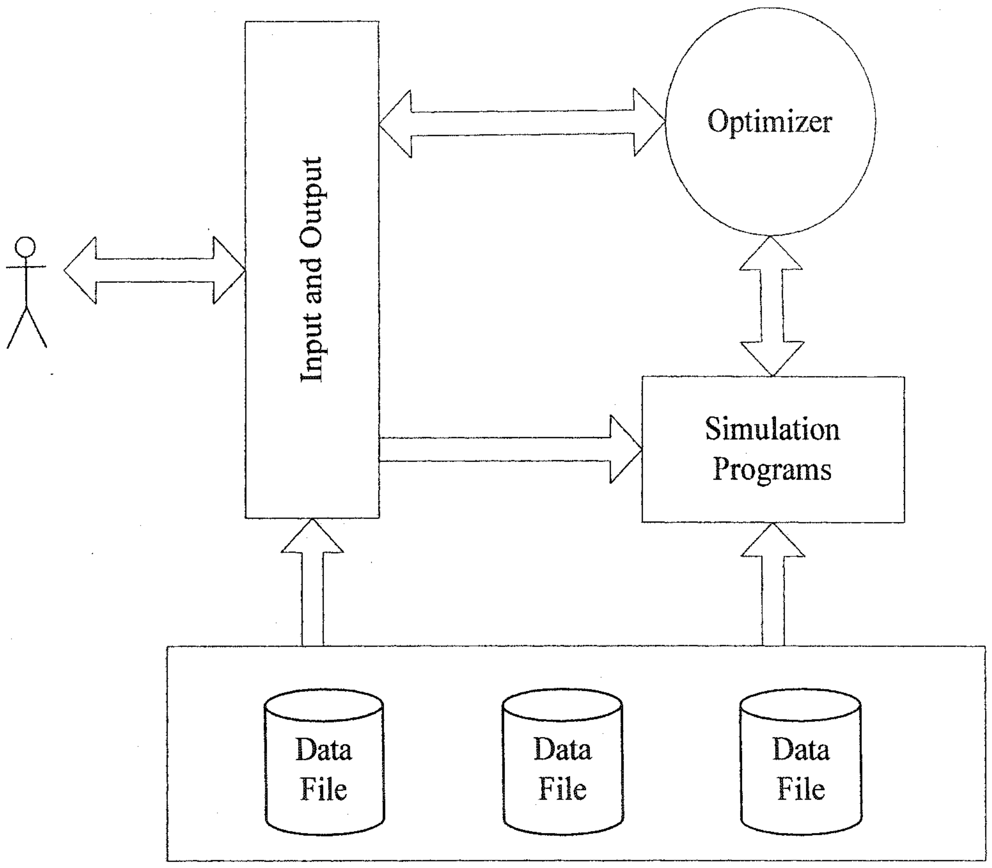


Figure 4.1 Architecture of the simulation-based optimization system GBOptimizer

In a simulation-based optimization system, there are two types of simulation program: external simulation and internal simulation. An external simulation does not compile together with the optimizer, whereas an internal simulation does. External and internal simulation programs have different dialog mechanisms with the optimizer. This leads to the following three possible interfaces between optimization and simulation programs:

- External interface. With an external interface, external simulations are used, and they communicate with the optimizer through files. This means that a translator (Riche et al. 2003) is usually required to fulfill two tasks: (1) to write the input text file for the external simulation programs according to the variable values obtained from the optimizer; and (2) to read the output file produced by the external simulation programs and return the required values to the optimizer. The translator can write input text files from scratch or through a template mechanism as employed by Wetter (2004). The external interface is used when the optimizer and the simulation are separated from each other. This can be the case if a commercial simulation program or a commercial optimization package is employed to solve an optimization problem.

- Internal interface. With an internal interface, internal simulation programs are used, and they communicate with the optimizer directly by the values of variables and functions. Compared with the external interface, a particular advantage of the internal interface is that the variables are allocated in computer memory once and shared by the simulation and the optimizer. Therefore, for a given optimization problem, the internal interface consumes less computation resources than the external interface. However, the system developed with the internal interface limits its application scope to the pre-programmed list of variables, performance criteria, and optimization approaches (Le Riche et al. 2003).

- Hybrid interface. With a hybrid interface, both external and internal simulations are employed to compute function values required by the optimizer. The internal simulation can be isolated from the external simulation or extended from the latter by post-processing the results of the external simulation. The hybrid interface requires that external and internal interfaces should be established respectively for the two types of simulation program. A hybrid interface inherits the characteristics from both external and internal interfaces.

All the above three interfaces are common to simulation-based optimization for building design because users have different accessibilities to the source code of building simulation programs. Therefore, the system should support all three interfaces.

4.2 Simulation Programs

Simulation programs evaluate objective functions and functional constraints in the optimization model. Although simulations usually refer to stochastic programs (e.g., a discrete-event simulation program using Monte Carlo methods) in the area of simulation-based optimization (Fu 2002; Gosavi 2003), here they cover deterministic computer programs that are commonly used in building simulations. This section first gives a brief introduction about the ASHRAE toolkit for building load calculations (Pedersen et al. 2000), a commercial simulation program employed in this research. Then, the extension work to derive the objective function values from the outputs of the ASHRAE toolkit is explained.

4.2.1 ASHRAE Toolkit for Building Load Calculations

As reviewed in Chapter 2, numerous energy simulation programs have been developed. The tradeoff between simplicity and accuracy must be considered when selecting a simulation program. The ASHRAE toolkit for building load calculations is chosen because it has the following advantages:

- The toolkit has no formal user interface. Both the input and output are text files. In this respect, it greatly facilitates the coupling with other programs. For example, the results written in a text file by the toolkit can be conveniently processed to get the intended function values.

- The toolkit has accessible source code. Therefore, the underlying calculation procedure can be inspected. Moreover, the modularized implementation makes it easier for users to develop their own code for specific requirements.

- The toolkit calculates building loads with the heat balance method, which is the recommended method by ASHRAE (2001).

- The toolkit facilitates expanding the research scope to include the HVAC system in the future. The HVAC system could be conveniently added in two possible ways. First, the ASHRAE toolkits for secondary and primary HVAC systems can be integrated into the simulation programs. The integration could be easily carried out because the ASHRAE toolkits for load calculation and HVAC systems are developed with the same methodology in terms of the programming language and the programming styles. Second, the state-of-the-art energy simulation program EnergyPlus (U.S. Department of Energy 2003) can replace the ASHRAE toolkit used in the current study. Because EnergyPlus and the ASHRAE toolkit have the same input and output mechanisms, the simulation program could be easily updated to EnergyPlus.

The ASHRAE toolkit for building load calculations is an updated library collection of load-related models and algorithms. All routines in the toolkit are written in Fortran 90, and all the algorithms in the toolkit are organized in a three-level structure comprising subroutines and functions, components, and sample zone models. The subroutines and functions are the fundamental building block of the toolkit; for example, the BLAST model used to calculate the convective heat transfer coefficient of exterior surfaces (Pedersen et al. 2000) is a subroutine of the toolkit. Multiple closely related subroutines and functions are assembled together to form various toolkit components; for example, exterior convection is a component including all implemented methods to calculate the convective heat transfer coefficient of exterior surfaces. Components are then assembled to form three zone models representing three different solution methods; for example, successive substitution is a solution method to solve the heat balance equations.

Although the original purpose of the zone models available in the Toolkit is intended to demonstrate how the components can be integrated, they can be directly used as simulation programs. In this study, the zone model with the successive substitution method is used to calculate the hourly loads of a building design.

The toolkit uses text-based data files as its input and output. For the input, two text files are required: the input data dictionary (IDD) named *toolkit.idd* and the input data file (IDF) named *toolkit.idf.* The IDD defines the fixed format to organize the objects and their attributes. The IDF provides the specific data values for objects. The data values in the IDF and the attributes in the IDD are matched through the predefined keywords of objects. A particular advantage of this data input mechanism is that it does not have a special sequence requirement for the objects in the IDF as long as the name of each object is unique and corresponds to the predefined keyword in the IDD. For the output, the toolkit presents hourly loads in the text file *toolkit.out*. A text file with the name of *toolkit.err* is also produced to indicate whether the toolkit has terminated successfully, and if not, the error information is provided.

The ASHRAE toolkit is used to calculate hourly heating or cooling loads for a given day that may correspond to (1) the average weather condition for the calculation of energy consumption; or (2) the extreme weather condition for peak load calculation. Since the life-cycle cost and the life-cycle environmental impact are the two major objective functions in the optimization model, the ASHRAE toolkit needs to be extended to derive the objective function values from the hourly loads.

4.2.2 Toolkit Extensions

The extension work intends to bridge the gap between the toolkit outputs and the objective function values. Therefore, the objective functions must be analyzed in order to trace their connections to hourly building loads. The previous chapter has shown that the life-cycle cost includes two items: the initial cost and the operating cost over the service life of a building. Additionally, the life-cycle environmental impact also includes two items: the embodied environmental impacts and the operating environmental impacts. Both the operating cost and the operating environmental impacts are related to the annual operating energy consumption due to heating, cooling, lighting, and equipment. As can be seen from Equations 3-9, 3-10, and 3-11, the energy consumption for heating and cooling can be calculated from the hourly loads provided by the ASHRAE toolkit; the energy consumption for lighting and equipment can be calculated based on their densities defined in the ASHRAE toolkit input file.

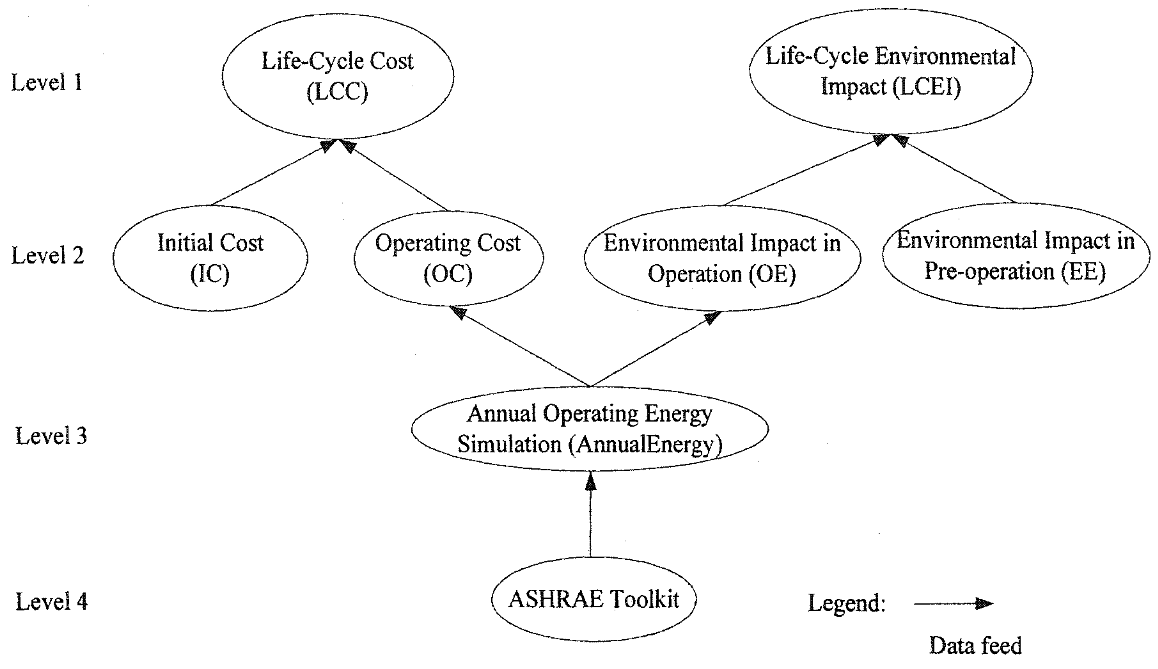

Level 1
Life-Cycle Cost (LCC)
Life-Cycle Environmental Impact (LCEI)
Level 2
Initial Cost (IC)
Operating Cost (OC)
Environmental Impact in Operation (OE)
Environmental Impact in Pre-operation (EE)
Level 3
Annual Operating Energy Simulation (AnnualEnergy)
Level 4
ASHRAE Toolkit
Legend:
Data feed

Figure 4.2 Module hierarchy of the simulation programs

The module hierarchy of the simulation programs is illustrated in Figure 4.2. The ASHRAE toolkit is the foundation, and the extension work can be summarized as follows.

- Prepare the input data file for each typical day in the heating and cooling seasons.
- Call the toolkit for each typical day in the heating and cooling seasons via the operating system.
- Check the error file to see whether the toolkit has finished successfully.
- Compute the operating energy consumption for the heating and cooling system by

considering the efficiency of the mechanical systems.

- Compute the operating energy consumption for lighting and equipment based on the floor area and their predefined densities and schedules.
- Sum up together the energy consumption used for heating, cooling, lighting, and equipment to obtain the annual operating energy consumption.
- Find the peak load for each month corresponding to the extreme weather conditions.
- Calculate the life-cycle operating cost based on the annual operating energy consumption, the monthly peak loads, the utility rates, the service life, and the effective interest rate.
- Calculate the initial construction cost based on the building geometry and the construction cost data per unit area for walls, windows, roofs, floors, and overhangs.
- Sum up together the life-cycle operating cost and the initial construction cost to obtain the life-cycle cost of a building design.
- Calculate the operating environmental impacts based on the annual operating energy consumption, the service life, and the emission factors of operating energy.
- Calculate the embodied environmental impacts based on the building geometry

and the embodied environmental impact data per unit area for walls, windows, roofs, floors, and overhangs.

- Sum up together the operating and embodied environmental impacts to obtain the life-cycle environmental impacts of a building design.

4.3 Data Files

The data files play the role of databases. They store the required data and are visited frequently by the simulation programs in the optimization process. This section presents two aspects of data files: data organization and data preparation.

4.3.1 Data Organization

Generally, a data file corresponds to a discrete variable representing an entity with many attributes. The correspondence relationship between the data file and the entity can be seen in two ways: (1) each record in the data file corresponds to a potential design alternative of the entity; (2) each data item in a record corresponds to an attribute of the entity. In this study, discrete variables that represent entities with many attributes include opaque envelope layers, window types, and overhang types. These three kinds of entities have different attributes; for example, windows have an attribute to measure the ability to transmit sunlight, whereas opaque envelope layers and overhangs do not have that attribute. Therefore, separate data files are needed for those three kinds of entities.

Data retrieval efficiency is a major consideration in organizing data files for opaque envelope layers. An opaque envelope usually consists of several layers with different functionalities. Many possible materials can be used to achieve the desired functionality for a given layer. Hence, there would be numerous data records if all layer alternatives were put in the same file. Because a large file with many data records slows down the process of locating data, it is better to divide the large file into several small ones according to the functionality of opaque envelope layers. Thus, in this research, a data file is used for each of the following eight layers: cladding, finish, insulation, membrane, sheathing, stud-insulation, structure, and other miscellaneous layers such as air space. The envelope layers are thus classified for the convenience of data organization although there is no strict clear-cut boundary between different functionalities for envelope layers. For example, stud-insulation, which means the steel or wood stud with insulation in cavities, plays the role of both structure and insulation.

Therefore, there are in total ten data files: eight for envelope layers, one for window types, and one for overhang types. These files have fixed formats. Figures 4.3, 4.4, and 4.5 show the file template, respectively, for envelope layers, window types, and overhang types. Each template is illustrated with an example in the lower part of the three figures. Explanations of the file templates are given below.

- The letters A and N in the templates are used to identify the nature (an alphabetic string or a numeric value) of the data item in a data file. The meaning of each data item is briefly described after a backward slash.

- Each data item is separated with a comma, and each record is ended with a semicolon. Comments preceded by an exclamation mark are permitted in the data files, but they must be placed between data records, not data items.

- A long name and a short name are required for each kind of material and each window type, and they are denoted respectively as A1 and A2 in Figures 4.3 and 4.4. The long name makes it easier for users to distinguish materials or window types in the same file, so it can be descriptive and has no length limit. The short name is used in the IDF (input data file) of the ASHRAE toolkit, so it has a maximum length of 40 characters as required for alphabetic strings in the IDF.

- All numeric data items in the file templates can be grouped into three subsets: physical and thermal property data, construction cost data, and environmental impact data. These data items are thus defined in the data files in order to satisfy the simulation requirement. The first subset of data is mainly used by the ASHRAE toolkit, and the other two subsets of data are used by the developed toolkit extensions.

- The cost and environmental impact data are based on a 100 m^2 area, which aims at scaling up data to improve accuracy.

- The embodied energy in terms of fuel types, the considered non-fuel resources, and the considered emissions are completely listed in the figure for the file template of envelope layers. The same three groups of data are abbreviated in the other two templates to save space.

- Embodied energy needs to be differentiated in terms of fuel types because the ratio between exergy and energy (α in Equation 3-5) may vary with fuel types.

A1,	\long name of material layer
A2,	\short name of material layer
N1,	\absorptivity
N2,	\emissivity
N3,	\roughness value
N4,	\cost in \$ per 100 m^2
N5,	\embodied energy from hydro electricity (MJ/100 m^2)
N6,	\embodied energy from liquefied propane gas (LPG) (MJ/100 m^2)
N7,	\embodied energy from diesel (MJ/100 m^2)
N8,	\embodied energy from gasoline (MJ/100 m^2)
N9,	\embodied energy from natural gas (MJ/100 m^2)
N10,	\embodied energy from wood (MJ/100 m^2)
N11,	\embodied energy from coal (MJ/100 m^2)
N12,	\embodied energy from heavy oil (MJ/100 m^2)
N13,	\embodied energy from nuclear (MJ/100 m^2)
N14,	\embodied energy from feedstock fuels (MJ/100 m^2)
N15,	\limestone consumption (kg/100 m^2)
N16,	\clay and shale consumption (kg/100 m^2)
N17,	\iron ore consumption (kg/100 m^2)
N18,	\sand consumption (kg/100 m^2)
N19,	\gypsum consumption (kg/100 m^2)
N20,	\coarse aggregate consumption (kg/100 m^2)
N21,	\fine aggregate consumption (kg/100 m^2)
N22,	\scrape steel consumption (kg/100 m^2)
N23,	\wood fiber consumption (kg/100 m^2)
N24,	\phenol formaldehyde resin consumption (kg/100 m^2)
N25,	\CO_2 emissions (kg/100 m^2)
N26,	\CO_2 emissions due to biomass (kg/100 m^2)
N27,	\CH_4 emissions (kg/100 m^2)
N28,	\N_2O emissions (kg/100 m^2)
N29,	\SO_x emissions (kg/100 m^2)
N30, ;	\NO_x emissions (kg/100 m^2)

```
1/2" Firerated Gypsum Board on Wall, F1, 0.65, 0.9, 5, 678, 277, 0, 389, 0, 4099, 0, 1130,
857, 460, 13, 0, 48, 1, 0, 840, 0, 0, 2, 0, 0, 256, 0, 0.5, 0, 3.4, 0.8, ;
```

Figure 4.3 File template and an example for opaque envelope layers

```
A1,     \long name of window type
A2,     \short name of window type
N1,     \surface height from ground (m)
N2,     \outside surface solar absorptivity
N3,     \inside surface shortwave absorptivity
N4,     \outside surface longwave emissivity
N5,     \inside surface longwave emissivity
N6,     \window reveal (m)
N7,     \transmissivity of interior shading device
N8,     \diffuse solar heat gain coefficient (SHGC)
N9,     \diffuse transmittance
N10,    \diffuse absorptance of glazing layer 1
N11,    \diffuse absorptance of glazing layer 2
N12,    \diffuse absorptance of glazing layer 3
N13,    \number of pairs of SHGC and corresponding incident angles
N14~N23,        \incident angle #1 to #10
N24~N33,        \SHGC corresponding to incident angle #1~#10
N34~N43,        \absorptance for glazing layer 1 corresponding to incident angle #1~#10
N44~N53,        \absorptance for glazing layer 2 corresponding to incident angle #1~#10
N54~N63,        \absorptance for glazing layer 3 corresponding to incident angle #1~#10
N64~N73,        \transmittance corresponding to incident angle #1~#10
N74,            \cost in $ per 100 m2
N75~N84,        \embodied energy for fuel types as defined in the previous template
N85~N94,        \resource consumption as defined in the previous template
N95~N100, ;     \emissions as defined in the previous template
```

```
Double Pane Standard Glazing, DoublePaneWindow, 1.2, 0, 0, 0.9, 0.9, 6,
0.1, 1, 0.60, 0.51, 0.19, 0.11, 0, 10,
0,10,20,30,40,50,60,70,80,90,
0.70, 0.70, 0.69, 0.68, 0.67, 0.64, 0.58, 0.45, 0.23,0,
0.17, 0.17, 0.17, 0.18, 0.18, 0.19, 0.20, 0.21, 0.20,0,
0.11, 0.11, 0.11, 0.12, 0.12, 0.12, 0.12, 0.10, 0.07,0,
0,0,0,0,0,0,0,0,0,0,
0.61, 0.61, 0.60, 0.59, 0.58, 0.55, 0.48, 0.36, 0.17,0,
24758, 8569, 0, 2311, 0, 6903, 0, 2869, 556, 1180, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3299,
0, 4.4, 0, 22.2, 22.3, ;
```

Figure 4.4 File template and an example for window types

```
A1,       \long name of overhang type
N1,       \extension beyond left edge of window (m)
N2,       \extension beyond right edge of window (m)
N3,       \thickness (m)
N4,       \cost in $ per 100 m^2
N5~N14,        \embodied energy for fuel types as defined in the first template
N15~N24,       \resource consumption as defined in the first template
N25~N30, ;     \emissions as defined in the first template
```

```
Aluminum Overhang, 0, 0, 0.08, 21528, 31460, 9, 1748, 0, 23021, 0, 9603, 1745, 1692,
13050, 152, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1560, 0, 3.7, 0, 19.5, 18.0,;
```

Figure 4.5 File template and an example for overhang types

4.3.2 Data Preparation

The data has been obtained from various sources. Construction cost data are mostly obtained from the cost data books of R.S. Means (2004a, 2004b). The cost data for the few construction items unavailable in the R.S. Means data books are obtained from manufacturers or the Internet. For example, the construction cost of an aluminum overhang was obtained from the Ametco Manufacturing Corporation (Mitrovich 2004). Physical and thermal property data are obtained from two main sources: the ASHRAE handbook of fundamentals (2001) and the material database of the EE4 software (2000). Environmental impact data are extracted from the life-cycle assessment software Athena EIE (2003), which is the most time-consuming process in the data preparation. The software Athena EIE and the process of using it to extract environmental impact data are briefly presented below.

There are many life-cycle assessment programs as presented in Chapter 2. Athena EIE (2003) is selected in this study because it has the following advantages:

- The environmental impact data extracted from Athena EIE are applicable in this study because its life-cycle inventory databases are specifically developed for many locations in North America including Montreal.

- This program can model the commonly used assemblies of building structures and envelopes because its life-cycle inventory databases cover typical materials for the construction of structures and envelopes.

- Overlap, waste, and other miscellaneous ancillary materials are considered in estimating the life-cycle environmental impacts for an assembly construction, thus improving the accuracy.

- The life-cycle environmental impact data extracted from Athena EIE can be directly used in computing the objective function values because the program presents its results based on the construction surface area, not on mass.

- Athena EIE presents an exhaustive list of values for the natural resource consumption, the embodied energy consumption by fuel types, and the waste emission of a given assembly. These values are essential to calculate the objective function values.

The Athena EIE software can estimate the life-cycle environmental impacts of a given construction with two methods: using a predefined assembly (e.g., steel-stud wall and concrete flat plate floor) in the program or using basic materials (e.g., 20 MPa concrete in m^3 and nails in kg). The two methods differ in that on-site construction environmental impacts are considered in the first method but ignored in the second one.

The first method is employed to extract environmental impacts for opaque envelope layers. This method requires an envelope layer be dealt as a component of an assembly. In this study, the following steps are used to estimate the environmental impact data of an envelop layer:

(1) A basic wall assembly (e.g., steel-stud wall 39*92@400) is defined, and the Athena EIE software is run to obtain the environmental impact data including resource consumption, embodied energy, and waste emissions.

(2) An envelope layer (e.g., 12.7 mm regular gypsum wallboard) is added to the basic wall assembly defined in the first step, and the program is run again to obtain the environmental impact data for the new assembly.

(3) The difference between the results from the above two steps are the environmental impacts for the envelope layer (i.e., 12.7 mm regular gypsum wallboard).

The second method is employed to extract the environmental impact data for windows and overhangs because of the following two reasons. First, Athena EIE requires detailed information about window frames and mullions if windows are dealt with as assemblies. Second, Athena EIE does not define overhangs as assemblies.

After the values for construction cost data, physical and thermal property parameters, and embodied environmental impacts are obtained, the data files for envelope layers, windows and overhangs are prepared according to the formats presented in the previous subsection. Each file contains a number of design alternatives commonly used in practice. Table 4.1 shows some statistics about the number of alternatives available in each file. All files are presented in Appendix A of this thesis.

Table 4.1 Overview of data files

File name	**Number of total alternatives**	**Main alternatives**
cladding.txt	7	brick, concrete block, wood siding, vinyl siding, steel, siding, and stucco
finish.txt	14	gypsum, mineral fiber acoustical tile
insulation.txt	38	expanded polystyrene, extruded polystyrene, ployisocyanurate, and mineral fiberboard
membrane.txt	7	built-up roof, modified bitumen membrane, EPDM, polyethylene, sheathing paper
sheathing.txt	17	oriented strand board (OSB), plywood, gypsum
structure.txt	11	concrete block, cast-in-place concrete, open-web-steel-joist with steel deck
studInsulation.txt	8	steel-stud with fiberglass, steel-stud with rockwool
other.txt	7	air space, green roof
windowType.txt	9	single, double clear, double low-e, triple clear, triple low-e
overhangType.txt	1	aluminum overhang

4.4 Optimizer

The optimizer refers to the optimization program implemented in the system. This section first analyzes the particularities of the optimization problem formulated in this study. Then, the selected optimization technique – genetic algorithms – is presented.

4.4.1 Analysis of the Optimization Problem

The selection of the optimization algorithm depends on the particularities of the optimization problem. The optimization problem presented in the previous chapter has the following characteristics:

- It might be a single- or multi- objective optimization problem. Single-objective optimization aims at finding one single solution while multi-objective optimization aims at finding a set of Pareto solutions. A solution is said to be Pareto optimal if and only if it is not dominated by any other solution in the whole decision space. If solution X_1 dominates another solution X_2, it implies that X_1 is non-inferior to X_2 for all the considered performance criteria, but it is better than X_2 for at least one criterion. Pareto solutions are non-dominated with each other because neither of them is better than the other for all the considered performance criteria. All the points in the objective function space corresponding to Pareto solutions form a Pareto front, which is useful to understand the trade-off relationship between different performance criteria.

- It might be an optimization problem with or without functional constraints.

- It is a hard combinatorial problem. To illustrate this point, let us consider a simple design problem with the following variables and corresponding number of alternatives (the number in parenthesis): orientation (10), aspect ratio r_0 (10), *winType* (3), *wallType* (3), *roofType* (3), *winRatio*$_i$ (10), each *wallLayer*$_j$ (5), each *roofLayer*$_j$ (5), with i=1 to 4 standing for façades and j=1 to 5 standing for layers; there are about $2.6*10^{10}$ possible solutions to explore.

- The objective functions may be discontinuous with respect to certain variables. The ASHRAE toolkit used in this study is a simulation program that contains code features such as an iterative solver and if-else-then logic. These code features could cause the objective functions to be discontinuous (Wetter and Wright 2004).

- The optimization problem has structured variables as presented in the previous chapter. The structured variables make some sub-level variables active and some not.

- Both continuous and discrete variables coexist in the same optimization problem.

There are many optimization techniques such as direct search methods, gradient-based methods, and stochastic optimization methods (Rao 1996). Direct search methods evaluate and compare a series of variable vectors to find the optimal solution. Because direct search methods do not explicitly use derivatives, they are usually inefficient to solve an optimization problem with a large search space. Despite some advancements (Kolda et al. 2003), direct search methods cannot perform well for optimization problems with functional constraints and discrete variables. Gradient-based optimization methods use the first or second derivatives of objective functions to determine the search direction. Although gradient-based optimization methods can quickly converge to an optimal solution, they usually have strict requirements on the optimization problem such as second-order differentiable objective functions and continuous variables. Stochastic optimization methods search for an optimal solution involving randomness in some constructive ways such as the Metropolis acceptance probability in simulated annealing and the selection operation in genetic algorithms. Because stochastic optimization methods accept moves with worse objective function values in a limited way, they are not easily trapped in a local optimum. In this respect, Wetter and Wright (2004) compared the deterministic and stochastic algorithms for simulation-based, single-objective optimization problems. They found that compared with the direct search methods and the gradient-based methods, the genetic algorithm can get much closer to the optimal solution with a comparable number of simulation calls.

In addition, the ability to deal with multi-objective optimization is another important consideration in selecting optimization algorithms for this study. Traditional numerical optimization methods usually solve a multi-objective optimization problem by converting it to a single-objective optimization problem (Hwang and Masud 1979; Miettinen 1999). Thus, only one Pareto optimal solution can be obtained in each run, and many runs are required to obtain a set of Pareto solutions. Even so, the diversity of Pareto solutions cannot be easily controlled by the numerical optimization methods, which means that the obtained Pareto optimal solutions may be the same for different runs. Compared with the traditional numerical methods, genetic algorithms are able to locate multiple Pareto optimal solutions in a single run.

Besides the global search and the ability to find multiple Pareto optimal solutions, genetic algorithms have several other advantages: (1) they can deal with discrete variables easily; (2) they do not require the objective functions to be continuous; (3) they are efficient to search a large design space; (4) they are applicable to a wide variety of problems. Because of these advantages, genetic algorithms are used in this research.

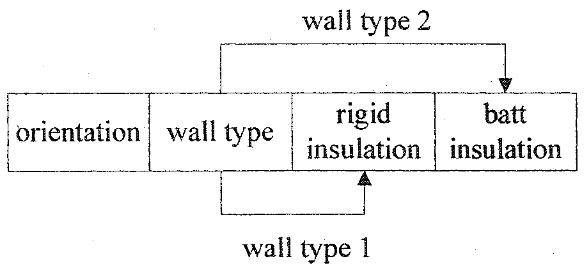
4.4.2 Genetic Algorithms

Genetic algorithms (GAs) are a stochastic global search technique inspired from the principles of biological evolution (Goldberg 1989). The basic aspects of GAs such as representation and operators are introduced first. Then, specific issues regarding multi-objective genetic algorithms are presented.

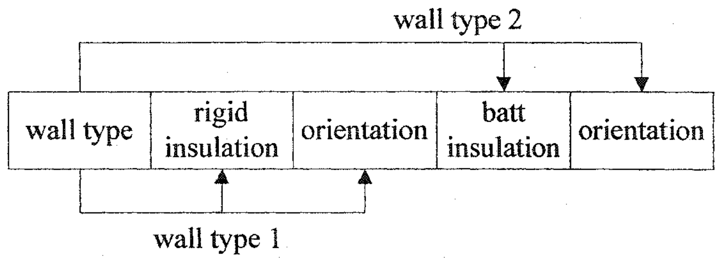
In genetic algorithms, a variable is usually coded into a fixed-length string of bits consisting of "1"s and "0"s. The number of bits required depends on the precision and the interval for a continuous variable, and on the number of alternative values for a discrete variable. The binary codes of all variables are concatenated to form a binary string (i.e., a chromosome), representing a potential solution to the optimization problem. Decoding is required to map the binary string back to real values for all variables.

A special mechanism is needed to handle structured variables in the GA representation to identify those active variables at different levels. This issue is addressed by the "structured GA" proposed by Dasgupta and McGregor (1993). In structured GAs, the chromosome is represented as hierarchical genomic structures, which means that dominant and recessive genes for low-level variables may coexist in a chromosome. High-level genes determine which low-level genes are active. The following example illustrates the workings of a structured GA. Suppose that orientation, wall type, and wall insulation are the three variables considered in an optimization problem. The wall type is at a different level from the wall insulation because it has an impact on the selection of the insulation materials. In this example, the two considered wall types are the concrete block wall and the steel-stud wall. The wall insulations consist of several rigid insulation alternatives for the concrete block wall and several batt insulation alternatives for the steel-stud wall. The coding representation using the structured GA for this case is illustrated in Figure 4.6 (a). Acting as a switch, the variable at the higher level (i.e., the wall type) determines which sub-level variable (i.e., rigid insulation or batt insulation) is active. If the gene of the variable wall type corresponds to the concrete block wall, the genes of the rigid insulation are dominant while those genes for the batt insulation are recessive. In other words, the batt insulation is ignored in the definition of a building design.

It needs to be noted that in this study, the way to represent variables not located in the hierarchical levels (e.g., orientation in the above example) is different from the former study (Rafiq et al. 2003). They are represented in parallel with the top-level structured variables (e.g., wall type in the example) in this research. However, they are represented in parallel with the lowest-level variables of the hierarchy (e.g., rigid insulation and batt insulation in the example) by Rafiq et al., as shown in Figure 4.6 (b). Since non-hierarchical variables such as orientation are represented repetitively, a major disadvantage of the representation method by Rafiq et al. lies in the much longer chromosome, which requires more time for the GA to converge.



(a) representation in this study



(b) representation in the previous study (Rafiq et al. 2003)

Figure 4.6 Comparison of structured GA representation

Genetic algorithms maintain and operate on a set of potential solutions (chromosomes), called a population of individuals. The robustness or fitness of each individual is related to its objective function values. The evolution process starts from an initial population generated randomly. This population is subject to genetic operators such as selection, crossover, and mutation to create a new population. These operators are then applied on the new population, and this process is repeated until some predefined stopping criterion (e.g., the maximum number of generations) is satisfied. The three basic genetic operators are presented below.

The selection operator is used to select individuals for reproduction. The selection procedure follows the survival-of-the-fittest principle and thus provides more opportunity to reproduce the individuals with higher fitness than those with lower fitness. This procedure can be realized with various techniques, which are broadly classified into two groups (Cantu-Paz 2001): fitness-proportionate selection and rank-based selection.

- Fitness-proportionate selection implies that the number of solutions copied to the mating pool is proportional to their fitness values. Some examples are roulette wheel selection (RWS), stochastic universal selection (SUS), and stochastic remainder selection (SRS). RWS uses a simulated roulette wheel divided into N (population size) divisions with sizes in proportion to the fitness of each individual. Thereafter, the wheel is spun once for each individual to be selected. SUS and SRS are two less noisy versions of RWS because both of them have reduced the variance of results from the production of random numbers. SUS requires N (population size) equidistant markers placed outside the simulated wheel. The wheel is spun only once and the number of copies of each individual is equal to the number of markers landing in the corresponding division. SRS is carried out with two steps. Each individual is first assigned deterministically a number of copies equal to the integral part of its expected number, calculated by dividing its fitness by the average fitness over the whole population. Then, the fractional remainders are used in the same way as RWS to fill up the mating pool.

- Rank-based selection uses an individual's rank to determine its opportunity for reproduction. The rank refers to the ordinal position of an individual in a sorted population according to the objective function values. Two commonly used rank-based selection methods are linear ranking and binary tournament selection. In linear ranking selection, the selection probability of an individual is linearly proportional to its rank in the population. In binary tournament selection, two individuals are taken at random from the population, and the better one is selected. An advantage of rank-based selection is to avoid premature convergence caused by a few extraordinary robust individuals.

After the mating pool is formed with the selection operation, the crossover operator is applied. Usually, two individuals are taken from the mating pool at random. A random number is generated and compared with a predefined crossover probability. If the random number is less than the crossover probability, the chosen two individuals mate; otherwise, they are copied to the new population directly. Various crossover procedures exist in the GA literature (Spears 1997). A brief description of the single-point crossover is presented here because it is the most frequently used in GA applications. The single-point crossover operator recombines two individuals by exchanging part of their binary strings, starting from a randomly chosen crossover point along the string length. This leads to two new solutions that most probably inherit desirable qualities of their parents.

The usual single-point crossover operation, however, has a drawback for structured GAs. It is better to illustrate this drawback with an example. For the purpose of this example, the design problem has one structured variable with two alternatives. For this problem, two chromosomes representing two potential solutions are shown at the top of Figure 4.7. These two chromosomes have the following two characteristics: (1) they have the same high-level gene for the structured variable; and (2) they have the same active genes on one side (i.e., the right side) of the structured genes. In this case, if the crossover point is located in between the genes of the inactive alternative (i.e., the first alternative), the two children generated after the crossover operation will have the same active genes as their parents, but the simulation program will still be executed. This is not a rare phenomenon in the evolution process, especially at the later stage when the GA has almost converged. The frequent occurrences of such situations have a couple of unfavorable effects. First, it wastes computation time on unnecessary simulation calls because two individuals with the same active genes have the same function values. Second, it slows down the GA convergence because the crossover operation does not produce more robust individuals. In this study, the problems resulting from the usual single-point crossover for structured GAs are addressed by modifying the crossover operation. Figure 4.7 shows the modified version of single-point crossover, which involves three steps as follows.

- The first step concatenates the active genes together while ignoring the inactive genes. This step usually results in a shortened chromosome, based on which a proxy crossover point is chosen at random.
- The second step inserts the inactive genes back to the chromosome and determines the actual crossover point in the chromosome.
- The third step exchanges the genes on one side of the actual crossover point.

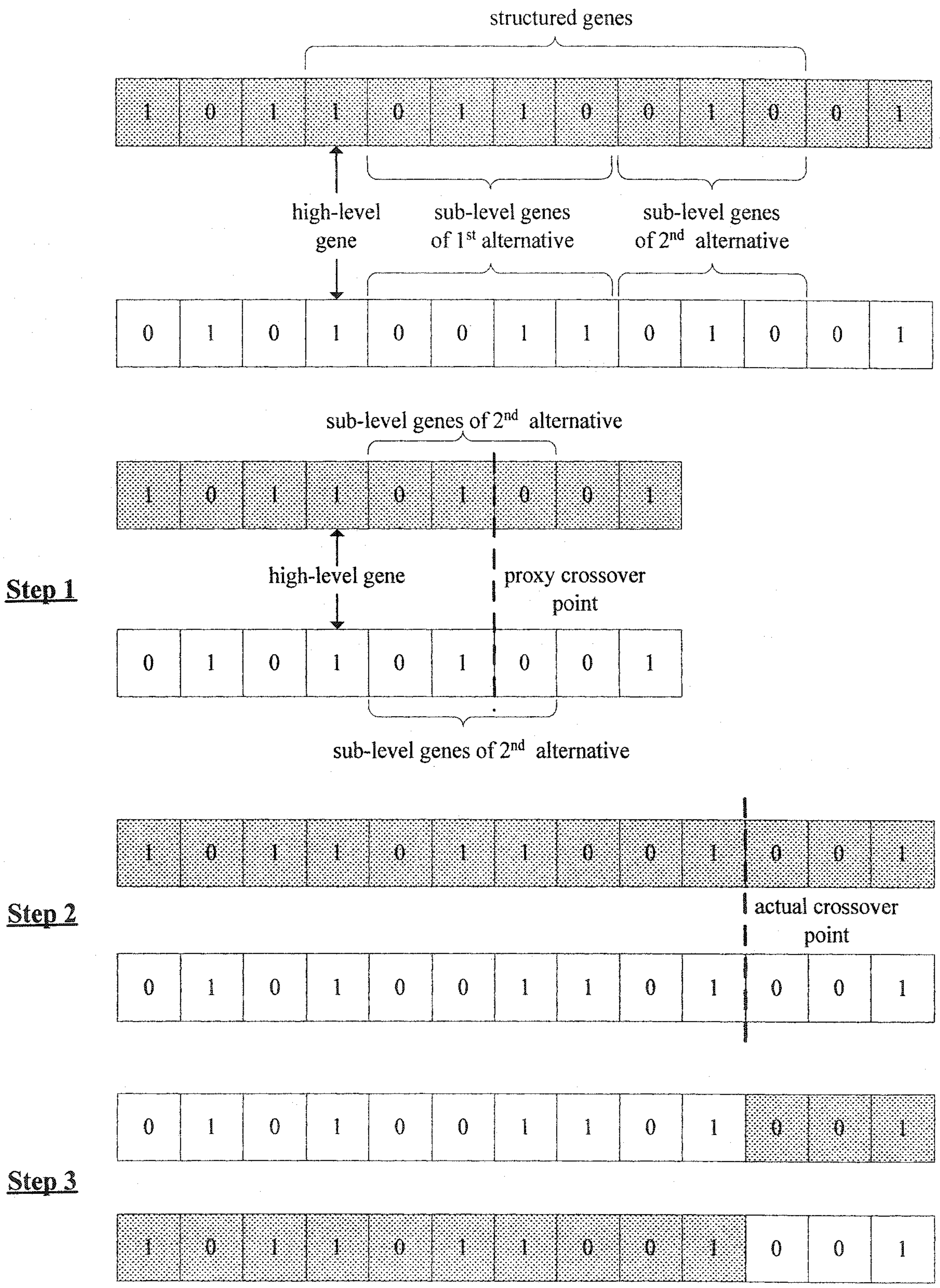


Figure 4.7 Modified single-point crossover for the structured genetic algorithm

The mutation operator is applied on the new offsprings created by the crossover. A bit-wise procedure is usually required to implement the mutation. A random number is generated for each bit in the gene pool to determine whether the bit value should be flipped according to a predefined mutation probability. The purpose of the mutation operator is to keep diversity in the population by exploring new or retrieving lost genetic information.

Of the above three basic GA operators, the selection operator plays an exploitation role by utilizing robust individuals that have been found in the evolution process. The crossover and mutation operators play an exploration role by creating new individuals. An appropriate balance between exploration and exploitation is important for the good performance of GAs. This can be achieved by tuning the GA parameters such as the selection method, the crossover probability, and the mutation probability. Besides the basic genetic operators, additional handling techniques can be employed to improve the performance of GAs, particularly for multi-objective genetic algorithms as discussed next.

In contrast to the single-objective GAs aiming at finding one global optimal solution, the multi-objective genetic algorithms (MOGAs) aim at two distinct goals (Deb 2001):

- to discover solutions as close to the true Pareto optimal solutions as possible; and
- to find solutions as diverse as possible along the obtained Pareto front.

In recent years, there has been an active and growing interest in implementing and applying GAs for multi-objective optimization. Hence, many MOGAs have been proposed in the GA literature. Several reviews regarding MOGA developments have been made by Coello (1999), Veldhuizen and Lamont (2000), and Tan et al. (2002). These reviews cover various techniques necessary to achieve the convergence and the diversity of Pareto optimal solutions. For example, Tan et al. grouped the handling techniques for MOGAs into eleven basic elements and five supporting elements according to their roles in the optimization. Basic elements (e.g., weights and the Pareto dominance scheme) have a direct relationship with the methodology used for finding the optimal solutions. Supporting elements (e.g., elitism and distribution of individuals) play an indirect role with the aim of achieving better results. Tan et al. (2002) also found that the Pareto dominance scheme is the most popular basic element while elitism and mating restriction are the two most popular supporting elements in MOGA studies. These popular elements are adopted in this research and presented below.

The Pareto dominance scheme explicitly uses the non-domination concept in the selection operation, which means that non-dominated individuals are normally assigned higher fitness values than those dominated ones. The Pareto dominance scheme can be implemented in different ways. In this study, the optimizer provides two ways: the rank-based fitness assignment method (Fonseca and Flemming 1998) and the front-based fitness assignment method (Srinivas and Deb 1994). For the rank-based method, the individuals are ranked, and the rank of an individual is equal to one plus the number of individuals in the current population that dominate it. Afterward, the linear function suggested by Deb (2001) is used to map ranks to initial fitness values so that the individual with the lowest rank has the maximum fitness value and vice versa. Then, the initial fitness values of the same-rank individuals are processed further with a niche sharing technique to be discussed later. For the front-based method, the fronts (i.e., the non-dominated individual sets) are sorted. The first front consists of all non-dominated individuals in the current population, and the subsequent front consists of all non-dominated individuals after those individuals belonging to the previous fronts are removed from the population. The front-based method starts from the first front and successively proceeds to other fronts. All individuals in the first front take an arbitrary large number, which could be equal to the population size (Deb 2001), as their initial fitness values. The individuals in the subsequent front are given the same initial fitness value that is a little smaller than the minimum shared fitness value of the individuals in the preceding front, where the shared fitness value of an individual is obtained through the niche sharing with other individuals in the same front.

The niche sharing is an important technique for MOGAs to maintain diversity in the obtained non-dominated front. Motivated by the natural phenomenon that species living in a space have to share the available resources around them, the niche sharing in GAs requires that close individuals in the population be penalized by a reduction in fitness. The following sharing function is used to estimate the extent of sharing with others around a solution (Goldberg 1989):

$$\mathrm{Sh}(d) = \begin{cases} 1-\left(d/\sigma_{\mathrm{share}}\right)^{\alpha} & (d \le \sigma_{\mathrm{share}}) \\ 0 & (d > \sigma_{\mathrm{share}}) \end{cases} \qquad (4\text{-}1)$$

where,

d : the distance between two solutions (e.g., the normalized Euclidean distance)

σ_{share}: the niche radius

α: the exponent of the sharing function, usually taking the value of 1

The niche radius σ_{share} is a critical parameter in the sharing function; however, its appropriate value cannot be easily acquired in the absence of any knowledge about the Pareto optimal front. The dynamic update approach suggested by Fonseca and Flemming (1998) is employed here to determine the value of σ_{share}. If the normalized Euclidean distance between individuals in the performance space is used, the niche radius can be calculated as (Deb 2001):

$$(1+\sigma_{share})^M - 1 = N \cdot (\sigma_{share})^M \qquad (4\text{-}2)$$

where,

M: the number of objective functions

N: the population size

For the case of two objective functions, that is, M=2, the niche radius derived from Equation 4-2 is equal to $2/(N-1)$.

After all the sharing function values are known for an individual, the niche count of this individual can be obtained by adding them together. The shared fitness value of this individual, which is used in the selection process, is calculated by dividing its initial fitness value by its niche count.

In addition to the Pareto dominance scheme, elitism and mating restriction are two other popular techniques in MOGA studies (Tan et al. 2002). Elitism preserves elites, i.e., robust individuals, in the evolution process. Appropriate elitism can effectively improve the performance of MOGAs because: (1) it makes sure the non-dominated solutions do not deteriorate with generations; (2) it can enhance the probability of creating better offspring; and (3) it can speed up the convergence. There are many ways to introduce elitism into MOGAs (Deb 2001). The external population method is employed in this research. It is an elitist strategy that has been used in several studies such as Osyczka and Kundu (1995), Zitzler and Thiele (1999), and Knowles and Corne (2000). Here, the external population has a predefined capacity. It works as follows. After each generation is produced, the non-dominated individuals are copied to the external population, and the individuals that become dominated in the external population are removed. If the external population cannot accommodate all the elites, some individuals located in crowded regions are removed.

Mating restriction is a non-random mating mechanism to form and maintain subpopulations with different characteristics (Huang 2002). With the usual random mating mechanism, two good but dissimilar individuals may become partners. Their mate, however, sometimes produces low-performance offspring in MOGAs, as noticed in a few previous studies (e.g., Fonseca and Flemming 1998). To reduce the number of degraded offspring caused by random mating, mating restriction requires that an individual actively search for its partner so that the two individuals are similar but not identical according to some metrics (e.g., the normalized Euclidean distance used in this research). Thus, the mating restriction allows the crossover operation to occur between two different individuals only if their normalized Euclidean distance is less than the mating radius, a threshold parameter taking the same value as the niche radius presented earlier. If an individual cannot find its partner, this means that the individual is in a sparse region where the GA should explore more. Therefore, that individual is copied twice to the new population.

4.5 Input and Output

The input and output component works as the interface between the user and the system. Through this component, the user can customize the simulation programs and the optimizer. Both inputs and outputs are text files. The text files for the simulation programs and for the optimizer are described in this section.

4.5.1 Input Text Files for the Simulation Programs

The input data file for the ASHRAE toolkit, *toolkit.idf*, varies with design alternatives. For a given design alternative, the file *toolkit.idf* also varies with the weather data for each typical day. This toolkit input file, however, has some data objects such as lighting and occupancy schedules that can remain unchanged during optimization. To facilitate generating the file *toolkit.idf*, the user needs to prepare the following text files for the ASHRAE toolkit: weather data files and the file *fixedPart.idf*. Each typical day has a weather file, which contains the following data objects: *Date, Environment, TempInside, TempOutside, TempWetOutside, TempSpecial, and TempDeck*. The file *fixedPart.idf* contains all data objects that remain constant in the toolkit input file for all design alternatives. Both weather files and *fixedPart.idf* have the same formats as the toolkit input file.

Besides the text files related to the ASHRAE toolkit, the user needs to prepare another text file called *simulationData.txt*, which is required by the extended simulation program as presented in section 4.2. The file template of *simulationData.txt* is shown in Figure 4.8, which uses the same formats as the data files presented in section 4.3. Several explanations regarding the file template are given below.

- The data in the file *simulationData.txt* are divided into three sections: data for the annual operating energy consumption, data for the LCC (life-cycle cost), and data for the LCEI (life-cycle environmental impact). Each section follows an identifier; for example, the data section for the annual operating energy consumption follows the identifier of "#Energy Data Begin#", as shown in Figure 4.8. This division organizes the data in a logical manner because it allows the user to ignore one or two data sections in some situations. For example, if an unconstrained optimization problem is defined with the LCC as the single objective function, the data section for the LCEI can be ignored.

- Ellipses (i.e., "...") and two-level index numbers (e.g., A3-1 and N8-1) are used for those fields with indeterminate data items such as energy demand rates and the weather files for heating energy consumption. The ellipses indicate that there are omitted data items, and the two levels of index numbers represent the ordinal positions of values respectively in the data file and in the data field.

- For both energy demand rates and energy consumption rates, the number of threshold values is always one less than the number of rates. If there is only one single rate, the data items indicating threshold values should be left out.

#Energy Data Begin#

N1,	\floor area (m^2)
N2,	\floor height (m)
N3;	\building service life (year)
N4,	\performance index of heating system
N5;	\performance index of cooling system
A1;	\file path for *fixedPart.idf*
A2;	\file path for the toolkit execution program *toolkit.exe*
A3-1, …, A3-h,;	\file path for weather files used for heating energy consumption \h is the total number of weather files for heating energy consumption
A4-1, …, A4-c,;	\file path for weather files used for cooling energy consumption \c is the total number of weather files for cooling energy consumption

#LCC Data Begin#

N6;	\local cost factor
N7,	\total number of energy demand rates, assumed equal to m
N8-1, …, N8-m-1,	\upper threshold values of energy demand (kW)
N9-1, …, N9-m,;	\demand rates ($/kW)
N10,	\total number of energy consumption rates, assumed equal to n
N10-1, …, N10-n-1,	\upper threshold values of energy consumption (kWh)
N11-1, …, N11-n,;	\energy consumption rates ($/kWh)
N12,	\discount rate
N13,;	\fuel price escalation rate
A5-1, …, A5-h,;	\file path for weather files used for heating peak load
A6-1, …, A6-c,;	\file path for weather files used for cooling peak load

#LCEI Data Begin#

N14~N23,;	\ratios between exergy and energy for hydro electricity, LPG, diesel, gasoline, natural gas, wood, coal, heavy oil, nuclear energy, and feedstock fuels
N24~33,;	\chemical exergy for limestone, clay, iron ore, sand, gypsum, coarse aggregate, fine aggregate, scrape steel, wood fiber, and phenol formaldehyde resin (MJ/kg)
N34,;	\ratio between exergy and energy for the fuel type used in building operation
N35,;	\overall efficiency value for the fuel type used in building operation
N36~N42,;	\waste emission factors for the fuel type used in building operation (kg/MJ), in sequence of CO_2, CO_2 biomass, CH_4, N_2O, SO_x, and NO_x
N43~N47,;	\unit abatement exergy for waste emissions (MJ/kg), in sequence of CO_2, CO_2 biomass, CH_4, N_2O, SO_x, and NO_x

Figure 4.8 Template for the file *simulationData.txt*

4.5.2 Text Files for the Optimizer

The input text file *environment.txt* is used to customize the optimizer in terms of the variables, the objective functions, the functional constraints, and the optimization parameters. Since this file is closely related to the system design and implementation, the discussion of its details is deferred to the next chapter.

Compared with the input, the output is not the emphasis of this study. Hence, output text files are designed for the convenience of results analysis with little consideration of the flexibility. There are two output files: *result.txt* and *elite.txt*. The file *result.txt* shows the evolution process. Every generation of individuals are printed in *result.txt* with different details. For the first, final, and intermediate populations at every twenty generations, detailed information is printed including the chromosome and the values of variables, objective functions, and functional constraints. For other generations, only brief information is printed including the objective function values and the functional constraint values. The file *elite.txt* shows the dynamic changes of the external population for multi-objective GAs. This file contains the objective function values and the functional constraint values for each individual in the external populations every twenty generations.

4.6 Summary

GBOptimizer is a simulation-based optimization system that consists of the simulation programs, the data files, the optimizer, and the input and output. The simulation programs are developed to calculate the life-cycle cost and the life-cycle environmental impact, which are the two major objective functions in the optimization model. The ASHRAE toolkit for building load calculations is employed to develop the simulation programs for this study. The data files act as the databases of construction costs, thermal properties, and embodied environmental impacts. The environmental impact data for building materials and assemblies are extracted from the life-cycle assessment program Athena EIE. Based on the problem characteristics, the genetic algorithm is selected as the optimization method because of its many advantages. The input allows users to customize the optimization problem in terms of the variables, the simulation programs, and the optimization parameters. The output provides the intermediate and the final optimization results. The underlying computer model of the simulation-based optimization system GBOptimizer is presented in the next chapter.

Chapter 5
Object-Oriented System Framework

This chapter presents the computer model of the system GBOptimizer. An object-oriented (OO) framework is developed to facilitate the reuse and the extension of the software design. This chapter begins with some fundamental issues of OO software design such as design patterns and the Unified Modeling Language (UML). Then, the aspects that are most likely to change for simulation-based optimization problems are identified to form the basis of the OO framework design. The framework design is presented with its major modules in the third section, following which the framework customization and implementation are discussed sequentially.

5.1 OO Framework Basis

As mentioned in Chapter 2, one of the limitations of former studies lies in their rigid system design. The rigid system design leads to a stand-alone application that is developed for a fixed optimization problem in terms of variables, optimization models, and simulation programs. Such an application is difficult to adapt, so it may require significant modifications to handle a new problem that differs from the original one only in the number of variables. The modifications require considerable time and resources in the analysis, redesign, implementation, and validation of the new application. Therefore, one objective of this research is to address the inflexibility of previous optimization systems by making the software design and code reusable and extendable to many applications in the same problem domain. This can be achieved by developing an OO framework. As defined by Mattsson (1996), an OO framework is a reusable software architecture represented by a set of abstract and concrete classes and their interactions, which are designed with the aim of reuse, extension, and customization for specific applications.

As an object-oriented design technique, an OO framework models a software system as a collection of cooperating objects. Each object is an instance of a class within a hierarchy of classes. Each class defines a group of objects with common properties and behaviors by its attributes and operations, respectively. In the modeling of a system, classes can be grouped into three types: entity, boundary, and control (Quatrani 2003). An entity class represents a real-word entity, which usually stores long-lived information independent of the surroundings. A boundary class represents a user interface component of the system to handle the communication between the system and its surroundings. A control class represents the procedure to accomplish a task of the system, and it involves sequencing and coordination of messages to other objects.

The UML by Booch et al. (1998) is employed in this study to visualize the classes and their interrelationships. Some notations and concepts relevant to this research are introduced below. A class is rendered in the UML as a rectangle with three compartments, which are used for the class name, the attributes, and the operations, respectively, as shown on the top left of Figure 5.1. There are four commonly used relationships between classes: generalization, dependency, association, and aggregation.

Generalization represents an "is-a" relationship between two classes, where one class (the subclass) is a specialized version of the other (the superclass). A generalization relationship is rendered in the UML as a solid line connecting the related classes with a hollow arrowhead pointing to the superclass. Dependency represents a unidirectional relationship from one class (supplier) to the other class (client), where the client has no semantic knowledge about the supplier but can still be affected by changes in specification of the supplier. Graphically, a dependency is rendered as a dashed line pointing from the client to the supplier. Association represents a link between the instances of two classes. An association relationship enables the navigation from an instance of one class to an instance of the class at the other end. An association is rendered in the UML as solid line that could have an arrow at one end indicating the direction of navigation. Aggregation is one special kind of association representing a "whole-part" relationship. An aggregation is rendered as a solid line with a diamond next to the class acting as the whole. The graphical representation of classes and their relationships are shown in Figure 5.1.

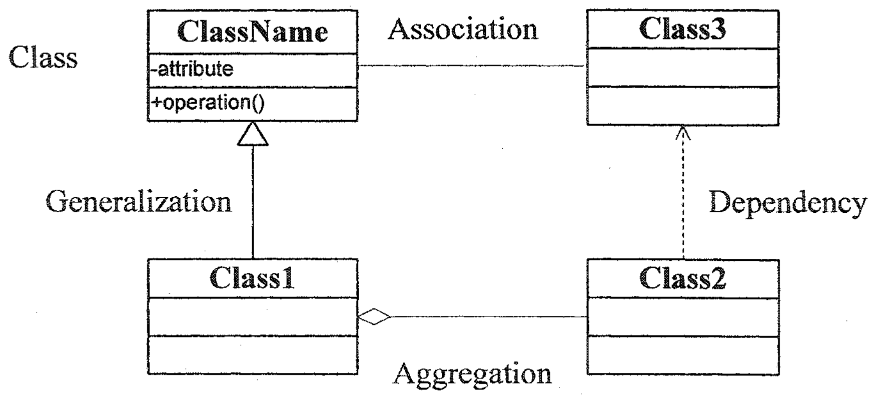


Figure 5.1 UML graphical representation of classes and their relationships

Since an OO framework intends to be reused by many applications belonging to the same problem domain, it requires the identification of abstract classes, their roles and collaborations, and the distribution of responsibilities. Design patterns may be employed to help in the design of the framework. Since a design pattern is a proved successful solution to a problem that occurs repeatedly in software design, a framework encompassing appropriate design patterns tends to be flexible and facilitates the communication throughout the development procedure. Gamma et al. (1995) presented a catalog of common design patterns. Only three of them are used in this research and are briefly presented here.

- Composite design pattern. The composite pattern allows building complex objects by recursively composing similar individual objects in a tree-like manner. With a common interface, both individual objects and complex objects are manipulated uniformly. The composite pattern is used when a whole-part hierarchy of objects exists and it is beneficial to ignore the difference between individual objects and their collections.

- Singleton design pattern. The singleton pattern ensures that a class has one instance and provides a single point of access and maintenance for that class. It is used when a system only needs one instance of a class and that instance needs to be accessible from many different parts of a system.

- Strategy design pattern. The strategy pattern decouples related algorithms from their host, and these algorithms are encapsulated in separate subclasses with a common superclass. The strategy pattern facilitates the change of algorithms in different situations. It is used when a class has many different behaviors to perform the same task.

5.2 Problem Domain Analysis

Problem domain analysis is an essential step prior to the design of an OO framework. Since many features of simulation-based green building optimization problems have been discussed in Chapter 3 and 4, this section focuses on the aspects of the framework that need to change across different applications. These aspects are the emphasis of the framework design.

The examination of the previous studies listed in Table 2.2 can identify the following aspects where simulation-based optimization problems are most likely to vary:

- The type and the number of variables. There are three types of variables: continuous, discrete, and structured as presented in Chapter 3. The number of variables considered varied from a minimum of four in Johnson et al. (1990) to a maximum of seventeen in Peippo et al. (1999).

- The number of objective functions. Optimization problems can be classified into single- and multi- objective optimization according to the number of objective functions. The majority of previous studies are single-objective optimization while most of the multi-objective studies have two or three objective functions.

- The existence of functional constraints or not. Optimization problems can be classified into unconstrained and constrained optimization depending on the presence of functional constraints. About 20 percent of previous studies have considered functional constraints.

- The simulation programs used for the evaluation of objective functions or functional constraints. About one third of previous studies employ external, executable simulation programs while the rest use self-developed simulation programs that need to be compiled.

- The optimization algorithms. A variety of optimization algorithms are used to find the optimal solutions, but genetic algorithms are used by more than half of the studies published since 2000.

- The simulation-optimization interface. External, internal, or hybrid interface may be used between the optimizer and the simulation programs, as discussed in Chapter 4.

Therefore, the framework design should provide ways to capture the above aspects that may change with different applications. Specifically, the OO framework design intends to reach the following objectives:

- The framework should be able to deal with continuous variables, discrete variables, and structured variables as well. There should be no limitation on the number of hierarchical levels for structured variables.

- The framework should be able to solve single- and multi- objective optimization problems with or without constraints.

- The framework should completely implement the GA so that framework users can use the optimizer directly without programming. In addition, the framework should be extendable to accommodate the important GA handling techniques proposed in the literature.

- Considering that the accessibility to the source code of building simulation programs varies, the framework should support the hybrid simulation-optimization interface.

5.3 OO Framework Design

A simulation-based optimization system requires the collaboration of several components as illustrated in Figure 4.1. The optimizer changes and transfers the values of variables to the simulation programs, which use these values to calculate the objective functions or functional constraints and return the results to the optimizer. The above procedure is iterated until a predefined stopping criterion is satisfied. Since text files are used for the inputs and outputs, the emphasis of the framework design is to model the entity classes and their relationships in the system. All entity classes in the object-oriented system framework are grouped into three modules: a variables module to define and organize variables, a simulation module to define and instantiate simulation programs, and an optimizer module to implement the optimization algorithms. In this section, each module is presented with the relevant classes and their relationships using class diagrams. For the purpose of simplification, the class diagrams follow the same conventions defined below:

- Not every attribute and operation of a class is shown in the diagrams. The types of attributes, the arguments and the return value types of operations are omitted.

The detailed documentation of all classes in the system is available in Appendix B of this thesis.

- The symbols "+", "-", and "#" correspond respectively to the public, private, and protected access modifiers. The public modifier means that a class member (e.g., attribute or operation) can be accessed from the outside; the protected modifier means that a member can be accessed by the class and its subclasses; and the private modifier means that a member can be accessed only by the class itself.

- Class names start with an upper case letter while attributes and operations start with a lower case letter. The attributes are named following the same conventions: the prefix "ptr" means a pointer, and the suffix "Vec" implies that a vector data structure is used.

- A class with an italic name implies that it is an abstract class (i.e., a class that cannot be instantiated to create its objects). An abstract class provides common features for its subclasses, which may have instances.

5.3.1 Variables Module

An optimization problem has some variables that may be continuous or discrete. The purpose of the variables module is to define variables and to organize them according to their relationships. Figure 5.2 shows the class diagram for the variables module.

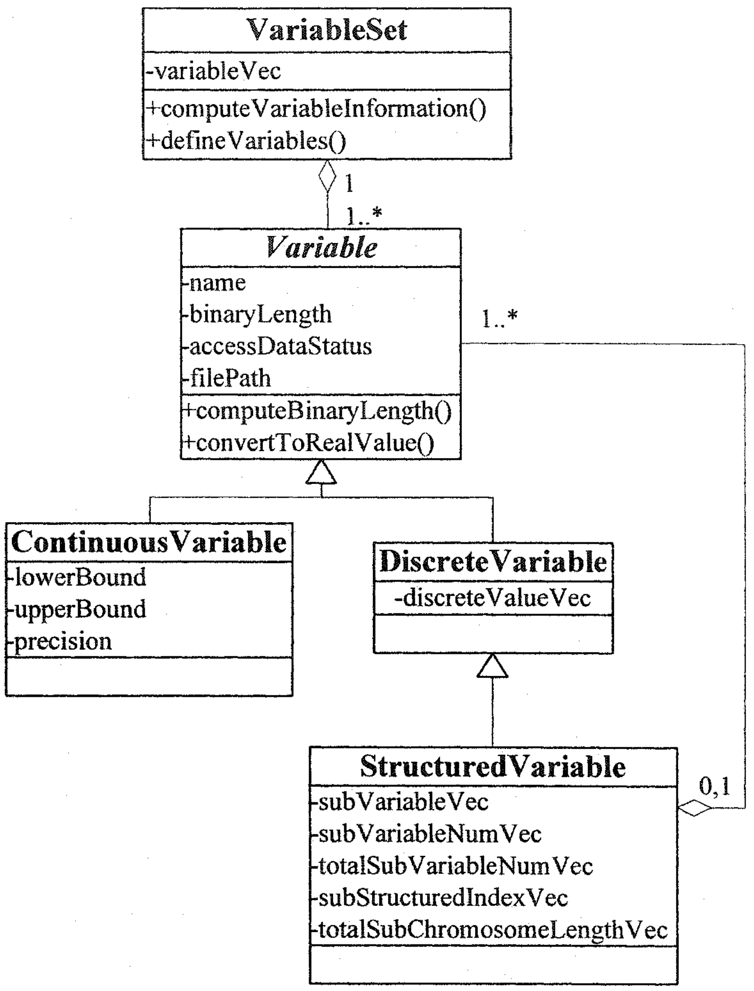


Figure 5.2 Class diagram for the variables module

The class *Variable* provides the attributes and operations that are applicable to all variable types. Its attribute *name* is used to distinguish variables. The attribute *binaryLength* defines the necessary length of a binary string to obtain the required precision for a continuous variable or to accommodate the number of available values for a discrete variable. For simulation-based optimization problems, a discrete variable that represents an entity with many parameters may take values standing for indices in an external data source such as the data files presented in Chapter 4 for this research. For example, a window type is a discrete variable, for which each value is an index in a data file for window types storing many parameter values such as embodied energy, construction cost, solar transmittance and absorptance at a number of incident angles. A continuous variable may also need to access an external data source to define the parameters of the represented entity. Therefore, the attribute *accessDataStatus* is used to indicate whether a data source needs to be visited in the simulation process. If so, the path of the data source is stored in the attribute *filePath*. The operation *computeBinaryLength* computes the binary string length of a variable and assigns the value to the attribute *binaryLength*. The operation *convertToRealValue* decodes a variable from a binary string to a real value. Both of the above two operations depend on the variable type, so they are overridden in each subclass of *Variable*.

The subclasses of *Variable* define the variable types considered in the framework design. The classes for continuous variables and discrete variables are derived directly from *Variable*. The class for structured variables is derived from the class *DiscreteVariable*. This is because a structured variable can only take a few discrete values. The composite design pattern is applied here to ensure that there is no limitation on the number and the type of variables contained by a structured variable. This means that multi-level structured variables are allowed within this framework.

The classes *ContinuousVariable* and *DiscreteVariable* have simple interfaces. Lower bound, upper bound, and precision are the three necessary attributes to define a continuous variable. The vector of discrete values that a discrete variable can take is an attribute in *DiscreteVariable*. In contrast to *ContinuousVariable* and *DiscreteVariable*, the class *StructuredVariable* has more attributes and operations.

The design of *StructuredVariable* needs to facilitate the traversal of sub-level variables in different situations. A structured variable can take several discrete values, each of which is regarded as an alternative. Each alternative usually consists of a group of sub-level variables, and some of these variables may be structured. For example, the structured variable V1 shown in Figure 5.3 has two alternatives. The first alternative has three variables at the next level: V2, V3, and V4, of which V3 is a further structured variable with two sub-level variables (V7 and V8). The second alternative has two variables at the next level: V5 and V6, neither of which is structured. The attribute *subVariableVec* is used to sequentially store all variables at the next level, which are the five variables from V2 to V6 for the example. When traversing the sub-level variables of a structured variable, there exist two different situations: (1) all its variables at the next level need to be examined when an alternative is active; and (2) all its sub-level variables can be skipped when an alternative is inactive. To facilitate the operation of jumping from one member to the other in the *subVariableVec*, the attributes *subVariableNumVec* and *totalSubVariableNumVec* are used to store the number of sub-level variables based on two different counting procedures: counting the variables at the next level only; and counting all the sub-level variables. For the example shown in Figure 5.3, the values in the vector *subVariableNumVec* of structured variable V1 are 3 and 2, but they are 5 and 2 in *totalSubVariableNumVec* because the first alternative has two indirect sub-level variables V7 and V8. The feature of multi-level structured variables makes it necessary to know whether some sub-level variables are structured. If so, mechanisms should be provided to conveniently locate the structured sub-level variables. Thus, the attribute *subStructuredIndexVec* is used to store the indices of those variables that are of the structured type at the next level. In the example of Figure 5.3, the vector *subStructuredIndexVec* contains the index of variable V3, which is equal to 1 since the indices start from zero for V2. The attribute *totalSubChromosomeLengthVec* stores the value of the total binary string length for each alternative of a structured variable in order to facilitate the calculation of the effective chromosome length used by the modified crossover operator. The binary string length considered here includes all sub-level variables. For example, the attribute *totalSubChromosomeLengthVec* of variable V1 in Figure 5.3 contains two values, which are for the total binary string length of all sub-level variables affiliated to the alternative 1-1 and 1-2, respectively.

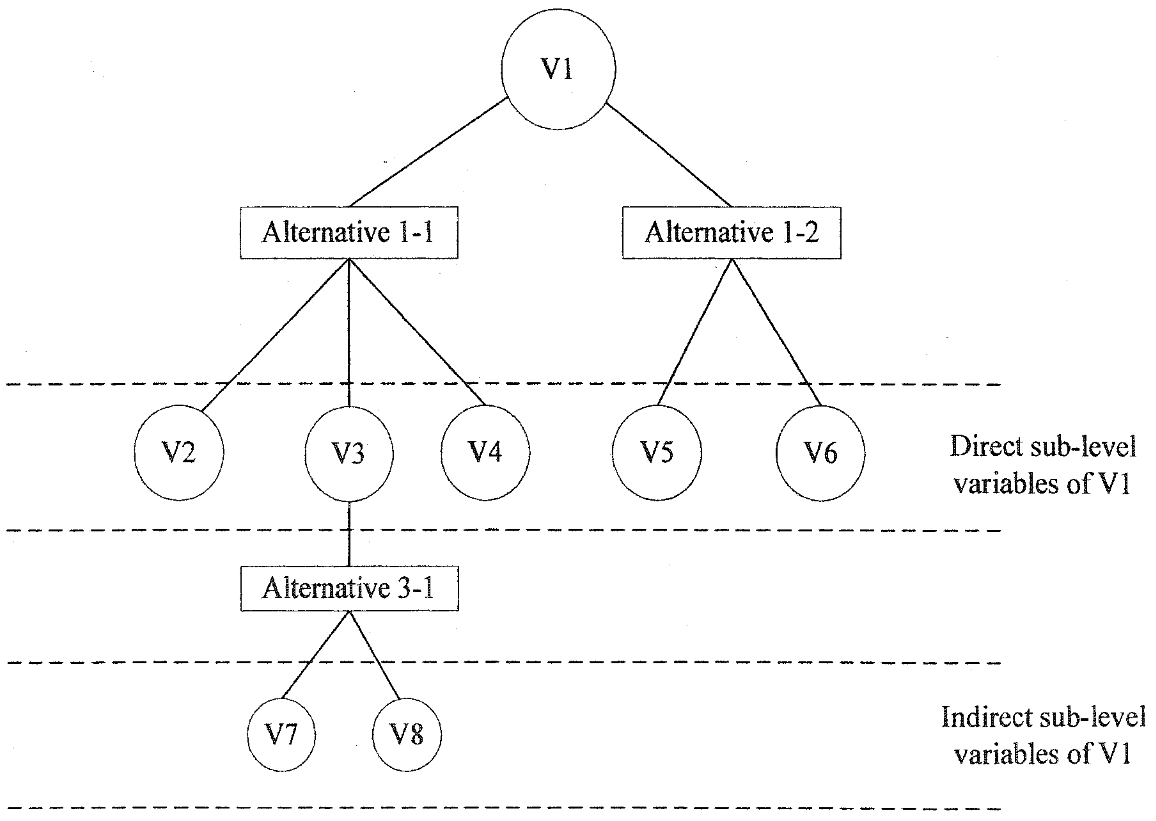

V1
Alternative 1-1
Alternative 1-2
V2
V3
V4
V5
V6
Direct sub-level variables of V1
Alternative 3-1
V7
V8
Indirect sub-level variables of V1

Figure 5.3 Illustration of a multi-level structured variable

All variables are contained in an instance of the class *VariableSet*. The singleton design pattern is employed because: (1) there is only one variable set for a given optimization problem; and (2) variables need to be accessed at many places in the simulation module and the optimizer module. It must be noted that the attribute *variableVec* contains only top-level variables, which include the variables at the root level of hierarchies and the variables not located at any hierarchical levels. Sub-level variables need to be accessed indirectly through the structured variables at the root level. The values of those attributes specific to structured variables are calculated by the operation *computeVariableInformation* in the class *VariableSet*. In addition to store the variables, the class *VariableSet* is responsible to create variable instances based on the text file *environment.txt* through the operation *defineVariables*. The basic information for variable definition is provided in the text file *environment.txt* according to the predefined format to be described in the next section.

5.3.2 Simulation Module

The purpose of the simulation module is to evaluate objective functions and functional constraints. The class diagram of the simulation module is shown in Figure 5.4.

As the container for all simulation programs required by the system, the class *SimulationSet* provides access to simulations. The singleton design pattern is used for the class *SimulationSet* because of two reasons: first, it has only one instance for a given optimization problem; second, some information about the simulation module (e.g., the number of objective functions and constraints) is required by a number of classes in the optimizer module.

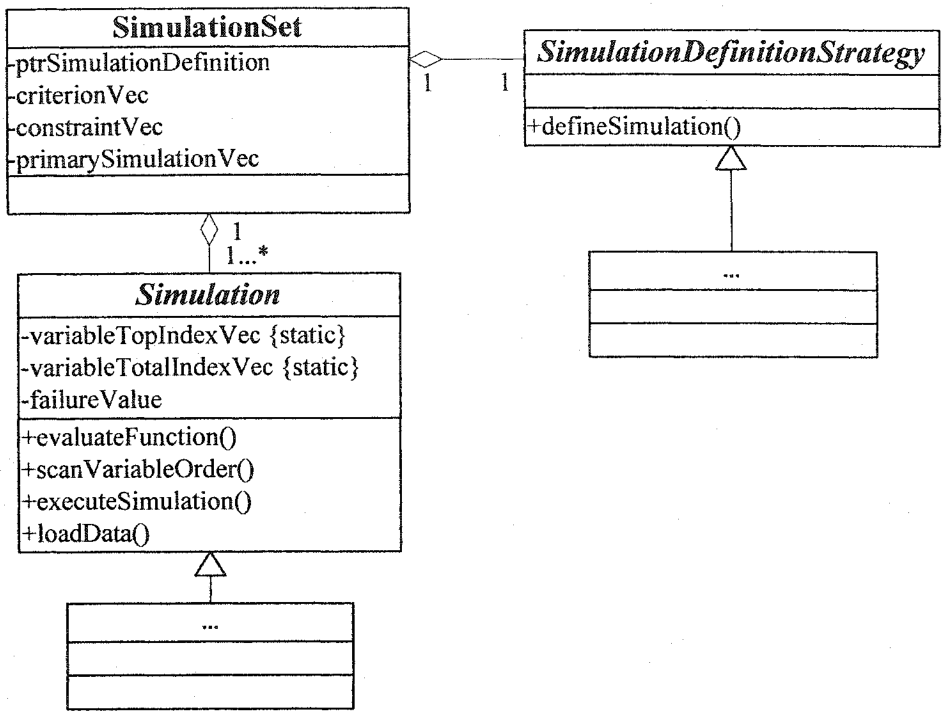


Figure 5.4 Class diagram for the simulation module

The class *SimulationSet* organizes simulations according to their usages in the optimization. Since a simulation program is used for either an objective function or a functional constraint, the class *SimulationSet* contains two vectors, *criterionVec* and *constraintVec*, to store the instances of simulations to calculate objective functions and functional constraints, respectively. Another vector *primarySimulationVec* is used to record primary simulations. A primary simulation is simulation program that is computationally expensive and whose results are essential to calculate a number of objective functions or functional constraints. In this study, for instance, the energy simulation is a primary simulation program. The attribute *primarySimulationVec* is necessary because in multi-objective or constrained optimization problems, the results from a primary simulation might be post-processed in different ways to compute more than one objective function or functional constraint. In such a situation, it is important to ensure that the primary simulation is evaluated only once for each transferred variable list because the primary simulation is a computation-intensive program. This objective can be achieved through the following measures collectively: (1) the primary simulation is executed first by the operation *executeSimulation* before all other simulation programs; (2) the primary simulation results are stored for later use; and (3) the classes corresponding to those objective functions and constraints that use the primary simulation results are inherited from the class corresponding to the primary simulation.

In addition to its role as the container for simulations, the class *SimulationSet* is also responsible to define objective functions and functional constraints for an optimization problem. Because each objective function and functional constraint usually corresponds to one concrete subclass derived from *Simulation*, the *SimulationSet* needs to know the name of the corresponding classes to create the simulation instances. Therefore, this is the portion of the framework design that changes the most with applications. The strategy design pattern is employed to decouple the definition of objective functions and functional constraints from their host, the class *SimulationSet*. Framework users can customize the framework for an optimization problem by creating a subclass of the *SimulationDefinitionStrategy* class to override the operation *defineSimulation*.

The class *Simulation* is the base class to provide the common attributes and operation for all simulation programs. It needs to be specialized into concrete subclasses for a particular application to calculate function values required by the optimizer. Each subclass usually corresponds to one objective function or functional constraint. Several subclasses may be needed for multi-objective or constrained optimization problems. The derived subclasses from *Simulation* may perform different tasks depending on the source of the simulation program. If the simulation is an external and executable program, the derived subclasses need to prepare the input file for the primary simulation and to call it via the operating system. If the simulation is developed by the framework user, the derived subclasses are actually the implementation of the simulations.

The vector of variable values is transferred from the optimization program as arguments of the operation *evaluateFunction* or *executeSimulation* in the class *Simulation*. The operation *evaluateFunction* is used when the simulation program evaluates an objective function or a functional constraint; the operation *executeSimulation* is used when the simulation program is a primary simulation to calculate the intermediate results used by other simulations. The positions of the top-level variables in the transferred vector is determined once by the operation *scanVariableOrder*, which maps the order of the top-level variables defined in the text file *environment.txt* to the order of variables programmed in the simulation by the framework user. The mapping operation is performed by matching the name of the variables. Due to the existence of structured variables, the positions of top-level variables need to be specified in two locations: in the top-level variable list only and in the total variable list including sub-level variables, which correspond to the two attributes *variableTopIndexVec* and *variableTotalIndexVec*, respectively.

Running errors might happen in the simulation process due to two possible reasons. First, the simulation may have some undetected programming errors. Second, a design alternative is prohibited by the simulation program; for example, a building with insufficient insulation may be regarded as an infeasible design alternative by some energy simulation programs if it causes condensation problems. In many cases, it is almost impossible to eliminate or to avoid running errors because: (1) the source code of simulation programs is not accessible; and (2) infeasible design alternatives are not known in advance. Therefore, in a simulation-based optimization system, it is essential to find a way to circumvent running errors so that optimization can continue without being stalled. This is addressed by the attribute *failureValue* in the *Simulation* class. A simulation program will return a *failureValue* to the optimizer if a running error happens. The actual value of the attribute *failureValue* must be able to distinguish between a normal simulation process and a process with running errors. For example, in most building optimization problems, the attribute *failureValue* can be set equal to -1 since objective functions usually do not take negative values.

A simulation program may require some constant design parameters defined by the user; for example, energy simulation programs require constant parameters such as the indoor design temperature and the climatic data. These constant design parameters are usually specified in the input file for executable simulation programs having an external interface with the optimizer. If there are user-developed simulations having an internal or a hybrid interface with the optimizer, the operation *loadData* in the *Simulation* class can be used to read these parameter values from an external data source (e.g., the file *simulationData.txt* in this study). The loaded data are stored in the attributes of the derived classes corresponding to the implemented simulations. Hence, these data are read into memory once for the whole optimization process.

5.3.3 Optimizer Module

The purpose of the optimizer module is to implement optimization algorithms. As can be seen from the class diagram shown in Figure 5.5, the class *Optimizer* provides two general operations: (1) the operation *defineOptimizationParameter* to read optimization parameters from a data file and assign their values to the attributes of the classes in this module; and (2) the operation *optimize* to execute optimization algorithms. The current optimizer module implements only the GAs (genetic algorithms), but it is possible to implement other optimization algorithms by specializing the class *Optimizer*. The GA optimizer has been designed to achieve two objectives:

- to be able to solve different problems such as single- or multi- objective, and constrained or unconstrained optimization problems;
- to be extendable in order to incorporate different handling techniques for genetic algorithms.

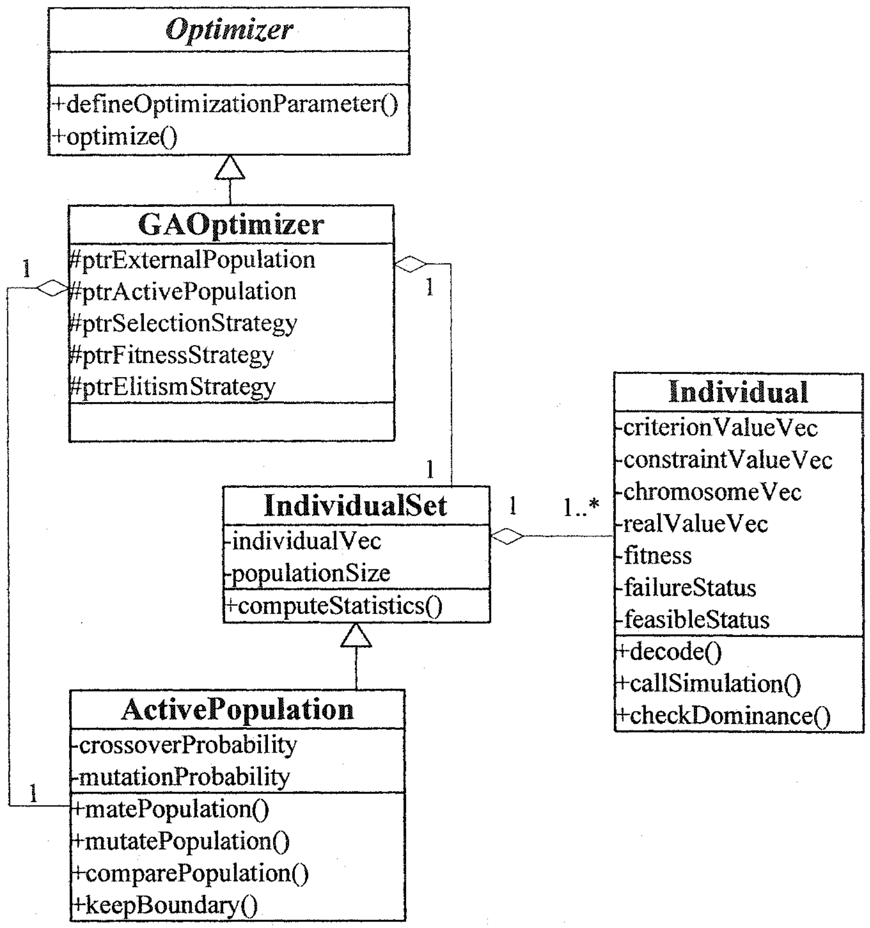


Figure 5.5 Class diagram for the optimizer module

The individual is an important concept in genetic algorithms and is represented as an entity class to store the relevant information about a chromosome. The class *Individual* can be regarded as the link between the GA optimizer and the other two modules in the framework. The GA optimizer works on the genotype space defined by the chromosome and its fitness while the other two modules work on the phenotype space defined by the variable values and the objective function values. The linking role played by the class *Individual* is reflected through its attributes and operations. The attributes *chromosomeVec* and *realValueVec* represent the binary string and the values of the variables, respectively. The conversion between them is performed by the operation *decode*. This operation sends the whole binary string and the starting position to decode as function parameters to the operation *convertToRealValue* for each variable. Using the decoded real values, the operation *callSimulation* fulfills the following tasks: (1) to call each primary simulation; (2) to call the simulation program for each objective function and store its value in *criterionValueVec*; and (3) to call the simulation program for each functional constraint and store its value in *constraintValueVec*. The pseudocode of *callSimulation* is given in Figure 5.6. The flag *failureStatus* determines whether an individual has caused running errors in the simulation. The flag *feasibleStatus* determines whether an individual has violated some functional constraints. These two flags need to be dealt with separately because they imply different robustness of individuals, as measured by the attribute *fitness*. An individual with running errors is usually given the worst fitness value while an individual with constraints violation is assigned a fitness value according to some refined strategies such as the penalty function approach (Deb 2001).

```
for each primary simulation
  call the executeSimulation
end for

for each criterion
  call the evaluateFunction
  if the function value is equal to the failure value
      set failureStatus to true
      return
  else
      store the function value in the criterionValueVec
end for

for each constraint
  call the evaluateFunction
  if the function value is equal to the failure value
      set failureStatus to true
      return
  else
      store the function value in the constraintValueVec
      if the function value <0
          set feasibleStatus to false
      endif
  endif
end for
```

Figure 5.6 Pseudocode for the operation *callSimulation* in the class *Individual*

The genetic algorithm operates on a population of individuals. Besides this population, an external population has also been used in many multi-objective genetic algorithms proposed in recent years to improve the performance (Tan et al. 2002). These two populations perform different roles in the evolution process. The active population is the main population that actively participates in the evolution process through selection, crossover, and mutation. The external population is a secondary population that stores the elites found during the evolution process. The individuals in the external population do not mate and mutate, but they may affect the active population (Zitzler and Thiele 1999).

Since both the active population and the external population include a number of individuals, they require similar functionalities, which are provided by the superclass *IndividualSet*. This class contains a set of individuals. It supports the operations to add an individual to *individualVec*, to replace a member with another one, and to empty the *individualVec*. *IndividualSet* also computes the population statistics such as the average, the standard deviation, the minimum, and the maximum value for each criterion.

Derived from *IndividualSet*, the class *ActivePopulation* provides several additional operations. The operation *matePopulation* applies crossover to the population of individuals. It has a function parameter that determines whether the mating restriction mentioned in Chapter 4 is used. The operation *mutatePopulation* applies mutation to the population. The operation *comparePopulation* compares a new generation of population with its previous generation. If an individual in the new generation has the same chromosome as another individual in the previous generation, its objective function values and functional constraint values can be directly copied from the previous generation without calling the simulation programs. Since simulation usually dominates the computation time, the purpose of the *comparePopulation* operation is to improve the efficiency of the optimization system. The operation *keepBoundary* aims to preserve boundary solutions. A boundary solution performs best for at least one of the considered criteria, so it is the outmost solution along the Pareto front. Since boundary solutions play positive roles in extending the spread of the Pareto front, it is worthwhile to take measures to prevent their loss in the evolution (Zitzler et al. 2001). The operation *keepBoundary* protects boundary solutions as follows: the boundary solutions in a new generation of population are replaced by those in the previous generation if the former is dominated by the latter. For example, the Pareto solutions of two neighboring generation of populations are shown in the performance space as circles and squares in Figure 5.7. For the minimization of two performance criteria, n1 and n4 are the boundary solutions in the new population; p1 and p4 are the boundary solutions in the previous population. Of these four boundary solutions, n1 is dominated by p1 while n4 is not dominated by p4. In this case, p1 will take the place of n1 in the new population.

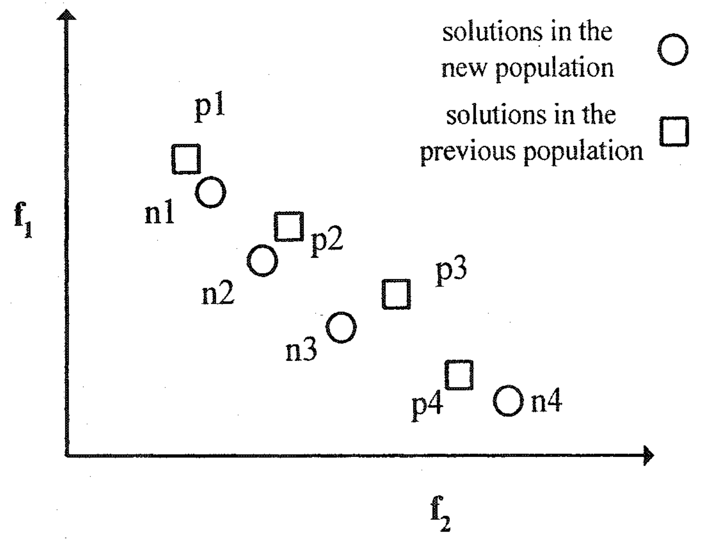


Figure 5.7 An example to keep boundary solutions

The basic genetic algorithm can be improved with one or more handling techniques as described by Tan et al. (2002). A given handling technique can be implemented in different ways. For example, elitism is realized through the external population in the strength Pareto evolutionary algorithm (Zitzler and Thiele 1999) while it is realized through choosing the best members from two neighboring generations to form a new population in the elitist non-dominated sorting GA (Deb 2001). Since a handling technique has different behaviors to obtain the same purpose, the strategy design pattern is an appropriate choice.

The strategy design pattern is applied to three handling techniques in the current version of the framework. There is a selection strategy for selecting individuals from populations, a fitness assignment strategy for assigning fitness to individuals, and an elitist strategy for preserving elites. Other handling techniques could be added to the framework similarly using the strategy design pattern. Each of the three implemented strategies has a base class acting as the public interface while the detailed methods are defined in derived classes. Several commonly used methods have been implemented for the three strategies as shown in Figure 5.8.

- There are five implemented selection methods: tournament (binary tournament selection), RWS (roulette wheel selection), SUS (stochastic universal selection), SRS (stochastic random selection), and *insertTournament* (insert tournament selection). The first four selection methods were explained in the previous chapter. The last selection method introduces a number of elites from the external population into the mating pool and selects the remaining individuals from the previous genetic population with the binary tournament method. The insert tournament selection can accelerate the convergence of GA because of the impact of elites.

- There are four implemented fitness assignment techniques: *MaxSOGA, DebConstrainedSOGA, RankMOGA,* and *FrontNSGA*. *MaxSOGA* assigns the difference between the maximum objective function value in the population and the function value of an individual as its fitness; it is used for unconstrained single-objective optimization. *DebConstrainedSOGA* combines the previous *MaxSOGA* and the technique proposed by Deb (2000) to purposely prefer feasible solutions over infeasible solutions; it is used for constrained single-objective optimization. *RankMOGA* implements the rank-based Pareto dominance scheme as presented in the previous chapter; it is used for unconstrained multi-objective optimization. *FrontNSGA* implements the front-based Pareto dominance scheme as presented in the previous chapter; it is also used for unconstrained multi-objective optimization.

- There are two implemented elitist strategies: *ConstrainedSOGAElitism* and *ClusterMOGAElitism*. *ConstrainedSOGAElitism* replaces a few worst individuals in the new population with the same number of best individuals in the previous population; it is used for single-objective optimization. *ClusterMOGAElitism* employs an external population with a predefined capacity to store non-dominated solutions; it is used for multi-objective optimization. The clustering technique used by Zitzler and Thiele (1999) is applied in the *ClusterMOGAElitism* to remove elites located in crowded regions after the external population reaches its capacity.

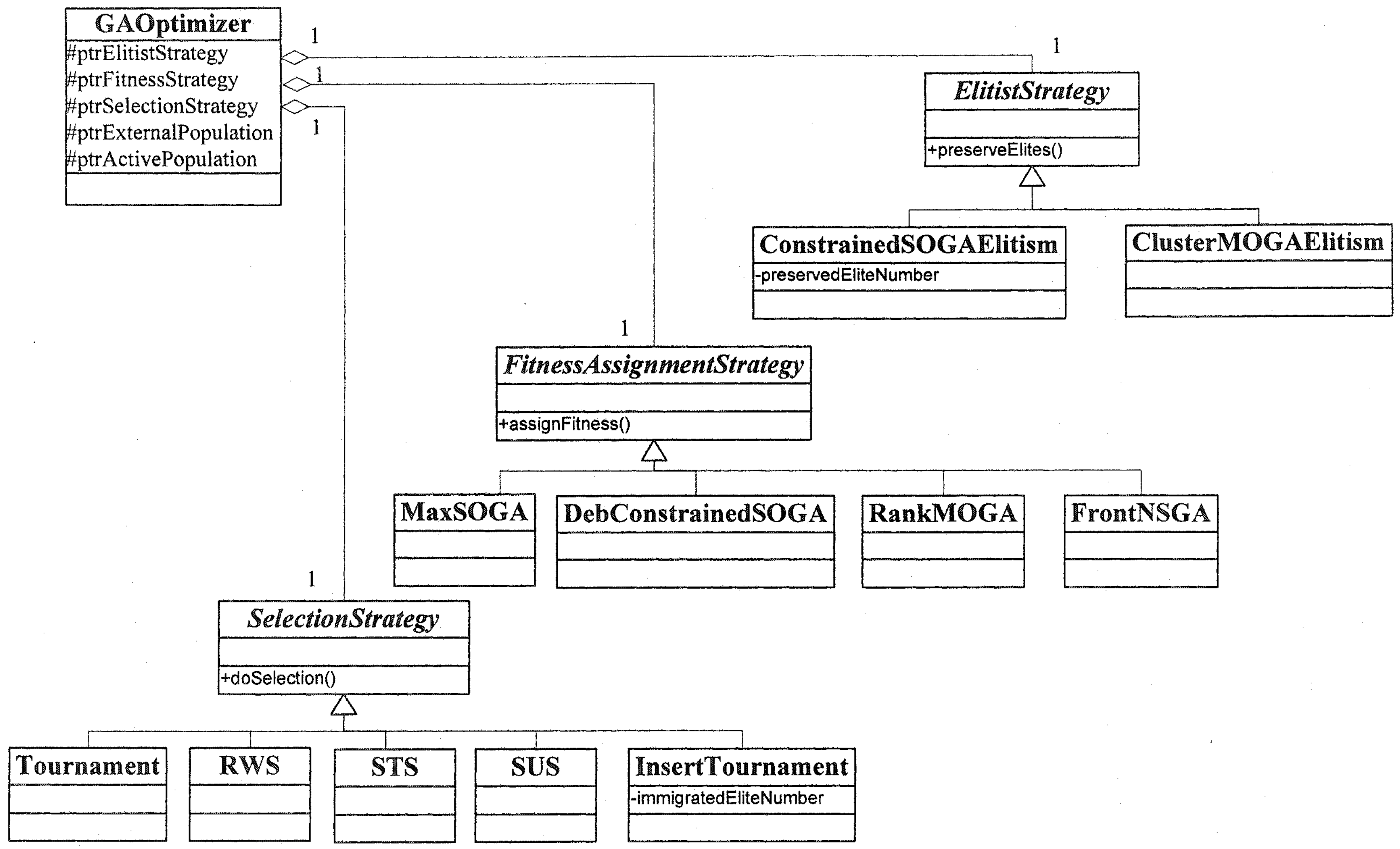


Figure 5.8 Class diagram for the GAOptimizer and its associated strategy patterns

The class *GAOptimizer* stores a reference to the superclass for each strategy. It also overrides the operation *optimize* to implement both single- and multi- genetic algorithms. The behavior of the implemented genetic algorithms depends on how the instances of *FitnessAssignmentStrategy*, *SelectionStrategy*, and *ElitistStrategy* are used. Framework users can change that behavior by indicating different methods for selection, fitness assignment, and elitism in the text file *environment.txt*, as will be discussed in the next section.

Two approaches are feasible to support both single- and multi- objective genetic algorithms. First, relevant classes such as *Individual*, *ActivePopulation*, and *GAOptimizer* could provide operations and attributes for both types of genetic algorithms. Second, separate classes could be created for single- and multi- objective GAs, respectively. The first approach is adopted in this study because: (1) it simplifies the framework by reducing the number of classes; and (2) single- and multi- objective GAs are different mostly in terms of the selection, fitness assignment and elitism, which have been dealt as classes. Since the first approach is adopted, operations and attributes in some classes of this module may serve only multi-objective GA. Two examples are the attribute *ptrExternalPopulation* in the class *GAOptimizer* used to store elites and the operation *checkDominance* in the class *Individual* used to check whether one individual dominates another.

5.4 Framework Customized to the System GBOptimizer

Inheritance and composition are the two main mechanisms to customize a framework for a particular application (Froehlich et al. 1999). Inheritance involves specializing superclasses by overriding some of their predefined operations and adding new functionality to the superclasses. Inheritance from a class requires framework users to know about the details of the involved superclass and its interactions with other classes. In addition, the creation of new subclasses requires programming. The advantage of inheritance lies in its extensibility because framework users can easily add new functionality to a subclass derived from an existing class. Composition involves setting parameters in a data file and calling available components of the framework to achieve the desired functionality. With composition, framework users do not need to have an in-depth understanding of the classes in the framework. However, composition has less flexibility than inheritance. Generally, composition is used when the framework has well-defined interfaces, whereas inheritance is used when the framework does not have a full range of functionality that can be anticipated.

Both inheritance and composition are needed to customize the OO framework for the simulation-based optimization system GBOptimizer in this research. Composition is used to customize the variables module; inheritance is used to customize the simulation module; and both composition and inheritance can be used to customize the optimizer module. The customization of each module and the system integration are discussed in this section.

5.4.1 Variables Module Customization

The classes in the variables module do not need to be extended. Since the attribute values required to define variables are read from the text file *environment.txt*, the customization work for the variables module is restricted to the preparation of that file.

The text file *environment.txt* is a comma delimited file including three parts for variables, simulation programs, and optimization parameters. The three parts can be in any sequence. They have their own syntaxes, but abide by the same rules as follows:

- Each statement line is ended with a semicolon.
- Comments preceded by the exclamation mark are permitted, but they must be placed between statement lines, not data items.
- A word in courier font indicates a keyword in the text file; the keyword is case-insensitive.
- Text within square brackets [ ] indicates optional values.
- Vertical bar | separates possible entries. Only one of the entries that are separated by | is allowed.

The part for variables starts with "#Variables Begin#" and finishes with "#Variables End#". Each variable is defined in this part according to the following syntax:

`Continuous`, variable name, `Local` | `Remote`, [data file path,] lower bound, upper bound, precision,;

or

`Discrete` | `Structured`, variable name, `Local` | `Remote`, [data file path,] discrete value 1, [discrete value 2, …, discrete value n,] ; [&]

In the above syntax, the words "Continuous", "Discrete" and "Structured" represent the three variable types considered. A continuous variable should indicate its name, boundary values, and precision. A discrete or structured variable should indicate its name and discrete values. The word "Local" or "Remote" indicates whether a data file needs to be accessed to define a variable during the simulation. If the word "Remote" is used, the subsequent field for the data file path must be provided; otherwise, the field is left out. The symbol '&' is necessary to indicate the end of each alternative for a structured variable because: (1) an alternative of a structured variable may have several sub-level variables; and (2) the number of sub-level variables may be undetermined in advance. For example, *wallType* is a structured variable, which may take a few discrete values. Each discrete value represents a certain wall type, which usually has many layers. In this case, the statement line defining the innermost layer for each wall type should be ended with the symbol '&'.

5.4.2 Simulation Module Customization

The customization of the simulation module involves deriving subclasses from *Simulation* and implementing simulation programs in the derived classes to calculate objective functions and functional constraints. The operations of *Simulation* may be overridden or not, depending on the simulation program implemented in the subclass. If the simulation program corresponds to an objective function or a functional constraint, the operation *evaluateFunction* must be overridden; if the simulation is dealt as a primary simulation, the operation *executeSimulation* must be overridden; if the simulation program reads design parameters from an external source, the operation *loadData* must be overridden. In addition to deriving subclasses from *Simulation*, a subclass also needs to be derived from the class *SimulationDefinitionStrategy*. The operation *defineSimulation* must be overridden to create the required simulation instances.

The system GBOptimizer supports two major objective functions: LCC (life-cycle cost) and LCEI (life-cycle environmental impacts). It also supports several other commonly used objective functions: IC (initial cost), OC (operating cost), LCEnergy (life-cycle energy), and AnnualEnergy (annual operating energy consumption), as presented in Chapter 3. Since all the possible objective functions are related to either cost or environmental impacts, two classes, *CostSimulation* and *ImpactSimulation*, need to be derived from the class *Simulation*. These two classes inherit from the same class *EnergySimulation*, a direct subclass of *Simulation*, in order to share the results obtained from the energy simulation program. If a functional constraint and an objective function depend on the same simulation program, their only difference is that the former has a right hand value and accompanying relational operations while the latter does not have. Therefore, the classes *CostConstraint* and *ImpactConstraint* are derived from *CostSimulation* and *ImpactSimulation* to implement the functional constraints related to cost and environmental impacts, respectively. Besides the extension of the *Simulation* class, the *SimulationDefinitionStrategy* is extended with its subclass *EnergyDefinitionStrategy*. Figure 5.9 shows the class diagram of the extended simulation module for this system.

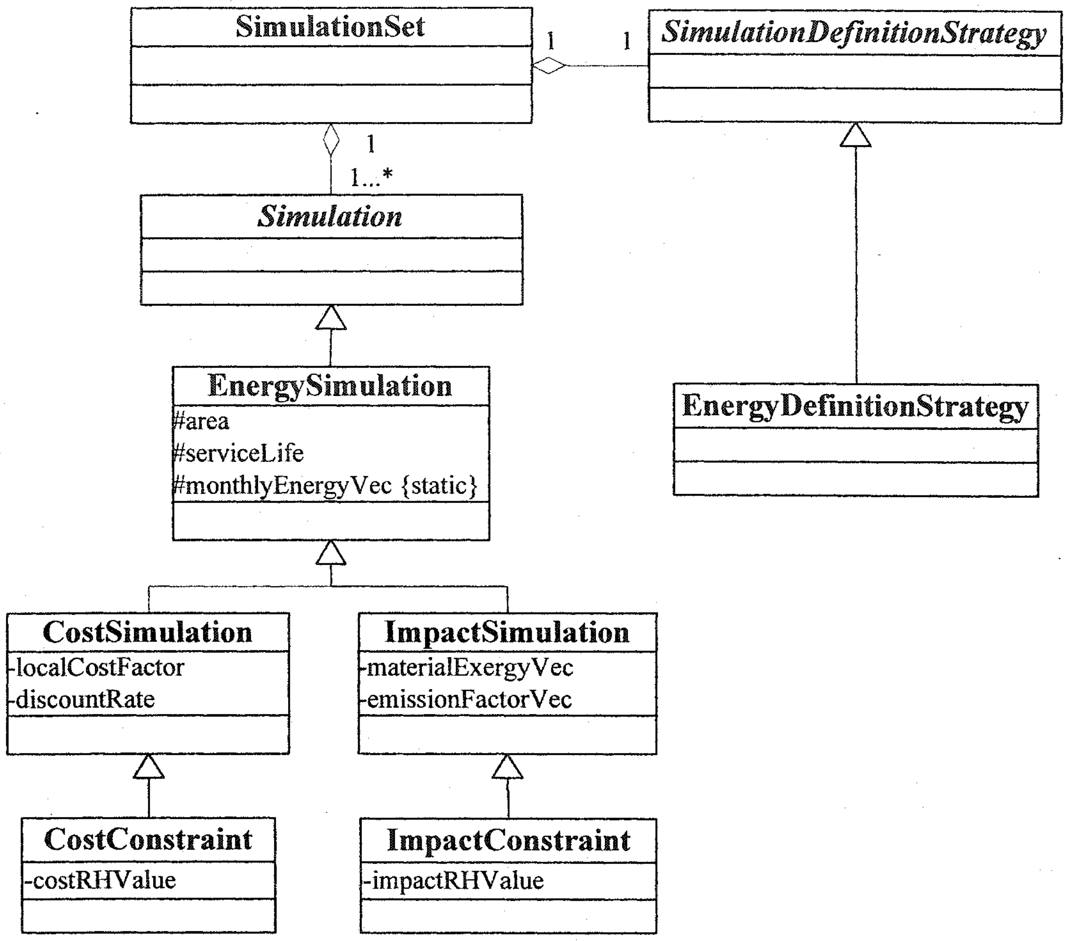


Figure 5.9 Extensions of the simulation module for the system GBOptimizer

Four operations from the class *Simulation* are overridden in the class *EnergySimulation*. The operation *scanVariableOrder* is overridden to determine the order of the top-level variables considered by the system. The operation *loadData* is redefined to read design parameters such as building area and mechanical system efficiency from the file *simulationData.txt*. The operation *executeSimulation* is overridden to calculate the energy consumption for each month in the heating and cooling seasons. It calls the ASHRAE toolkit and stores the results of monthly energy consumption in the attribute *monthlyEnergyVec*. The operation *evaluateFunction* is also overridden in the class

EnergySimulation because the annual operating energy consumption is treated as a possible objective function in this system.

The operations *loadData* and *evaluateFunction* are overridden again in *CostSimulation* and *ImpactSimulation*, which are the subclasses of *EnergySimulation*. In these two subclasses, the operation *loadData* reads cost and environmental impact related data from the file *simulationData.txt* and assigns these data values to the corresponding attributes. The operation *evaluateFunction* calls the appropriate subroutine based on the simulation identifier (e.g., LCC and AnnualEnergy as explained below in the syntax) specified in the simulation program part of the *environment.txt* file. The operation *evaluateFunction* is also overridden in both classes *CostConstraint* and *ImpactConstraint* to consider the right hand value of a constraint.

The text file *environment.txt* has a part for simulation programs. This part is necessary to customize the system GBOptimizer to an optimization problem because a list of objective functions and functional constraints are available for selection. However, that part is unnecessary when an application has fixed objective functions and constraints and each of them corresponds to a subclass derived from *Simulation*. The part of the file for simulation programs starts with "#Simulation Program Begin#" and finishes with "#Simulation Program End#". It has the following syntax:

`Criterion | Constraint | Primary Simulation`, simulation identifier, [right hand value,] failure value, simulation data file path | NULL;

The above syntax shows that five data fields are needed to define a simulation program.

Each field is explained below.

- The first data field determines the usage of a simulation program. The word "Criterion" means that the simulation program calculates an objective function; the word "Constraint" means that the simulation program calculates a functional constraint; the phrase "Primary Simulation" means that the simulation program is computationally expensive and its results are essential to calculate other objective functions or functional constraints.

- The second data field identifies the simulation program to be called by the optimizer. The values of this data field depend on the first data field. If the first data field indicates a primary simulation, "AnnualEnergy" (annual operating energy consumption) is the only choice for this data field; otherwise, the candidate values of this data field include "LCC", "LCEI", "IC" (initial cost), "OC" (operating cost), "LCEnergy" (life-cycle energy), and "AnnualEnergy".

- The third data field exists only for functional constraints. It defines the right hand value or the upper bound value for a functional constraint.

- The fourth data field indicates the failure value that should be returned to the optimizer when running errors happen in the simulation process.

- The last data field indicates whether the simulation program requires user-defined parameters. If so, the path for the file storing parameter values is provided in this field; otherwise, the word "NULL" is provided.

5.4.3 Optimizer Module Customization

The classes in the optimizer module do not need to be extended if the implemented genetic algorithms and handling techniques are used as is. In this case, composition is used to customize this module by specifying the optimization parameters and applicable strategies in the text file *environment.txt*. Otherwise, inheritance should be used to derive new classes from the framework. The optimizer module could be extended with inheritance in the following ways: (1) derive a subclass from *GAOptimizer* to implement a new genetic algorithm; (2) derive a subclass from one of the three base classes for strategies to implement a new method for selection, fitness assignment, or elitism; and (3) define another base class like *SelectionStrategy* to incorporate a new handling technique.

Since the system GBOptimizer uses the genetic algorithms and handling techniques implemented in the framework, the optimizer module is customized by specifying the GA parameter values in the text file *environment.txt*. The part for optimization parameters starts with "#Optimization Parameter Begin#" and finishes with "#Optimization Parameter End#". It has the following sequence and syntax:

`Generation`, a number,;

`Population Size`, a number,;

`Crossover Probability`, a number between 0 and 1,;

`MateRestrictionStartGeneration`, a number,;

`Mutation Probability`, a number between 0 and 1,;

Selection, RWS | Tournament | SUS | SRS | insertTournament, [a number,];

Fitness Assignment, MaxSOGA | DebConstrainedSOGA | RankMOGA | FrontNSGA,;

Elitist Strategy, NULL | ConstrainedSOGAElitism | ClusterMOGAElitism, [a number,];

The above statements are explained as follows.

- The first statement declares the maximum number of generations used as a stopping criterion.
- The second statement gives the population size which remains constant throughout the evolution.
- The third statement sets the crossover probability. Crossover probability is usually as large as about 0.9 for a good performance of GAs.
- The fourth statement indicates the generation number after which mating restriction is applied. If the number is greater than the maximum number of generations, no mating restriction is employed throughout the evolution process.
- The fifth statement sets the mutation probability. Mutation probability usually takes a small value. A general guideline is to set its value equal to the inverse of the chromosome length (Deb 2001).

- The sixth statement specifies the selection method from five choices: tournament (binary tournament selection), RWS (roulette wheel selection), SUS (stochastic universal selection), SRS (stochastic random selection), and *insertTournament*. If the *insertTournament* selection is used, the number of elites from the external population must be specified in the last data field of this statement; otherwise, the last data field is left out.

- The seventh statement defines the method to assign fitness to individuals. Four choices are *MaxSOGA, DebConstrainedSOGA, RankMOGA*, and *FrontNSGA*. Their application scopes have been discussed in the previous section.

- The last statement deals with elitism. There are three options for elitism: *NULL, ConstrainedSOGAElitism*, and *ClusterMOGAElitism*. *NULL* indicates that no elitism is employed. *ConstrainedSOGAElitism* and *ClusterMOGAElitism* are respectively for single- and multi- objective optimizations, as have been discussed in the previous section. If the option *NULL* is selected, the last field of this statement is left out; otherwise, a number must be specified in the last field. This number indicates the number of preserved elites from the previous generation and the capacity of the external population, respectively, for *ConstrainedSOGAElitism* and *ClusterMOGAElitism*.

5.4.4 System Integration

The three customized modules for variables, simulation programs, and optimization algorithms are integrated together with a control class called *SystemControl*. The high-level interactions between the three modules and the control class are illustrated using the sequence diagram shown in Figure 5.10. Based on the part for variables in the file *environment.txt*, the variables module creates the instances of variables through the operation *defineVariables* in the class *VariableSet*. After the variables are created, the control class calls the operation *computeVariableInformation* to compute the values of the following attributes: *binaryLength* for all variables, *totalSubVariableNumVec* and *totalSubChromosomeLengthVec* for structured variables. Then, the operation *defineSimulation* is called to create the simulation instances based on the part for simulation programs in the file *environment.txt*. After the instantiation of the simulations, the operation *scanVariableOrder* located in the simulation module is called to determine the positions of the top-level variables in *variableVec*. Then, the operation *defineOptimizationParameter* in the optimizer module is called to define optimization parameter values based on the text file *environment.txt*. At last, the optimization program is called through the operation *optimize*. The operation *optimize* has a loop for each generation of individuals that converts the binary string to real values and calls the simulations.

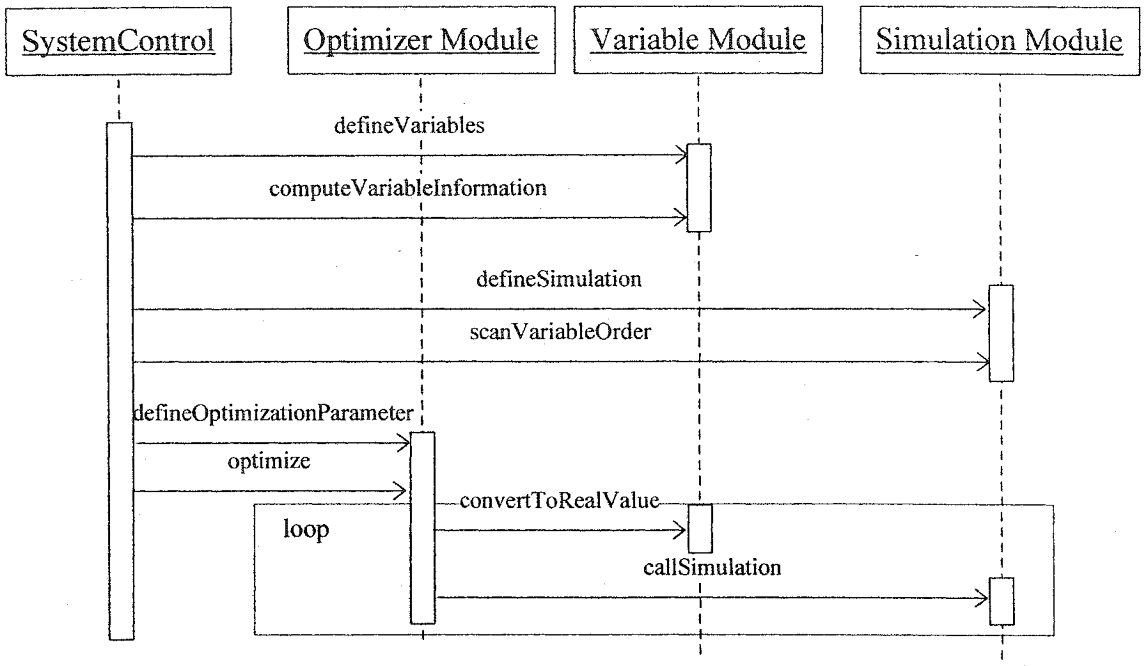


Figure 5.10 Sequence diagram for the integrated system

5.5 System GBOptimizer Customized to an Optimization Problem

In the previous section, the system GBOptimizer was used an example to demonstrate how the framework can be customized to an application. This section shows how the developed system can be customized to solve an optimization problem for green building design. Input files are used for this level of customization. Since the input files are mainly used by either the optimizer or the simulation programs, the customization of the system GBOptimizer is carried out from two aspects: customization for the optimizer and customization for the simulation programs.

5.5.1 Customization for the Optimizer

The user can customize the optimizer through the input text file *environment.txt* as presented in the previous section. The three parts of the file *environment.txt* define different aspects of an optimization problem. The part for variables defines the variables to be optimized in terms of their total number, types, and ranges. In this part, the variable for structural systems can have any number of alternatives; therefore, there is no limitation on the number of variables about wall types, roof types, floor types, overhang types, wall layers, roof layers, and floor layers. The part for simulation programs defines the objective functions and functional constraints for the optimization problem. The part for optimization parameters defines the GA parameters.

The data in the text file *environment.txt* are closely related; therefore, the consistency between these data is a prerequisite for the normal workings of the system. For example, if a single-objective function is indicated in the part for simulation programs, only optimization parameters applicable to single-objective GAs should be used. The consistency between the input data is checked by the system. If an inconsistency problem is noted, an indicative message is shown on the screen.

5.5.2 Customization for the Simulation Programs

As mentioned in the previous chapter, the input data files for the simulation programs include *fixedPart.idf*, weather files, and *simulationData.txt*. The file *fixedPart.idf* and weather files are used by the ASHRAE toolkit, and the *simulationData.txt* file is mainly used by the programs extended from the ASHRAE toolkit. These three input files can be changed in the following ways to customize the simulation programs:

- Change the number of weather files. Since each typical day has a weather file, the number of weather files changes with the number of typical days to be considered.
- Change the data in weather files. For example, the weather data such as outside

temperatures and wind speed can be changed for different building locations.

- Change the values of the data objects in the file *fixedPart.idf*. For example, the schedule of occupancy, lighting, and equipment can be changed in *fixedPart.idf*.

- Change the data values in the file *simulationData.txt*. For example, the utility rates and their structures can be changed in this file.

- Change the number and category of non-fuel natural resources, waste emissions, and fuel types considered in embodied energy. This change involves the file *simulationData.txt* and the data files working as databases to ensure the one-to-one relationship between their corresponding data fields. For example, the incorporation of a new non-fuel natural resource into this system is carried out in two steps: (1) add the consumption data for this new resource to the resource consumption fields in the data files for envelope layers, window types, and overhang types; and (2) add the chemical exergy of this new resource to the chemical exergy list in the file *simulationData.txt*.

5.6 System Implementation

Except for the ASHRAE toolkit, which is a commercial software package programmed with Fortran 90, the system was completely developed and implemented with the object-oriented programming language C++ of Microsoft Visual Studio 6.0. The Standard Template Library is used to implement the data structure of strings and vectors. There are total 35 classes and about 7000 lines of code, not including blank lines and comment lines.

In the GA implementation, the modified crossover operation as presented in Chapter 4 is applied to only the highest-level structured variables; for example, the building shape and the structural system configuration in this study. In other words, the procedures of the modified crossover shown in Figure 4.7 are followed only if the high-level gene belongs to the highest structured variables. This simplification has little impact on the achievement of the objective set for the modified crossover because the second sub-level structured variables such as wall types and roof types own much fewer genes in the chromosome than the highest-level structured variables.

5.7 Summary

Considering that simulation-based optimization problems vary in terms of the variables, the simulation programs, and the optimization model types, it is an advantage to have an object-oriented framework in order to avoid repetitive design and implementation for similar problems. The system framework presented in this research supports both single- and multi- objective genetic algorithms for green building design optimization. The main advantages of this framework can be summarized as:

- The framework reduces the burden on designers who want to pursue optimization with building simulation programs. Since the optimizer is implemented and is loosely coupled with the simulation program, designers can focus on the simulation side of an application, which is usually the area of their expertise.

- The framework provides means to ensure that the simulation is called only once for a transferred variable vector even if its results are used several times in multi-objective optimization or constrained optimization problems. This is essential since simulations are computation-intensive.

- The inclusion of the structured variable type greatly expands the application scope of this framework since the hierarchical relationship is common between building entities. The use of structured variables can deal with competitive design alternatives in parallel, which is useful to explore the large design space at the early stage of building design.

- The framework can be applied to other areas than green building design. The only modification required is to change from building simulations to other applicable simulation programs.

Chapter 6
Validation

Validation is important to make sure that the implemented system works properly as supposed. This chapter presents the validation of two aspects: validation of the simulation programs and validation of the optimizer.

6.1 Validation of the Simulation Programs

The ASHRAE toolkit has been tested with a broad range of conditions by the developers (Pedersen et al. 2000), so it can be regarded as a validated simulation program. With regards to the programs extended from the ASHRAE toolkit, they are verified by checking the results with hand calculations. For the optimization problem that is presented as the case study in the next chapter, two individuals are chosen from the final population. These two individuals are different in their structural system configuration to ensure that the activation and deactivation mechanism works properly for structured variables. Based on the implemented formulas and computation procedures, the objective function values of LCC (life-cycle cost) and LCEI (life-cycle environmental impact) are calculated by hand. Their results were found to be the same as the system outputs. This demonstrates that the extended simulation programs are implemented correctly.

6.2 Validation of the Optimizer

The optimizer is validated with mathematical optimization problems for which optimal solutions are known. A total of five test problems are used. Of the five problems, the first one (No. 1) is a constrained single-objective optimization problem taken from Deb (2000), and the other four (No. 2 to No. 5) are unconstrained multi-objective optimization problems taken from Deb (2001). The chosen problems, especially those for multi-objective optimization, have varied features such as the discontinuity of the Pareto front and the use of discrete variables. These features can test the ability of the optimizer to solve difficult problems.

All implemented techniques as shown in Figure 5.8 have been tested while validating the optimizer with the mathematical optimization problems. The tested techniques include the five selection methods: tournament, *RWS*, *SUS*, *SRS*, and *insertTournament*; the four fitness assignment methods: *MaxSOGA*, *DebConstrainedSOGA*, *RankMOGA*, and *FrontNSGA*; and the two elitist strategies: *ConstrainedSOGAElitism* and *ClusterMOGAElitism*. Although many groups of parameter settings are used with different methods for selection, fitness assignment, and elitism, only one group of parameter setting is presented for each test problem. Since the GA is a stochastic optimization method, each test problem is solved with the optimizer three times. The results from the three runs are similar; therefore, only one of them is presented in this section.

Test Problem No. 1

The unconstrained single-objective optimization problem has the following mathematical formulation:

$$\text{Minimize: } f_1(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \qquad (6\text{-}1)$$

Subject to:

$$\begin{aligned} &g_1(x) = 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geq 0 \\ &g_2(x) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0 \\ &0 \leq x_1 \leq 6 \\ &0 \leq x_2 \leq 6 \end{aligned} \qquad (6\text{-}2)$$

The optimization parameters are specified as follows: crossover probability=0.9, mutation probability=0.05, maximum number of generations=400, population size=50. *DebConstrainedSOGA*, *SUS*, and *ConstrainedSOGAElitism* are employed respectively for fitness assignment, selection, and elite preservation. No mating restriction is used in the crossover operation. With the above parameter settings, the result from the GA is x_1=2.246, x_2=2.375, with the objective function f_1 equal to 13.596. It is within 0.05% of the true optimal function value 13.591 at x_1=2.247 and x_2=2.382 (Deb 2000).

<u>Test Problem No. 2</u>

The second problem has two objective functions and thirty continuous variables as expressed in Formulas 6-3 and 6-4. The difficulty with this problem is that it has a non-convex Pareto front.

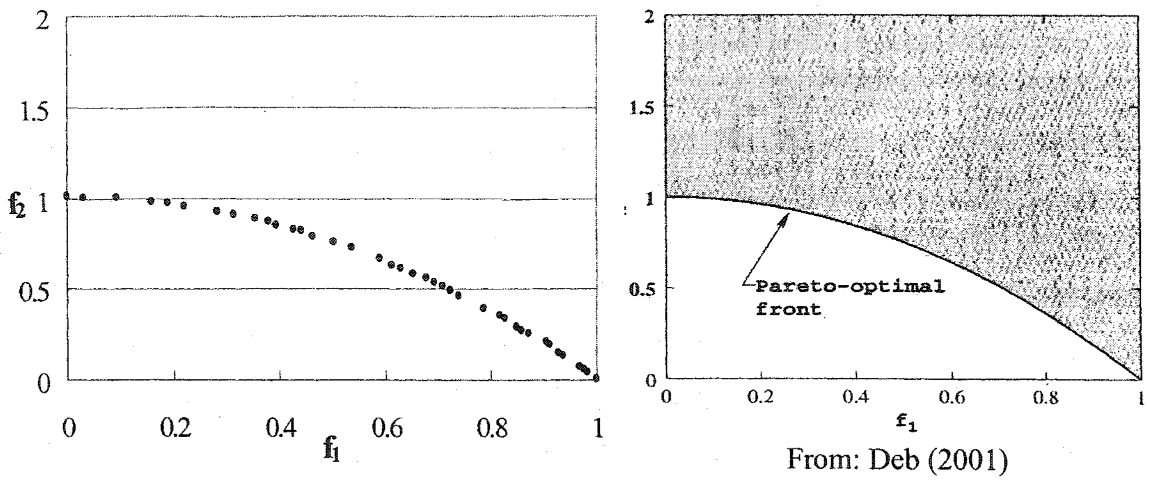
$$\text{Min: } \begin{aligned} &f_1(x) = x_1 \\ &f_2(x) = g(x) * h(x) \end{aligned} \qquad (6\text{-}3)$$

where,

$$g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i$$
$$h(x) = 1 - \left(\frac{x_1}{g(x)}\right)^2 \qquad (6\text{-}4)$$
$$0 \le x_i \le 1$$

Unless explicitly indicated, the following optimization parameters are used in this problem and the next three multi-objective optimization problems: crossover probability=0.9, maximum number of generations=500, population size=40. Mutation probability is approximately equal to the inverse of the whole chromosome length. *RankMOGA*, *insertTournament*, and *ClusterMOGAElitism* are employed respectively for fitness assignment, selection, and elite preservation. The capacity of the external population is 40, and at most 10 elites are immigrated from the external population in the *insertTournament* selection. No mating restriction is used in the crossover operation.

Figure 6.1 shows the comparison between the Pareto front obtained from the implemented optimizer and the true Pareto front. It demonstrates that the optimizer can find well-distributed Pareto optimal solutions along the Pareto front.



From: Deb (2001)

Figure 6.1 Pareto front generated by the optimizer (left) and the true Pareto front (right) for test problem No. 2

Test Problem No. 3

The third problem also has two objective functions and thirty continuous variables as shown in Formulas 6-5 and 6-6. The greatest difficulty with this problem lies in the disconnected Pareto fronts. Optimization parameters are the same as those for the second test problem. The comparison between the obtained results and the true Pareto front is illustrated in Figure 6.2. It demonstrates that the optimizer can locate the Pareto solutions in all of the five disconnected Pareto fronts.

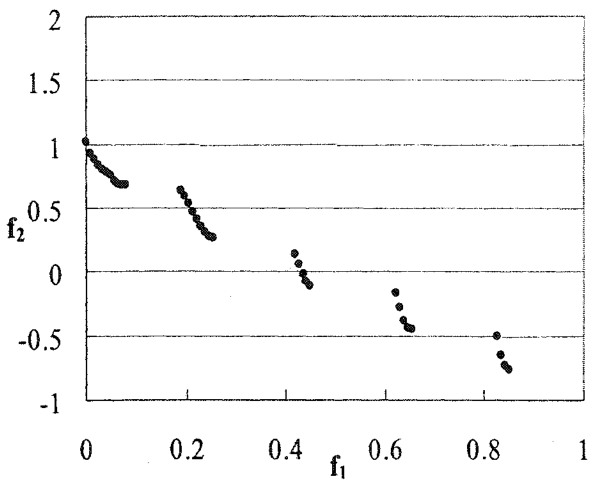
$$\text{Min: } \begin{matrix} f_1(x) = x_1 \\ f_2(x) = g(x) * h(x) \end{matrix} \qquad (6\text{-}5)$$

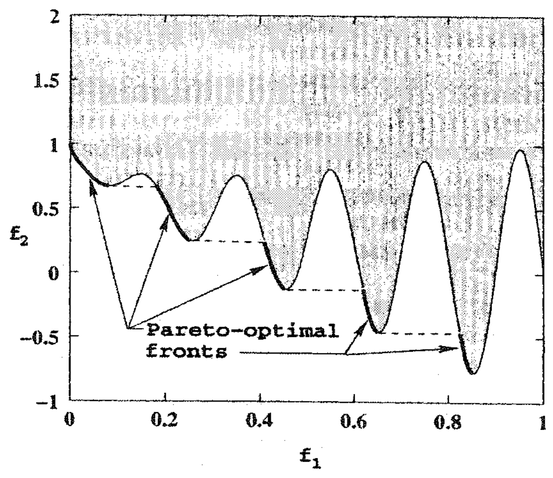
where,

$$g(x) = 1 + \frac{9}{29} \cdot \sum_{i=2}^{30} x_i$$

$$h(x) = 1 - \sqrt{x_1 / g(x)} - \left[x_1 / g(x) \right] \cdot \sin(10\pi x_1) \qquad (6\text{-}6)$$

$$0 \le x_i \le 1$$





From: Deb (2001)

Figure 6.2 Pareto front generated by the optimizer (left) and the true Pareto front (right) for test problem No. 3

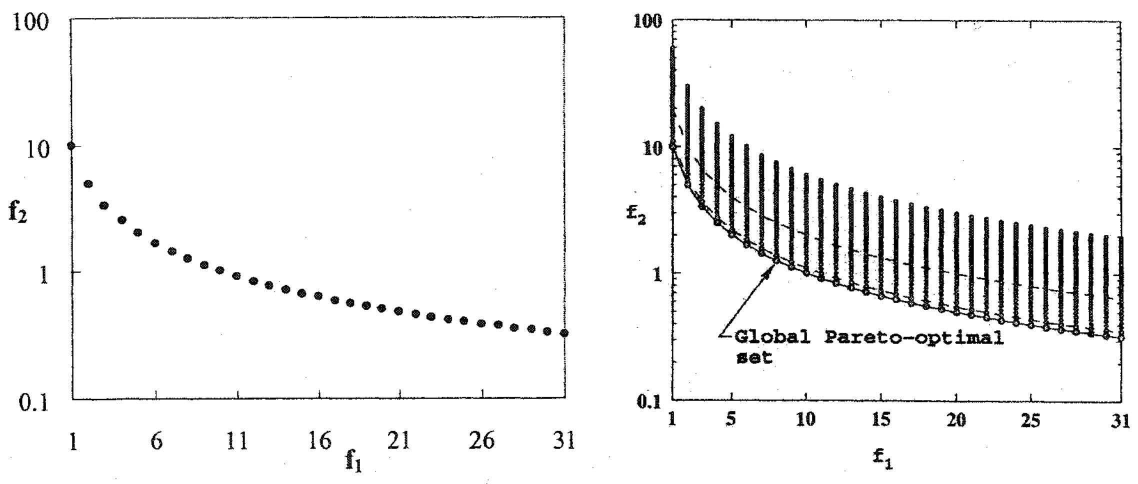
Test Problem No. 4

The fourth problem has two objective functions and eleven discrete variables as shown in Formulas 6-7 and 6-8. The Pareto front consists of 31 discrete points corresponding to the values of the first variable. This problem is solved with the same optimization parameter settings as those for the second test problem. The results are shown in Figure 6.3, which indicates that the optimizer has converged to all the discrete points in the Pareto front.

$$\text{Min:} \quad \begin{array}{l} f_1(x) = 1 + x_1 \\ f_2(x) = g(x) * h(x) \end{array} \qquad (6\text{-}7)$$

where,

$$
\begin{aligned}
& g(x) = \sum_{i=2}^{11} v(x_i) \\
& v(x_i) = \begin{cases} 2 + x_i & (x_i < 5) \\ 1 & (x_i = 5) \end{cases} \\
& h(x) = \frac{1}{1 + x_1} \qquad\qquad\qquad (6\text{-}8) \\
& x_1 \in \{0,1,2,\ldots,30\} \\
& x_i \in \{0,1,2,3,4,5\} \qquad (i = 2,3,\ldots,11)
\end{aligned}
$$



From: Deb (2001)

Figure 6.3 Pareto front generated by the optimizer (left) and the true Pareto front (right) for test problem No. 4

Test Problem No. 5

The fifth problem has two objective functions and ten continuous variables as shown in Formulas 6-9 and 6-10. It has a non-uniform Pareto front, which means that the density of Pareto solutions varies along the front. All parameters are kept the same as those for the second test problem. The obtained Pareto front and its comparison with the true one is illustrated in Figure 6.4, which indicates that the GA optimizer can capture the non-uniform Pareto solutions.

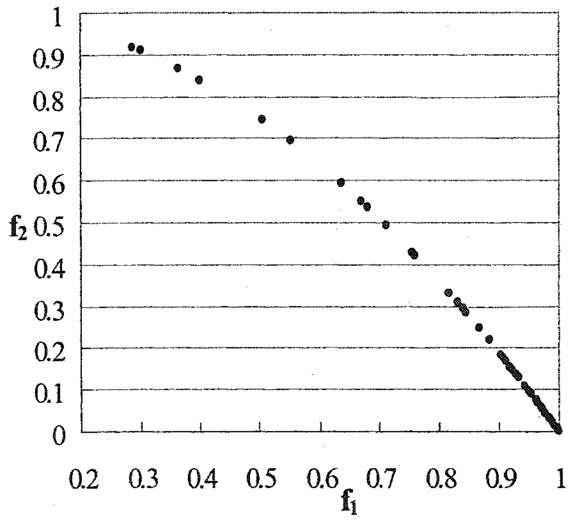
$$\text{Min:}\quad \begin{aligned} f_1(x) &= 1-\exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(x) &= g(x)*h(x) \end{aligned} \tag{6-9}$$

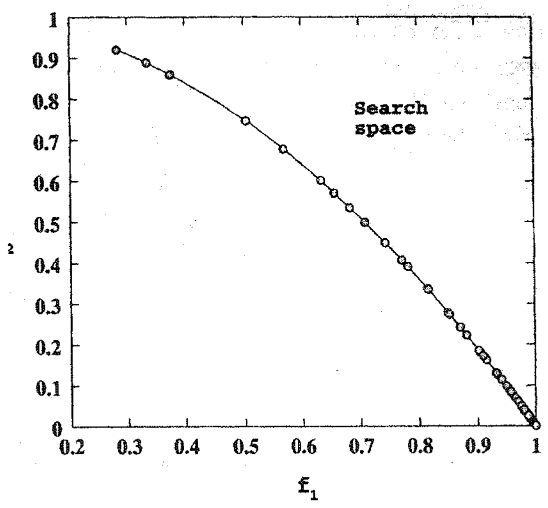
where,

$$g(x) = 1 + 9\left[\frac{\sum_{i=2}^{10} x_i}{9}\right]^{0.25}$$

$$h(x) = 1-\left(\frac{f_1}{g}\right)^2 \tag{6-10}$$

$$0 \le x_i \le 1$$





From: Deb (2001)

Figure 6.4 Pareto front generated by the optimizer (left) and the true Pareto front (right) for test problem No. 5

In conclusion, the validation showed that the implemented optimizer is able to find the optimal solutions for problems with varied difficulties.

Chapter 7
Case Study

This chapter applies the simulation-based optimization system GBOptimizer, developed and validated in the previous chapters, to optimize a green building. The case study is described first in terms of its design variables and parameter settings for the simulation program and the optimizer. Then, the results are presented and discussed. In the third section, the base case is altered slightly to investigate how the optimal results change with different performance criteria.

7.1 Case Description

The optimal design of a single-story office building located in Montreal is employed as a case study in this research. LCC (life-cycle cost) and LCEI (life-cycle environmental impact) are the two objective functions to be minimized. The study period of the life-cycle analysis is 40 years. The building has a total above-basement floor area of 500 m^2 and a floor-to-roof height of 3.6 m. This case study is carried out with the following simplifications: (1) Electricity is the only type of energy source used during the operation stage of a building, and the electricity rates do not change with seasons; (2) The extra labor cost with the construction of oblique wall is not considered; and (3) The building is simulated as a single thermal zone. In this section, the variable instantiations and the parameter settings for the building in this case study are presented.

7.1.1 Variable Instantiations

The variables used in this case study are listed in Table 7.1, which gives the name, the type, and the range of each variable. The definition of each variable can be referred back to Chapter 3 according to its name. In the column of variable type, the letters 'C', 'S', 'D', and 'T' represent continuous variable, structured variable, discrete variable, and constant, respectively. In the column of range, boundary values are indicated for a continuous variable, and a series of integers are indicated for a discrete variable. The integers denote the indices in the data file corresponding to that discrete variable; therefore, they may be not consecutive numbers. For example, the possible values for the discrete variable *winType* in Table 7.1 are a series of inconsecutive integers, each of which indicates the index of that window type in the data file *windowType.txt*.

Table 7.1 List of variables for the case study

Variable name			Variable type*	Range
orientation			C	[0, 90]
shape			S	1. Rectangular; 2. 'L' shape
Rectangular shape		r_0	C	[0.1, 1.0]
L-shape		r_0	C	[0.3, 1.0]
		r_1	C	[0.2, 0.7]
		r_2	C	[0.2, 0.7]
wallTilt$_i$ (i=1,2,3,4)			C	[75, 105]
winSwitch			S, T	off
winSwitch=off		winType$_i$ (i=1,2,3,4)	D	2,3,4,5,6,7,9
winRatio$_i$ (i=1,2,3,4)			C	[0.2, 0.8]
strSystem			S	1. Concrete frame; 2. Steel frame
Concrete Frame	wallType		S	1. Concrete block wall 2. Steel-stud wall
	Concrete block wall	Cladding	D	1,3,4,5,6,7
		Insulation	D	3,4,5,6,10,11,12,13
		Vapor barrier	T	Modified bitumen membrane
		Structure	T	200 mm concrete block
		Finish	T	12.7 mm gypsum board
	Steel-stud wall	Cladding	D	1,3,4,5,6,7
		Insulation	D	1,2,3,8,9,10
		Air barrier	T	Sheathing paper
		Sheathing	T	12.7 mm gypsum sheathing
		studInsulation	D	1,2,5,6
		Vapor barrier	T	Polyethylene
		Finish	T	12.7 mm gypsum board
	Roof type		S	1. Conventional CIP roof 2. Green CIP roof
	Conventional CIP roof	Roofing	T	2-ply modified bitumen
		Coverboard	T	12.7 mm mineral fiberboard
		Insulation	D	17,18,19,20,34,35,36,37
		Vapor barrier	T	polyethylene
		Roof deck	T	175 mm CIP concrete
		Other	T	400 mm ceiling air space
		Finish	T	16 mm acoustical tile

	Green CIP roof	Grass	T	Green roof layers
		Roofing	T	2-ply modified bitumen
		Coverboard	T	12.7 mm mineral fiberboard
		Insulation	D	16,17,18,19,33,34,35,36
		Vapor barrier	T	polyethylene
		Roof deck	T	175 mm CIP concrete
		Other	T	400 mm ceiling air space
		Finish	T	16 mm acoustical tile
	Floor type		S, T	CIP concrete floor
	CIP floor	Structure	T	200 mm CIP concrete
	overhangType$_i$ (i=1,2,3,4)		D	1. No overhang 2. Aluminum type
Steel Frame	Wall types and wall layers			The same as concrete frame
	Roof type		S	1. Conventional OWSJ roof 2. Green OWSJ roof
	Conventional OWSJ roof	Roofing	T	2-ply modified bitumen
		Coverboard	T	12.7 mm mineral fiberboard
		Insulation	D	17,18,19,20,34,35,36,37
		Vapor barrier	T	polyethylene
		Roof deck	T	Steel deck above OWSJ
		Other	T	400 mm ceiling air space
		Finish	T	16 mm acoustical tile
	Green OWSJ roof	Grass	T	Green roof layers
		Roofing	T	2-ply modified bitumen
		Coverboard	T	12.7 mm mineral fiberboard
		Insulation	D	16,17,18,19,33,34,35,36
		Vapor barrier	T	polyethylene
		Roof deck	T	Steel deck above OWSJ
		Other	T	400 mm ceiling air space
		Finish	T	16 mm acoustical tile
	Floor type		S, T	OWSJ floor
	OWSJ floor	Structure	T	64 mm concrete topping on steel deck
	overhangType$_i$ (i=1,2,3,4)		D	The same as concrete frame
overhangDepth$_i$ (i=1,2,3,4)			C	[0.1, 1.2]
overhangHeight$_i$ (i=1,2,3,4)			T	0.2 m

* C: continuous; D: discrete; S: structured; T: constant

For the building in this case study, the orientation may vary between 0 and 90 degrees. The shape of the building plan has two possible values: rectangular shape and L-shape. The aspect ratio r_0 is in the range between 0.1 and 1.0 for the rectangular shape while it lies in between 0.3 and 1.0 for the L-shape. Both the length ratio (r_1) and the width ratio (r_2) for the L-shape building plan are in the range between 0.2 and 0.7. These shape-related variable ranges are thus set to ensure that each side of a building is not too small to be practicable. The wall tilt for each side may vary between 75 and 105 degrees.

The window design switch is set off, which means that the facades with different orientations may have different window types. Seven available window types for this building design are listed below. All these window types have the glass thickness of 6 mm. The air space is 13 mm and 6.5 mm for double glazing and triple glazing, respectively. The leading number indicates the index of each window type in the data file *windowType.txt*.

2). Double clear glazing, transmission coefficient (U)=2.7 W/($m^2 \cdot K$), diffuse solar heat gain coefficient (SHGC) =0.60;

3). Reflective double glazing, U=2.7 W/($m^2 \cdot K$), SHGC=0.25;

4). Low-e double glazing, emissivity=0.2, on the exterior of the inside pane, U=2.0 W/($m^2 \cdot K$), SHGC=0.56;

5). Low-e double glazing, emissivity=0.1, on the exterior of the inside pane, U=1.8 W/($m^2 \cdot K$), SHGC=0.49;

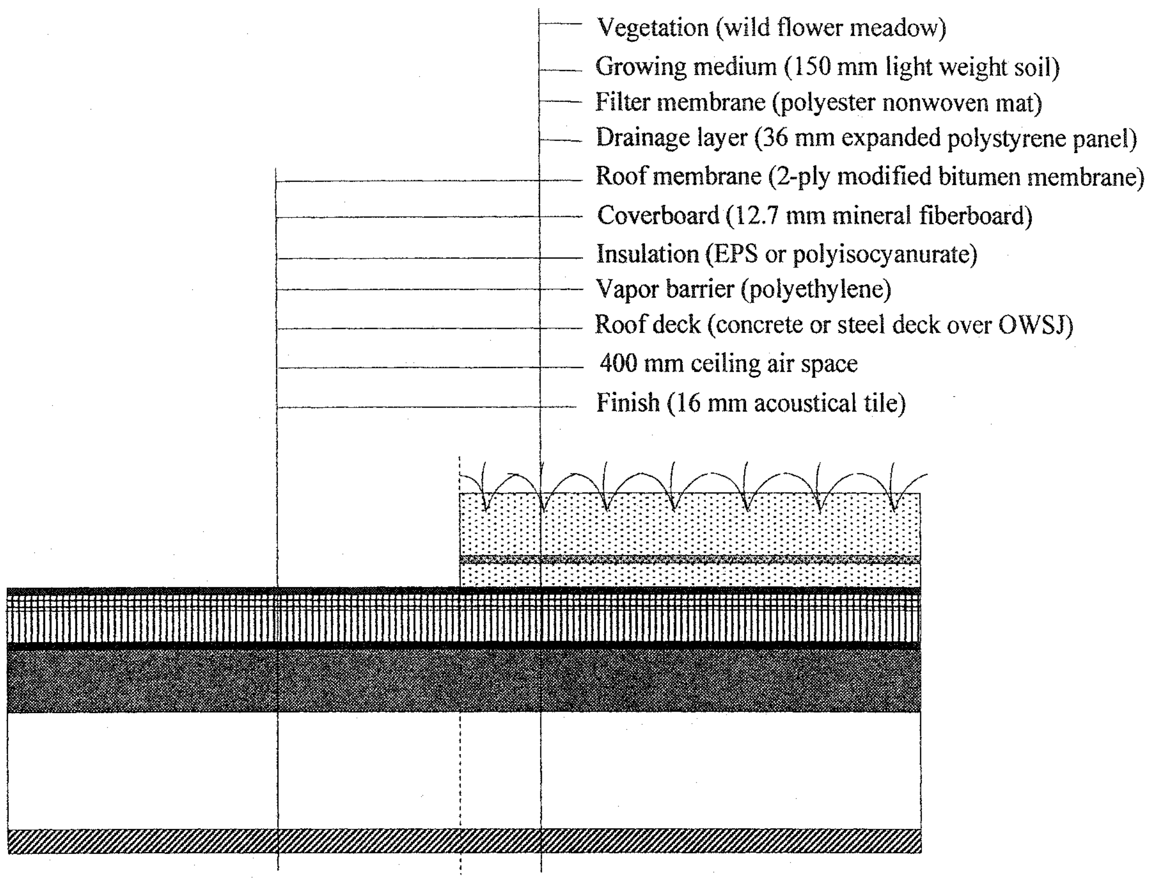
6). Low-e double glazing, emissivity=0.2, on the interior of the outside pane, U=2.0 W/(m^2·K), SHGC=0.53;

7). Low-e double glazing, emissivity=0.1, on the interior of the outside pane, U=1.8 W/(m^2·K), SHGC=0.44;

9). Low-e triple glazing, emissivity=0.2, on the exterior of the inside pane, U=1.9 W/(m^2·K), SHGC=0.47.

In this case study, steel frame and concrete frame are the two alternatives for the structural system. Both steel frame and concrete frame have the same two possible wall types: concrete block wall and steel-stud wall. However, the two structural systems have different roof types. For the steel frame structure, steel deck above OWSJ (open web steel joist) system is used for the roof with two possible alternatives: conventional OWSJ roof and green roof with OWSJ system. For the concrete frame structure, CIP (cast-in-place) concrete flat plate system is used for the roof with two possible alternatives: conventional CIP roof and green roof with CIP concrete system. The two structural systems also have different floor types. OWSJ floor with concrete topping and CIP concrete floor are used in the steel frame and the concrete frame, respectively. The compositions of each wall type, roof type, and floor type are described next.

Different wall types, roof types, and floor types vary with their compositions in terms of the material and the sequence of layers. The concrete block wall consists of cladding, rigid insulation, vapor barrier, concrete block, and finish, while the steel-stud wall consists of cladding, rigid insulation, air barrier, sheathing, steel-stud with cavity insulation (i.e., *studInsulation* in Table 7.1), vapor barrier, and finish. The conventional OWSJ roof is composed of the following sequence of layers from outside to inside: roofing membrane, coverboard, rigid insulation, vapor barrier, steel roof deck, and finish. The composition of the conventional CIP roof is the same as that for the conventional OWSJ roof except that the roof deck changes from steel to concrete. Compared with the conventional roof system, the green roof system has several additional layers including vegetation, growing medium, filter membrane, and drainage layer, listed from the top down (Peck and Kuhn 2001). These additional layers are generally called the "green roof layers" in Table 7.1. The comparison between the green roof system and the conventional roof system is illustrated in Figure 7.1, where the material for each layer is indicated in the parenthesis. With respect to the composition of floors, the CIP concrete floor has a thickness of 200 mm, and the OWSJ floor has a thickness of 64 mm concrete topping above the steel deck.


Vegetation (wild flower meadow)
Growing medium (150 mm light weight soil)
Filter membrane (polyester nonwoven mat)
Drainage layer (36 mm expanded polystyrene panel)
Roof membrane (2-ply modified bitumen membrane)
Coverboard (12.7 mm mineral fiberboard)
Insulation (EPS or polyisocyanurate)
Vapor barrier (polyethylene)
Roof deck (concrete or steel deck over OWSJ)
400 mm ceiling air space
Finish (16 mm acoustical tile)

Conventional Roof Green Roof

Figure 7.1 Comparison of configuration for conventional roof and green roof

Although there is no limitation on the number of wall layers and roof layers to be optimized as variables, some of them are dealt as constants because of their minor impacts on the two considered performance criteria. The layers that are defined as variables include cladding, wall insulation, and roof insulation. Table 7.2 lists the discrete values for the insulation layers in wall and roof, together with their indices in the corresponding data files. The Quebec energy code regulation (Editeur Officiel du Quebec 1992) has been used in defining these alternative insulation levels so that all solutions satisfy mandatory requirements such as the minimum thermal resistance of walls and roofs. The cladding for both wall types can take one of six possible values, which are listed below. Again, the leading numbers indicate the indices in the data file *cladding.txt*.

1). 100 mm clay brick

3). 100 mm split faced concrete block

4). wood bevel siding

5). vinyl siding

6). commercial steel siding

7). stucco over metal mesh

Table 7.2 Insulation materials in walls and roof

wallType1	**wallType2**		**roof**
Insulation*	studInsulation	Insulation	Insulation*
3. 76 mm EPS	1. 102 mm fiberglass	1. 25 mm EPS	17. 76 mm EPS
4. 102 mm EPS	2. 152 mm fiberglass	2. 51 mm EPS	18. 102 mm EPS
5. 127 mm EPS	5. 102 mm rockwool	3. 76 mm EPS	19. 127 mm EPS
6. 152 mm EPS	6. 152 mm rockwool	8. 25 mm XPS	20. 152 mm EPS
10. 76 mm XPS		9. 51 mm XPS	34. 51 mm PI
11. 102 mm XPS		10. 76 mm XPS	35. 76 mm PI
12. 127 mm XPS			36. 102 mm PI
13. 152 mm XPS			37. 127 mm PI

* EPS: Expanded Polystyrene; XPS: Extruded Polystyrene; PI: Polyisocyanurate

The variable *overhangType* takes one of two possible values: no overhang and aluminum overhang. Aluminum overhang is used because of its advantages in light weight and corrosion resistance. The overhang depth is defined as a continuous variable ranged between 0.1 m and 1.2 m, while the overhang height is fixed to be 0.2 m.

7.1.2 Parameter Settings

In this case study, the heating season is considered from November to March (5 months), and the cooling season is from June to August (3 months). Each month needs a typical day for energy estimation and a design day for peal load calculation. Thus, a total number of 16 days are defined. The hourly outdoor dry-bulb and wet-bulb temperatures for each day are extracted from a weather file for Montreal: for a given month, the hourly temperatures used for energy estimation are taken from the day with minimum deviation from the monthly average temperatures; the hourly temperatures used for peak load calculation are taken from the day with minimum deviation from the monthly peak temperatures. The indoor design temperatures are 21°C and 23°C in the heating and cooling season respectively, without night setback or setup. Rooftop units with a coefficient of performance of 3.0 are used for cooling; electric baseboard heaters are used for heating, assuming an efficiency of 100%. Internal loads comply with the ASHRAE handbook (2001) and Model National Energy Code of Canada for Buildings (National Research Council 1997). Specifically, the lighting power density is 18 W/m^2; the equipment load factor is 10.8 W/m^2; and the occupancy density is about 25 m^2 per person. Daily operating schedules take the default values for office buildings according to the national energy code (National Research Council 1997).

Green roof systems have a wide range of economical and environmental benefits such as energy savings, the extended life of roof membrane, sound insulation, the reduction of the heat island effect, and the improved storm water retention (Peck and Kuhn 2001). Only energy savings are considered here because it is closely relevant to this study. The thermal behavior of a green roof does not depend only on the thermal properties of its layers, but also on the plants' biological functions such as photosynthesis, respiration, transpiration and evaporation (Eumorfopoulou and Aravantinos 1998). This makes it difficult to model the impact of a green roof on energy consumption. Although simulation models for green roofs have been proposed in the literature (Niachou et al. 2001; Onmura et al. 2001), none of them is incorporated in the ASHRAE toolkit simulation program used in this research. Therefore, a simplified approach is employed to circumvent the difficulty of energy simulation with a green roof. Since the impact of vegetation can be reflected through the temperatures at layers underneath the green roof layers, the thermal impact of the green roof layers is approximately modeled in the ASHRAE toolkit by using the measured temperatures at the roof membrane layer as the special boundary conditions for the green roof system.

The measured temperatures at the roof membrane layer were received from the experimental study by the National Research Council of Canada (Bass and Baskaran 2003; Liu and Baskaran 2003). In that experimental study, a facility with 36 m^2 of green roof area was constructed, and sensors embedded between different roof layers were used to collect field-measured temperatures. Although the experimental study was carried out in Ottawa, the measured data can be employed in this research because (1) Ottawa and Montreal have similar climate conditions; and (2) The same green roof layers are used in that experimental study and this research.

The hourly temperatures at the roof membrane layer fluctuate in a much smaller range than outdoor air temperatures. For example, the roof membrane temperature remained almost steady between 3 and 4 °C throughout a winter day in January with outdoor temperature ranged between -20 and -10 °C (Bass and Baskaran 2003). The small temperature fluctuation of the roof membrane layer makes it feasible to group the sixteen days according to their maximum and minimum daily outdoor temperatures. The purpose of this grouping is to reduce the amount of required experimental data. The grouping results are listed in Table 7.3, where the words "average" and "peak" in parenthesis represent the day respectively for energy estimation and peak load calculation. All the days in a group have similar maximum and minimum daily outdoor temperatures as the corresponding field monitoring day. Thus, for each group, the hourly roof membrane temperatures are taken from a field monitoring day in the experimental study by the National Research Council of Canada.

Table 7.3 Groups of weather conditions for the simulation of the green roof system

Group No.	Group Members	Field Monitoring Day
1	January (average) March (average) November (average) November (peak) December (average)	December 3, 2000
2	January (peak) February (average) February (peak) March (peak) December (peak)	February 5, 2002
3	June (average) August (average)	May 20, 2001
4	June (peak) July (average)	July 16, 2001
5	July (peak)	July 22, 2002
6	August (peak)	August 1, 2002

The measured roof membrane temperatures are obtained in two different ways. For the 2nd, the 5th, and the 6th groups in Table 7.3, the temperatures were provided by Dr. Liu (2004) from the National Research Council of Canada. Since the original data were recorded temperatures every 15 minutes for three different locations at the roof membrane, the average temperature of the three locations is calculated every hour and then used in this case study. For the 1st, the 3rd, and the 4th groups in Table 7.3, the roof membrane temperatures are derived from the published technical report (Bass and Baskaran 2003) and the data provided by Dr. Liu. The technical report provides the daily temperature ranges of the roof membrane for several days with similar outdoor air temperature ranges as the 1st, the 3rd, and the 4th groups in Table 7.3. Because hourly roof membrane temperatures are not given in that technical report, they are calculated from the daily ranges and the known hourly temperature profile from Dr. Liu with the following formula:

$$T_t^j = T_l^j + \left(\frac{T_t^i - T_l^i}{T_h^i - T_l^i}\right) \cdot (T_h^j - T_l^j) \tag{7-1}$$

where, T represents the roof membrane temperature in Kelvin; the subscripts h and l represent the daily maximum and minimum temperatures; the superscripts i and j respectively stand for the days with and without known hourly temperature profile for the roof membrane. In the above formula, the two days represented by the superscripts i and j should have similar ranges for the daily outdoor air temperature. Therefore, the superscript i stands for the field monitoring days of the 2nd, the 5th, and the 6th groups (see Table 7.3), which respectively correspond to what the superscript j represents: the field monitoring days of the 1st, the 3rd, and the 4th groups.

The initial construction cost of the green roof layers is estimated as 104 $\$/m^2$ (Peck and Kuhn 2001). Of the green roof layers, vegetation and growing medium have almost no negative environmental impacts; filter membrane has negligible quantity. Thus, the embodied environmental impact of the green roof layers depends entirely on the drainage layer, which is a 36 mm expanded polystyrene panel in this case study.

The discount rate and the expected energy cost escalation rate (both including general inflation) are 9% and 3%, respectively (National Research Council 1997). Local electricity rate structure (Hydro-Quebec 2004) is used: $12.33 per kW of billing demand, $0.0383 per kWh for the first 210000 kWh, $0.0249 per kWh for the remaining electricity consumption.

The ratios (the value of α in formula 3-5 and 3-6) between exergy and energy content for different fuel types are taken from Szargut et al. (1988) and Wall (1977). Table 7.4 lists the values of α for the fuel types considered in embodied energy and operating energy. The chemical exergy values of non-fuel materials together with their references are listed in Table 7.5. The values of unit abatement exergy for CO_2, SO_x, and NO_x are taken as 5.86, 57, and 16 MJ/kg, respectively (Dewulf et al. 2001). Since the values of unit abatement exergy for CH_4 and N_2O have not been found in the literature, they are derived by assuming that the abatement exergy is proportional to the global warming potential (GWP) index. Hence, they are calculated by multiplying the unit abatement exergy for CO_2 by their GWP indices over a 100-year period, which are 23 and 296 respectively for CH_4 and N_2O (Watson and the Core Writing Team 2001).

Table 7.4 Values of the ratio between exergy and energy (α) for different fuel types

Fuel Type	Hydro	LPG	Diesel	Gasoline	Natural gas	Wood	Coal	Heavy fuel	Nuclear	Feedstock	Electricity
α	1.0	1.0	1.07	1.07	1.04	1.0	1.09	1.04	1.0	1.0	1.0

Table 7.5 Chemical exergy of non-fuel materials

Non-fuel Material	Limestone	Clay	Iron ore	Sand	Gypsum	Coarse aggregate	Fine aggregate	Scrape steel	Wood fiber	Pheno formaldehyde resin
Chemical exergy (kJ/kg)	34	766	420	32	50	32	32	7000	17640	25585
Reference	(1)	(2)	(1)	(2)	(2)	(2)	(2)	(2)	(2)	(2)*

Notes:

Reference (1): Finnvedenand Ostlund (1997)

Reference (2): Szargut et al. (1988)

*: the average of Pheno and Formaldehyde

Since electricity is the only energy source for the operation of this building, the overall exergetic efficiency (η in formula 3-6) is calculated based on the electricity mix and the efficiency of delivered electricity from different primary energy sources. The electricity mix in Quebec is 96% from hydro, 2% from oil, and 2% from nuclear (Government of Canada 2001). The exergetic efficiency of the delivered electricity from hydro, oil, and nuclear is 0.86, 0.30, and 0.26, respectively (Zhang 1995). The generation and transportation losses are considered in the efficiency of delivered electricity from oil and nuclear, but generation loss is not considered for hydro-electricity because hydro is a renewable energy source. Based on the above data, the overall exergetic efficiency of the delivered electricity in this case study can be calculated as:

$$\eta = \frac{1}{0.96/0.86 + 0.02/0.30 + 0.02/0.26} = 0.79$$

The emission factors of delivered electricity are calculated from the electricity mix and the emission coefficients due to electricity generation from different fuel types. The emission coefficients for the electricity generated from hydropower stations, thermal power plants and nuclear plants are listed in Table 7.6. Thus, the CO_2 equivalent emission factor of delivered electricity is calculated as:

$$0.96*4.167*10^{-3} + 0.02*2.161*10^{-1} + 0.02*4.167*10^{-3} = 8.41*10^{-3} \quad \text{(kg/MJ)}$$

Similarly, the SO_x and NO_x emission factors can be calculated as $4.64*10^{-5}$ kg/MJ and $1.89*10^{-5}$ kg/MJ, respectively.

Table 7.6 Emission coefficients of the electricity generated from different sources

Data Items	**Primary Energy Source**		
	Hydro	**Oil**	**Nuclear**
CO_2 equivalent (kg/MJ)	$4.167*10^{-3}$	$2.161*10^{-1}$	$4.167*10^{-3}$
SO_2 (kg/MJ)	$1.944*10^{-6}$	$2.226*10^{-3}$	$8.333*10^{-7}$
NO_x (kg/MJ)	$1.167*10^{-5}$	$3.850*10^{-4}$	$5.556*10^{-7}$

Data source: Gagnon et al. (2002).

For the optimization parameters, the following values are used: crossover probability=0.9, mutation probability=0.006, maximum number of generations=300, population size=40, the mating restriction is applied after 100 generations, *insertTournament* is the selection technique and the maximum number of elites introduced from the external population to the original population in each generation is 10, *RankMOGA* is the fitness strategy, and *ClusterMOGAElitisim* is employed as the elitist strategy with an external population capacity of 30.

Overall, the system optimization environment is customized for this case study through the text file *environment.txt*, part of which is illustrated in Figure 7.2.

```
#Variables Begin#

Continuous, orientation, Local, 0, 90, 1.0,;
Discrete, windowTypeSide1, Remote, C:\system\windowType.txt, 2,3,4,5,6,7,9,;
Discrete, windowTypeSide2, Remote, C:\system\windowType.txt, 2,3,4,5,6,7,9,;

Structured, structuralSystem, Local, 1,2,;
! Concrete frame system

 Structured, wallType, Local, 1, 2,;

 ! Masonary Cavity Wall type
    Discrete, cladding, Remote, C:\system\cladding.txt, 1, 3, 4, 5, 6, 7,;
    Discrete, insulation, Remote, C:\system\insulation.txt, 3,4,5,6,10,11,12,13,;
    Discrete, membrane, Remote, C:\system\membrane.txt, 7,;
    Discrete, structure, Remote, C:\system\structure.txt, 2,;
    Discrete, finish, Remote, C:\system\finish.txt, 1,;&

 ! Steel Frame Wall type
    Discrete, cladding, Remote, C:\system\cladding.txt, 1, 3, 4, 5, 6, 7,;
 ! Other variables are not shown

#Variables End#

#Simulation Program Begin#

Primary Simulation, AnnualEnergy,  -1, C:\system\ simulationData.txt;
Criterion, LCC,  -1, C:\ system \simulationData.txt;
Criterion, LCEI,  -1, C:\ system\ simulationData.txt;

#Simulation Program End#

#Optimization Parameters Begin#

Generation, 300,;
Population Size, 40,;
Crossover Probability, 0.9,;
MateRestrictionStartGeneration, 100,; ! generation that starts mating restriction
Mutation Probability, 0.01,;
Selection, insertTournament, 10,;
Fitness Assignment, RankMOGA,;
Elitist Strategy, ClusterMOGAElitism, 30,;

#Optimization Parameters End#
```

Figure 7.2 Definition of the simulation-based optimization environment with the *environment.txt* file

7.2 Results and Discussion

The design space for the case described in the previous section consists of about $5.3*10^{32}$ possible solutions. The optimization is run three times. Each run takes approximately 70 hours on a computer with Windows XP (3.06 GHz Pentium-IV processor, 512 MB RAM). All non-dominated solutions from the external populations in the three runs are combined together to form a global Pareto front as shown in Figure 7.3. This Pareto front is composed of two isolated regions which are designated as zones *A* and *B*. Solutions in Pareto zone *A* have lower costs but larger environmental impacts, and solutions in zone *B* have lower environmental impacts but larger costs.

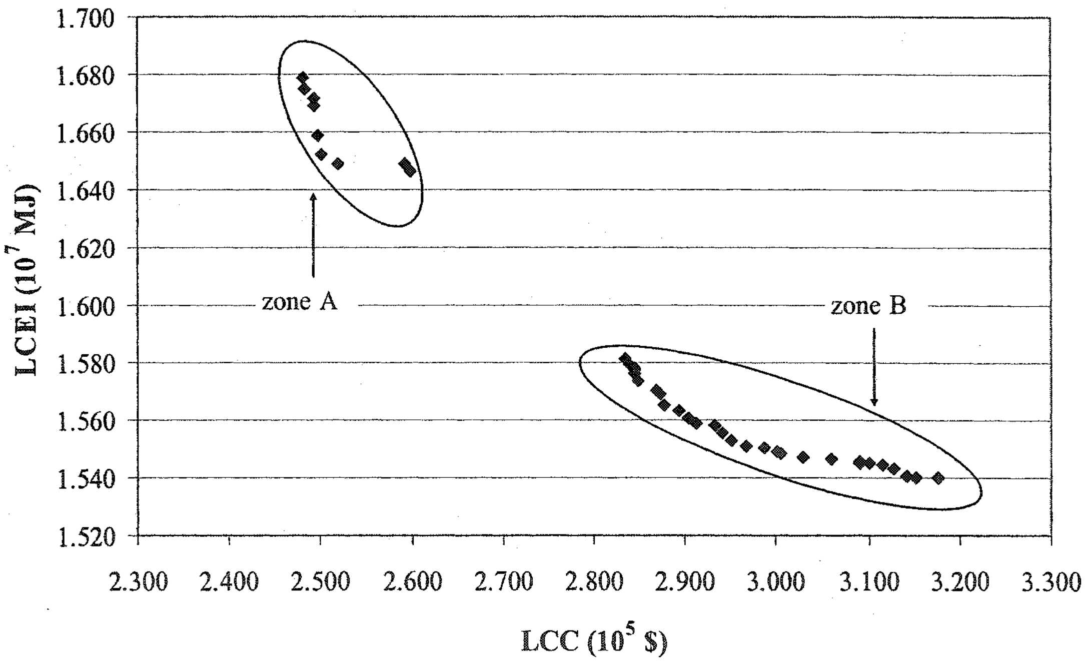


Figure 7.3 The global Pareto front compiled from three runs

Since the general trend of convergence is similar for the three runs, two quantified performance metrics for multi-objective GAs are used to compare the Pareto sets from the three runs in order to decide which run is selected for further analysis. The first performance metric is error ratio, which evaluates the closeness between the Pareto set from one run and the global Pareto front compiled from the three runs. The second metric is spread, which evaluates the diversity of non-dominated solutions. Let Q and P^* denote a Pareto set and the global Pareto set, respectively, the error ratio and spread can be calculated with the following formula (Deb 2001):

$$\text{errorratio} = \frac{\sum_{i=1}^{N} n_i}{N} \tag{7-2}$$

$$\text{spread} = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{N-1} |d_{ij} - \bar{d}|}{\sum_{m=1}^{M} d_m^e + (N-1)\cdot \bar{d}} \tag{7-3}$$

where,

N: the total number of solutions in the Pareto set Q;

n_i: a Boolean value that is equal to 0 if the i-th solution in Q belongs to the global Pareto set P^*; otherwise, it is 1;

d_{ij}: the distance between the *i-th* solution and its neighbouring solution j in the Pareto set Q;

$\bar{d}$: the average of d_{ij};

d_m^e : the distance between the extreme solutions of P^* and Q corresponding to the *m-th* objective function;

M: the total number of objective functions (M=2 for this case study);

m: the m-th objective function.

Before the calculation of d_{ij}, the Pareto solutions in *Q* need to be sorted according to the objective function values. Then, the distance d_{ij} is computed sequentially along the sorted Pareto solutions with the following formula:

$$d_{ij} = \sum_{m=1}^{M} \left| \frac{f_m^i - f_m^j}{f_m^{max} - f_m^{min}} \right| \tag{7-4}$$

where, *f* stands for the objective function value, and f_m^{max} and f_m^{min} are the maximum and minimum value of the *m-th* objective function in the Pareto set *Q*.

With the global Pareto set P^* comprising of the non-dominated solutions in the combined external populations from the three runs, the above-mentioned two performance metrics for each run are calculated and presented in Table 7.7. Since a smaller value indicates better performance for both metrics, the first run is the best in terms of the closeness to the global Pareto front, while the third run is the best in terms of diversity of the obtained Pareto solutions. Because the first run has much smaller error ratio than the third run, it is selected for further analysis to illustrate the Pareto solutions with respect to both performance space and variable space.

Table 7.7 Performance metrics for the three runs

Runs	**Error ratio**	**Spread**
Run1	0.23	0.78
Run2	0.73	0.92
Run3	0.70	0.73

For the first run, individuals in the initial, final, and external population are distributed in the performance space as illustrated in Figure 7.4. It can be seen from this figure that:

- The initial randomly produced individuals are widely distributed, while the final population is clustered to the lower left corner. The final population is close to the external population. Both of the two observations indicate that a good convergence has been achieved.

- The role of optimization is noticeable. Every solution in the initial population is dominated by some solutions in the external population at the end of the 300 generations. The averages of the life-cycle cost and the life-cycle environmental impact are $3.959*10^5$ \$ and $1.989*10^7$ MJ in the initial population, while they reached $2.835*10^5$ \$ and $1.586*10^7$ MJ in the external population.

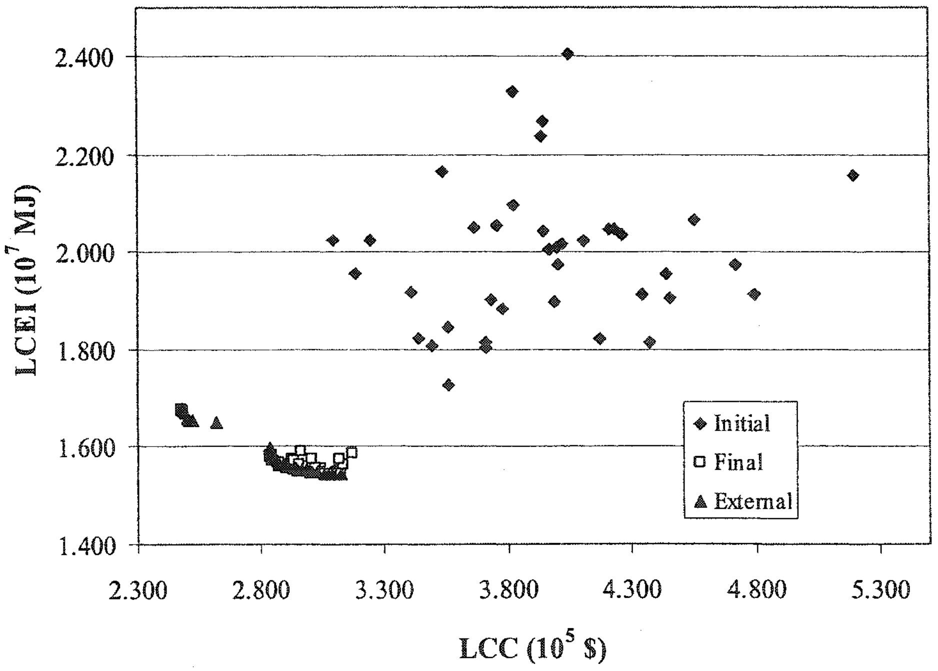


Figure 7.4 Distribution of initial, final, and external population for the first run

The evolution process is illustrated in Figure 7.5, where the external populations at the end of 100, 200, and 300 generations are compared. This figure shows that the genetic algorithm can quickly converge to the near optimal Pareto front. However, much time is required in the later generations to locate better solutions in the local area and to find a good distribution among the Pareto solutions.

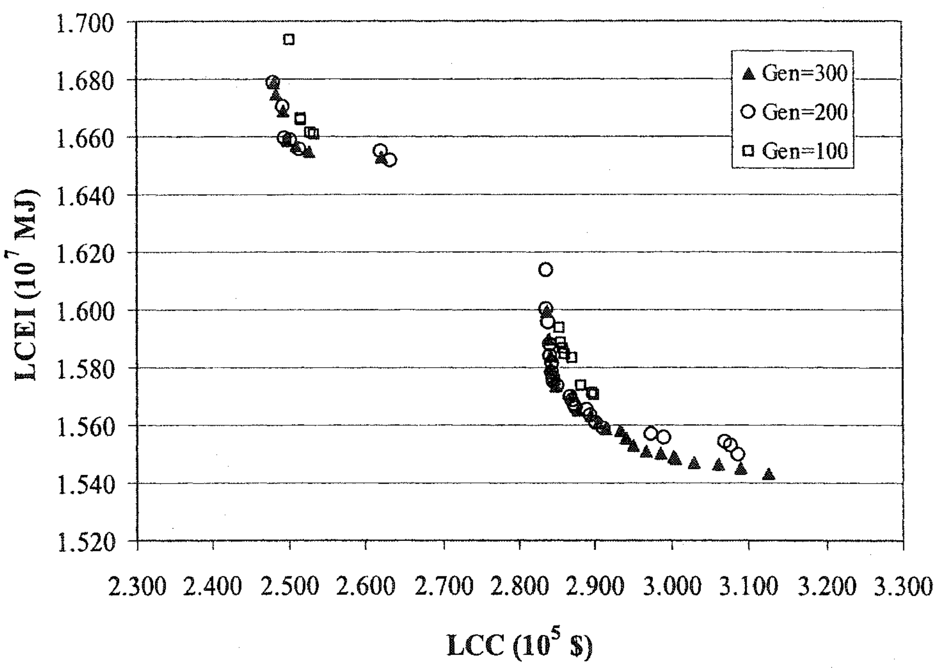


Figure 7.5 Comparison of the external population in the evolution process

Non-dominated solutions in the first run are listed in Table 7.8 with the corresponding values for variables and objective functions. The solutions are arranged in increasing order of the life-cycle cost, and those located in Pareto zone *A* are shaded. Some abbreviations are used because of space limitations. The numbers in the column heads indicate the side indices as shown in Figure 3.2. For example, 1, 3, 4 in the column titled "window ratio" means that the listed values are the window ratios for side1, side3, and side4. The two characters in the column of overhang type represents whether the aluminium overhang is used (Y) or not (N). For other discrete variables, only indices are presented in Table 7.8. The actual design alternatives corresponding to these indices are defined in section 7.1. Results are explained below.

Table 7.8 Non-dominated solutions in the external population from the first run

ID.	confi g.	orien. (deg.)	shape	r_0	tilt 1-4 (deg.)	window type				window ratio		wall type	wall			roof type	roof	overhang type		overhang depth 2 (m)	LCC (10^5 $)	LCEI (10^7 MJ)
						1	2	3	4	1,3,4	2		Clad.	Insu.	stud Insu.		Insu.	1,3,4	2			
1	2	0	1	0.93	75	7	4	7	5	0.2	0.2	2	5	3	1	1	20	N	N		2.480	1.678
2	2	0	1	0.93	75	7	4	7	5	0.2	0.2	2	5	3	1	1	20	N	Y	0.1	2.483	1.675
3	2	0	1	0.93	75	7	4	7	5	0.2	0.2	2	5	3	1	1	36	N	N		2.494	1.669
4	2	0	1	0.82	75	7	4	7	5	0.2	0.2	2	5	3	1	1	37	N	N		2.498	1.659
5	1	1.4	1	0.88	75	7	4	7	4	0.2	0.2	2	5	10	1	1	20	N	Y	0.17	2.844	1.578
6	1	1.4	1	0.77	75	7	4	7	4	0.2	0.2	2	5	10	1	1	20	N	Y	0.17	2.846	1.576
7	1	1.4	1	0.65	75	7	4	7	4	0.2	0.2	2	5	10	1	1	20	N	Y	0.17	2.849	1.574
8	1	1.4	1	0.88	75	7	4	7	4	0.2	0.2	2	5	10	1	1	37	N	Y	0.17	2.869	1.570
9	1	1.4	1	0.65	75	7	4	7	4	0.2	0.21	2	5	10	1	1	37	N	Y	0.17	2.877	1.565
10	1	1.4	1	0.65	75	7	4	6	4	0.2	0.28	2	5	10	1	1	37	N	Y	0.17	2.894	1.563
11	1	1.4	1	0.62	75	7	4	6	4	0.2	0.28	2	5	10	1	1	37	N	Y	0.32	2.903	1.560
12	1	1.4	1	0.62	75	7	4	6	5	0.2	0.31	2	5	10	1	1	37	N	Y	0.32	2.913	1.559
13	1	1.4	1	0.74	75	7	4	7	5	0.2	0.43	2	5	10	1	1	37	N	Y	0.32	2.933	1.558
14	1	1.4	1	0.74	75	7	4	7	5	0.2	0.43	2	5	10	1	1	37	N	Y	0.47	2.940	1.555
15	1	1.4	1	0.62	75	7	4	7	5	0.2	0.43	2	5	10	1	1	37	N	Y	0.47	2.950	1.553
16	1	1.4	1	0.51	75	7	4	7	5	0.2	0.43	2	5	10	1	1	37	N	Y	0.47	2.968	1.551
17	1	1.4	1	0.43	75	7	4	7	5	0.2	0.43	2	5	10	1	1	37	N	Y	0.47	2.987	1.551
18	1	7.1	1	0.55	75	6	4	7	5	0.2	0.54	2	5	10	1	1	37	N	Y	0.61	3.002	1.549
19	1	1.4	1	0.55	75	6	4	7	5	0.2	0.54	2	5	10	1	1	37	N	Y	0.61	3.005	1.548
20	1	2.1	1	0.55	75	7	4	7	5	0.2	0.64	2	5	10	1	1	37	N	Y	0.61	3.029	1.547
21	1	1.4	1	0.43	75	7	4	7	5	0.2	0.62	2	5	10	1	1	37	N	Y	0.61	3.060	1.546
22	1	2.1	1	0.55	75	7	4	7	5	0.2	0.51	2	4	10	1	1	37	N	Y	0.61	3.091	1.545
23	1	0	1	0.55	75	7	4	7	5	0.2	0.67	2	4	10	1	1	37	N	Y	0.61	3.127	1.543

- The two allowed structural configurations have caused the two isolated Pareto zones. Pareto zone *A* with solutions No. 1 to 4 has the steel frame system; Pareto zone *B* with solutions No. 5 to 23 has the concrete frame system. The steel frame system has lower cost but higher environmental impacts than the concrete frame system. Within each Pareto zone, different variable values such as insulation materials and window ratios lead to different Pareto solutions.

- The orientation angle has almost converged to zero because this orientation is helpful for the energy performance of buildings elongated on the east-west axis without any extra cost. The shape of the building has converged on a rectangular shape over an L-shape. This is because a rectangular shape has a more compact floor plan with smaller external envelope area, thereby reducing initial construction cost, environmental impacts due to material production and construction, and the operating energy consumption as well. For the rectangular shape, the aspect ratio r_0 varies with different Pareto solutions. This indicates that the optimal values for the aspect ratio are different for life-cycle cost and life-cycle environmental impacts. For example, only the aspect ratio is changed for solutions with ID between 14 and 17 (Table 7.8). A value close to 1 is favourable for cost reduction because a square shape has the minimum exterior envelope surface. However, compared with the square shape, a rectangular shape with the long side towards the south is better for energy performance.

- For all solutions, the wall tilt angle of each façade has converged to the lower bound value of 75°. In contrast to the design with vertical walls, the lower bound of wall tilt decreases the roof area by about 17% and increases the wall area by only about 3%. Since the extra construction cost associated with oblique wall is not considered in this study, the initial cost of the building is significantly reduced due to the reduction of roof area. Furthermore, the reduced envelope area reduces the embodied energy in the building materials.

- The window type on façade 2 (south orientation) has converged to the window type 4. Window types on the other three facades did not converge to a single value: the window type 7 is the most recommended on façade 1 (east) and façade 3 (west), while the window types 4 and 5 are used on façade 4 (north). The window types on facades 1, 3, and 4 did not converge because the objective functions are insensitive to the change between the four window types 4, 5, 6, and 7, all of which are double glazing with different coatings in terms of emissivity and position. To support the above claim, parametric analysis is performed to study the impact of the window type on the performance criteria. For example, for the solution with ID equal to 10, the window type on facades 1, 3, and 4 is changed to the other three possible values sequentially while all other variables are kept constant. The results are presented in Table 7.9. Similar analyses have also been made for other solutions. The parametric analyses show that: (1) the low emissivity coating on the outside pane is preferred in the cooling season while the coating on the inside pane is preferred in the heating season; (2) for both outside and inside panes, the windows with coating emissivity equal to 0.1 have lower cooling but higher heating load than the windows with coating emissivity equal to

0.2; (3) the window type 7 on the three facades (east, west, and north) usually can achieve slightly better performance for the total energy consumption and the two considered criteria in this study; however, the difference is negligible (less than 0.01%) for both LCC and LCEI. Since GA is not good at local tuning, the flat surface of the objective functions caused the diversity of window types on the north, east, and west facades.

- The window ratio on the north, east, and west facades has converged to the lower bound of 0.2 while it varies between 0.2 and 0.67 on the south facade. If all other variables keep constant, the life-cycle cost increases with an increase of the window ratio on the south façade because windows cost more than the opaque wall; the life-cycle environmental impact decreases with an increase of the window ratio on the south facade because the solar gains in the heating season are more important for a cold climate to reduce the energy consumption.

- There should be no overhang on the north, east, and west facades because (1) there is no direct sun on the north façade; and (2) the solar angle is low for the east and west facades. However, overhang is a suitable passive solar technique for south-facing windows to block sunlight out of the building in the summer and to permit sunlight into the building in the winter. The overhang depth is related to the window ratio: larger window ratio requires longer overhang projected out of the wall.

- The steel-stud wall is the optimal wall type for all solutions in the two Pareto zones. The steel-stud wall costs much less than the concrete block wall; for example, the unit cost for the same thickness (100 mm) of concrete block wall and steel-stud wall is about 57 and 13 \$/m^2 (R.S. Means 2004a), respectively. The remarkable cost difference leads to the use of the steel-stud wall in Pareto zone *A*, which favours the design with lower cost. The compromise between the operating and embodied environmental impacts can explain why the steel-stud wall is the optimal solution for Pareto zone *B*, which favours the design with lower environmental impacts. Compared with the steel-stud wall, the concrete block wall has a heavier thermal mass, which plays a positive role to reduce operating energy consumption. However, the positive effects are less than the associated increase of the embodied environmental impacts due to material production and construction. The above point can be verified with the following example. For the last solution (ID=23) in Table 7.8, if its wall type is changed from the steel-stud wall to the concrete block wall with vinyl cladding and 152 mm extruded polystyrene, the annual energy consumption reduces from 67730 kWh to 66900 kWh, but the life-cycle environmental impact increases from $1.543*10^7$ MJ to $1.576*10^7$ MJ.

- For the steel-stud wall type, most Pareto solutions have vinyl cladding as the optimal selection because it has the lowest initial cost among all the alternatives of cladding. Two Pareto solutions (ID No. 22 and 23) have identified wood siding as the optimal value because wood siding has less embodied environmental impacts than vinyl cladding. In addition, the lower thermal conductivity of wood siding contributes to reducing operating energy consumption. Regarding the insulation of the steel-stud wall, the cavity insulation converged to the minimum thickness (102 mm) of available alternatives while the rigid insulation converged to the maximum thickness (76 mm). This indicates that it is the cost-effective way to use rigid insulation to achieve a given thermal resistance because of two reasons: (1) cavity insulation must be installed between steel studs, and the total cost of steel-stud and the cavity insulation is much higher than that for rigid insulation; (2) the thermal bridge due to steel studs reduces the thermal performance of the cavity insulation. In terms of insulation materials, fiberglass is used as the cavity insulation material by the optimal solutions because it has much less embodied environmental impacts than rockwool, at almost equivalent thermal properties and construction cost. EPS (Expanded polystyrene) is used as the rigid insulation in Pareto zone *A* because it has lower cost and embodied environmental impacts than XPS (extruded polystyrene). On the contrary, XPS is used in Pareto zone *B* because its lower thermal conductivity leads to more positive effects of operating energy consumption reduction than the increase of embodied environmental impacts.

- The conventional roof is the optimal roof type for both steel-frame and concrete-frame structural systems. The green roof is not the optimal solution for this case study because of the following reasons. First, the green roof has much higher initial cost than the conventional roof; for example, for the OWSJ (open web steel joist) roof system, the unit cost is about 190 and 77 $\$/m^2$, respectively for the green roof and the conventional roof in this case study. Second, the green roof causes more embodied environmental impacts than the conventional roof because the structure needs to be strengthened to bear the additional loading of green roof layers. The additional loading is estimated to be about 170 kg/m^2 when the growing medium is at the saturation state (Peck and Kuhn 2001). Third, although the green roof can reduce operating energy consumption, this reduction may not offset the increase of embodied environmental impacts; therefore, the green roof is not the optimal solution even for Pareto zone *B*, which prefers environment-friendly design. An example can clearly illustrate the impact of the roof type on the performance criteria. For the first (ID=1) and the last (ID=23) solutions in Table 7.8, the roof type is changed to the green roof while keeping the same insulation level. The resulting changes of performance criteria are listed in Table 7.10, which compares the annual operating energy, the environmental impacts, and the life-cycle cost before and after the change of the roof type. It is found that the change to the green roof can achieve lower life-cycle environmental impacts for the first design alternative (ID=1) but not for the last design alternative (ID=23). This indicates that the ability of the green roof to reduce operating energy consumption depends on the base design. Since the last design alternative has much heavier thermal mass and more insulation than the first design alternative, the green roof leads to less noticeable energy reduction for the last design alternative (67730-67220=250 kWh) than that for the first one (76190-74240=1950 kWh). One can find that the solution with ID equal to 1-1 in Table 7.10 is non-dominated by all solutions in Pareto zone *A* in Table 7.8. This demonstrates that the green roof could be used by some Pareto solutions for a lightweight building design with the steel structural system and steel-stud walls. Note also that the green roof could become the optimal solution if the considered benefits of the green roof are expanded from energy savings to include other positive impacts such as the prolonged life of roof membrane and the moderation of urban heat island effect.

- For the roof insulation, almost all solutions have converged to two alternatives: 152 mm expanded polystyrene and 127 mm polyisocyanurate. Because the building in this case study has a large area of roof subject to outdoor weather conditions, the roof insulation plays an important role in the energy performance of this building; therefore, the optimal insulations are the thickest ones of all available choices. For the two kinds of insulation materials, expanded polystyrene has lower embodied impacts but a higher thermal conductivity than polyisocyanurate. This can explain that expanded polystyrene is used in the two Pareto zones by those solutions favourable of environmental performance.

Table 7.9 Impact of the window type on the operating energy consumption, LCC, and LCEI

ID	window types (side 1, 2, 3,4)	Annual operating energy (kWh)	Heating energy (kWh)					Cooling energy (kWh)			LCC ($)	LCEI (MJ)
			Nov.	Dec.	Jan.	Feb.	Mar.	Jun.	Jul.	Aug.		
10	(7,4,6,4)	68640	1110	2560	3360	4080	1420	2790	3280	3240	289370	15632800
10-1	(4,4,6,4)	68660	1090	2540	3330	4030	1380	2850	3340	3290	289410	15635600
10-2	(5,4,6,4)	68670	1110	2560	3360	4070	1410	2810	3300	3250	289410	15637200
10-3	(6,4,6,4)	68640	1090	2550	3340	4040	1380	2840	3330	3280	289370	15631700
10-4	(7,4,4,4)	68650	1109	2560	3360	4070	1420	2810	3290	3250	289370	15634500
10-5	(7,4,5,4)	68650	1120	2570	3380	4110	1440	2760	3250	3210	289370	15633000
10-6	(7,4,7,4)	68640	1130	2580	3390	4120	1450	2750	3230	3200	289360	15631700
10-7	(7,4,6,5)	68650	1120	2570	3380	4100	1440	2770	3260	3220	289370	15633100
10-8	(7,4,6,6)	68640	1110	2560	3370	4080	1420	2790	3270	3230	289360	15631200
10-9	(7,4,6,7)	68630	1120	2570	3380	4100	1440	2760	3250	3210	289350	15630400

Table 7.10 Impact of the roof type on energy consumption, environmental impacts, and LCC

ID	Roof type	Annual operating energy (10^4 kWh)	Environmental impacts			LCC (10^5 \$)
			operating (10^7 MJ)	embodied (10^6 MJ)	life-cycle (10^7 MJ)	
1	conventional roof (OWSJ)	7.619	1.446	2.323	1.678	2.480
1-1	green roof (OWSJ)	7.424	1.409	2.453	1.654	2.926
23	conventional roof (CIP)	6.773	1.286	2.577	1.543	3.127
23-1	green roof (CIP)	6.722	1.276	2.696	1.545	3.577

Some additional information can be obtained if the Pareto solutions are analyzed in terms of the detailed constituents of several performance criteria including annual operating energy consumption, life-cycle cost, and life-cycle environmental impacts. Since different solutions belonging to the same Pareto zone have similar characteristics, a representative design alternative is selected from each of the two Pareto zones to facilitate analysis. Hence, the 3rd and the 14th solutions in Table 7.8 are chosen to represent the Pareto zone *A* and *B*, respectively, because they are located in the median or near median position of the corresponding Pareto zone.

The operating energy end-use sectors considered in this study include lighting and equipment, heating, and cooling. The distribution of annual energy consumption among the above three sectors are illustrated in Figure 7.6. This figure shows that:

- Lighting and equipment consume a large portion (around 65%) of operating energy consumption for office buildings.

- Because of the cold climate in Montreal, the office building in this case study consumes more energy for heating than for cooling.

- The proportion of heating energy consumption reduces from 25% in Pareto zone *A* to 18% in Pareto zone *B* because the design alternatives in Pareto zone *B* have more energy efficiency measures such as the increased use of insulation, thermal mass, and window areas on the south façade. However, the energy efficiency measures have caused the proportion of cooling energy increases slightly and the proportion of lighting and equipment increases from 62% to 68%.

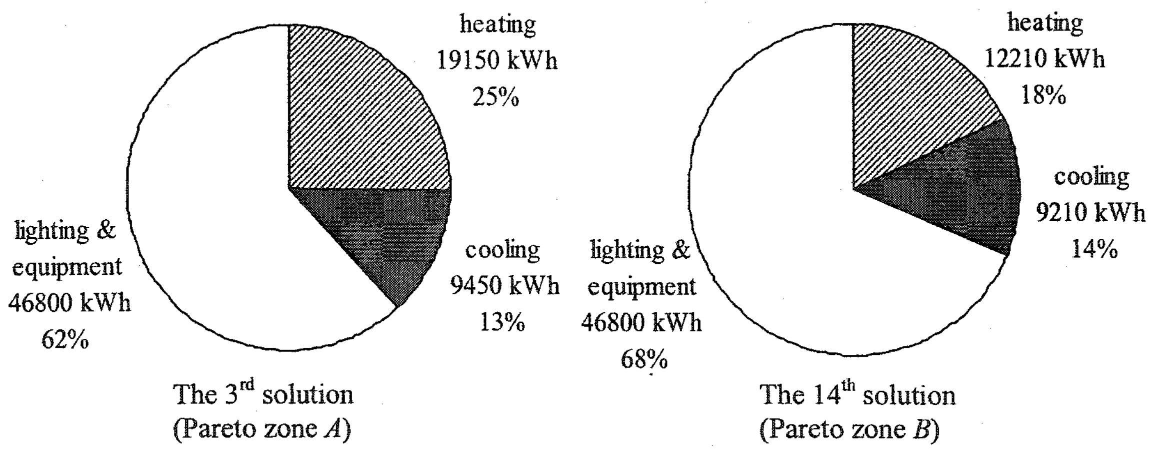


Figure 7.6 Distribution of operating energy consumption among heating, cooling, and lighting & equipment

The breakdown of the life-cycle cost for the 3rd and the 14th solutions is given in Table 7.11, where the percent is indicated in the parentheses. The initial cost is decomposed into the cost of opaque walls, roof, floor, windows, and overhangs. The operating cost is decomposed in two ways: (1) energy consumption cost vs. demand cost; and (2) cost in the heating season vs. in the cooling season. It can be seen from this table that:

- The initial cost takes a large portion (more than 60%) of the life-cycle cost. This is due to two reasons: (i) the electricity price in Montreal is low; and (ii) the structural components for floor and roof have high construction costs.

- Most initial costs come from the roof and floor because of the following reasons. First, the roof and floor in this case study have higher unit cost than the opaque wall; for example, for the representative design alternative in Pareto zone B, its floor and opaque wall cost about 150 and 78 $\$/m^2$, respectively. Second, the roof and floor have much larger areas than the opaque walls and windows; for example, the representative design alternative in Pareto zone *B* has about 234 m^2 areas of opaque walls, 86 m^2 of windows, 416 m^2 of roof, and 500 m^2 of floor.

- Demand cost is a considerable component of the life-cycle cost, and it is even higher than the energy consumption cost in this case study. For example, the representative design in Pareto zone *A* has a demand cost of 46960 \$ and an energy consumption cost of 44430 \$. This indicates that it is important to incorporate demand cost whenever the life-cycle cost is optimized for a building subject to demand charges.

- The heating season contributes more to the operating costs than the cooling season.

- Due to the use of more insulation and the change of structural system from steel frame to concrete frame, the proportion of initial cost increases from about 63% in Pareto zone *A* to 72% in Pareto zone *B*, and the proportion of the operating cost decreases by an equivalent amount.

The breakdown of the life-cycle environmental impact for the 3rd and the 14th solutions is listed in Table 7.12, where the percent is indicated in the parentheses. The composition of the life-cycle environmental impact is analyzed from two aspects: (1) the components of expanded cumulative exergy consumption (i.e., cumulative exergy consumption and abatement exergy consumption); and (2) the life-cycle stages (i.e., pre-operation stage and operation stage). Table 7.12 shows that:

- For both Pareto zones, CExC (cumulative exergy consumption) constitutes as large as about 93% of the life-cycle environmental impact of a building, while abatement exergy consumption is only about 7%. This situation is mainly attributed to two reasons: (i) the electricity consumed in the operation stage is generated mostly from hydro; hence, the electricity has low waste emissions; and (ii) this study limits to those emissions with global and long-lasting impacts on the environment while the waste emissions have local and transient impacts are not considered.

- Compared with the non-fuel exergy consumption, the fuel exergy consumption plays a dominant role in the CExC.

- The pre-operation stage as indicated by the embodied environmental impacts in Table 7.12 contributes to about 15% of the life-cycle environmental impact, while the building operation stage as indicated by the operating environmental impacts in the table takes the remaining 85%.

- Although the proportion of CExC in the pre-operation stage is much less than that in the operating stage (about 12% vs. 80%), the proportion of the abatement exergy consumption in the pre-operation stage is comparable with that in the operating stage. The major reason is due to the dominant share of hydro-electricity for building operations in Montreal, Quebec.

- As the Pareto zone changes from *A* to *B*, the proportion of the total CExC over the whole life cycle decreases slightly (about 1%) because of the improved energy performance of the building. For the same reason, the contribution of operating environmental impacts decreases about 3% from Pareto zone *A* to *B*.

Table 7.11 Constituents of life-cycle cost of representative solutions in the two Pareto zones

ID	Pareto zone	IC ($)					OC ($)		OC ($)	
		walls	roof	floor	windows	overhangs	consumption	demand	heating season	cooling season
3	A	158,000 (63%)					91,400 (37%)		91,400 (37%)	
		18,670 (7%)	57,740 (23%)	62,390 (25%)	19,180 (8%)	0	44,430 (18%)	46,960 (19%)	62,780 (25%)	28,610 (12%)
14	B	211000 (72%)					83,000 (28%)		83,000 (28%)	
		18,640 (6%)	87,130 (30%)	76,760 (26%)	25,820 (9%)	2,660 (1%)	40,200 (14%)	42,760 (14%)	55,260 (19%)	27,710 (9%)

Table 7.12 Constituents of life-cycle environmental impacts of representative solutions in the two Pareto zones

ID	Pareto zone	Life-cycle environmental impacts (MJ)			Life-cycle environmental impacts (MJ)			
		life-cycle CExC (MJ)		life-cycle abatement exergy (MJ)	operating environmental impacts (MJ)		embodied environmental impacts (MJ)	
		Fuel	Non-fuel materials		CExC	abatement exergy	CExC	abatement exergy
3	A	15,597,000 (93.5%)		1,092,500 (6.5%)	14,311,000 (85.8%)		2,378,800 (14.2%)	
		15,519,000 (93%)	78,000 (0.5%)		13,743,900 (82.4%)	567,100 (3.4%)	1,853,500 (11.1%)	525,400 (3.1%)
14	B	14,395,700 (92.6%)		1,157,800 (7.4%)	12,949,000 (83.3%)		2,604,500 (16.7%)	
		14,278,000 (91.8%)	11,7800 (0.8%)		12,435,900 (80%)	513,100 (3.3%)	1,959,800 (12.6%)	644,700 (4.1%)

The embodied environmental impacts can be studied further according to the building assembly groups such as walls and roof. Figure 7.7 illustrates the distribution of embodied environmental impacts among the assembly groups for the representative design alternatives in the two Pareto zones. This figure shows that:

- The roof and floor play the dominant role in the pre-operation stage as they together contribute about 90% of the total embodied environmental impacts. Although they have the similar structural components, the roof has more environmental impacts than the floor because the roof has several additional layers such as insulation, vapor barrier, and acoustical tile.

- The opaque walls contribute about 9% of the total embodied environmental impacts. Windows and overhangs have almost negligible environmental impacts because of their small amount in the building design.

- The proportions of embodied environmental impacts for the roof and windows increase slightly as the Pareto zone changes from A to B. This is because more roof insulation and windows are used.

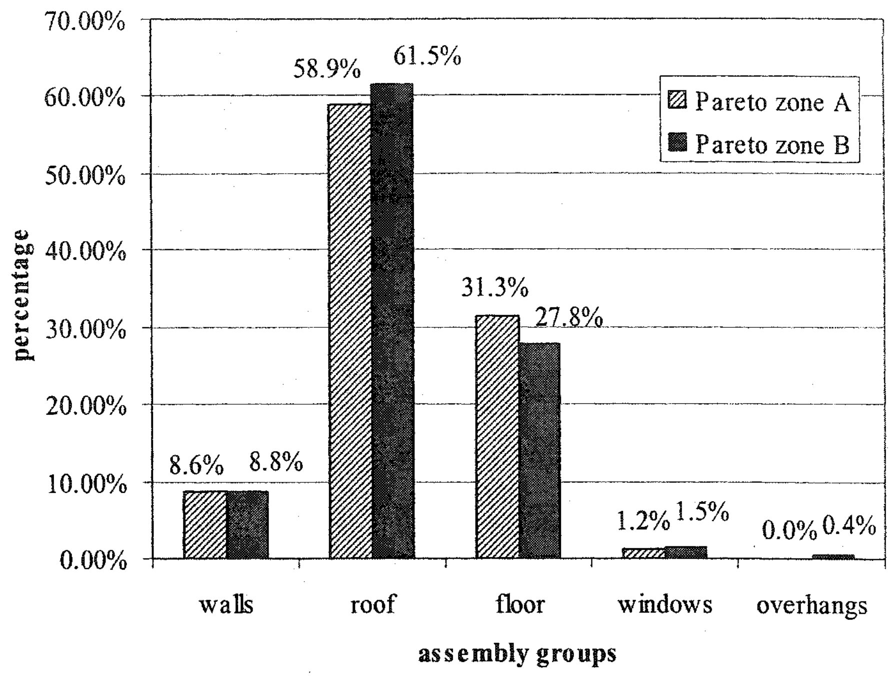


Figure 7.7 Distribution of embodied environmental impacts among assembly groups

7.3 Investigation of the Impact of Performance Criteria on Results

The previous section presented the optimal solutions for the multi-objective optimization problem with LCC and LCEI as the two performance criteria. This section aims to investigate the impact of changing performance criteria on the optimal results. The investigation is carried out by replacing the LCC or LCEI with several other commonly used performance criteria, including IC (initial cost), LCEnergy (life-cycle energy), and AnnualEnergy (annual operating energy consumption). Thus, the following three cases are studied:

(1) IC and LCEI are the two objective functions by replacing LCC with IC;

(2) LCC and LCEnergy are the two objective functions by replacing LCEI with LCEnergy;

(3) LCC and AnnualEnergy are the two objective functions by replacing LCEI with AnnualEnergy;

The same variables and optimization parameters are used in each case as those of the reference case in the previous section. For each case, the optimization is run three times, and all non-dominated solutions from the external populations in the three runs form a global Pareto front as shown in Figures 7.8, 7.9, and 7.10.

All these figures have isolated regions along the Pareto fronts. In Figure 7.8 and 7.9, the use of different structural systems caused the disconnected Pareto zones: the steel frame structural system is used in Pareto zones A-1 and A-2, which has lower cost; the concrete fame system is used in Pareto zones B-1 and B-2, which has better environmental performance. In Figure 7.10, besides the structural systems, the use of different wall types is another reason behind the disconnected Pareto zones: the steel frame structural system is used in Pareto zone A-3 while the concrete frame system is used in the other two Pareto zones; the concrete block wall is used in Pareto zone C-3 while the steel-stud wall is used in other two Pareto zones.

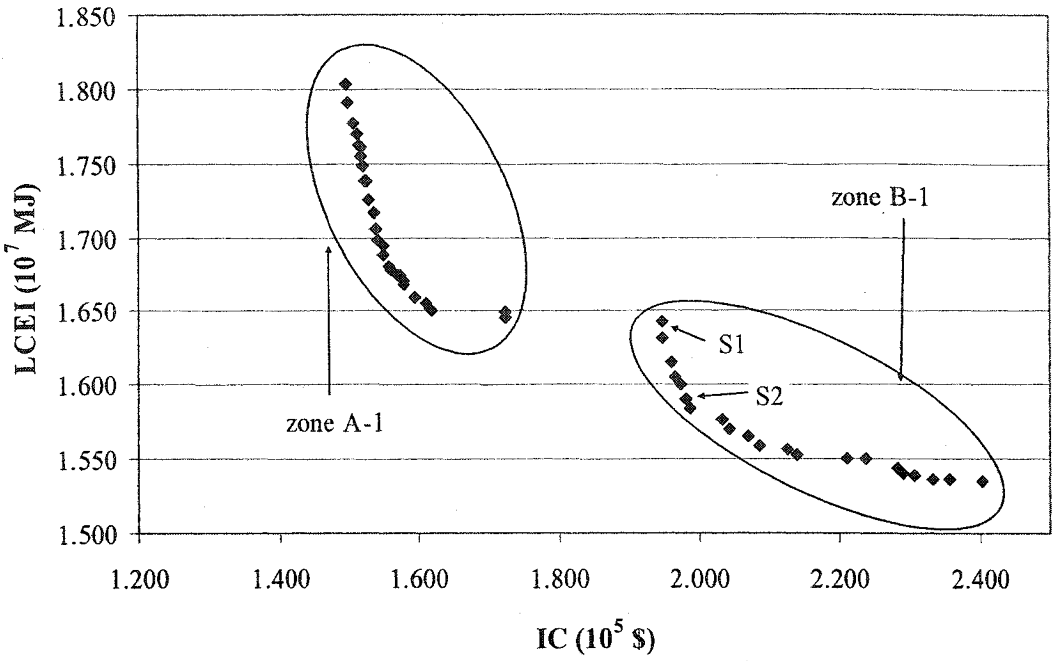


Figure 7.8 Pareto front for IC and LCEI optimization

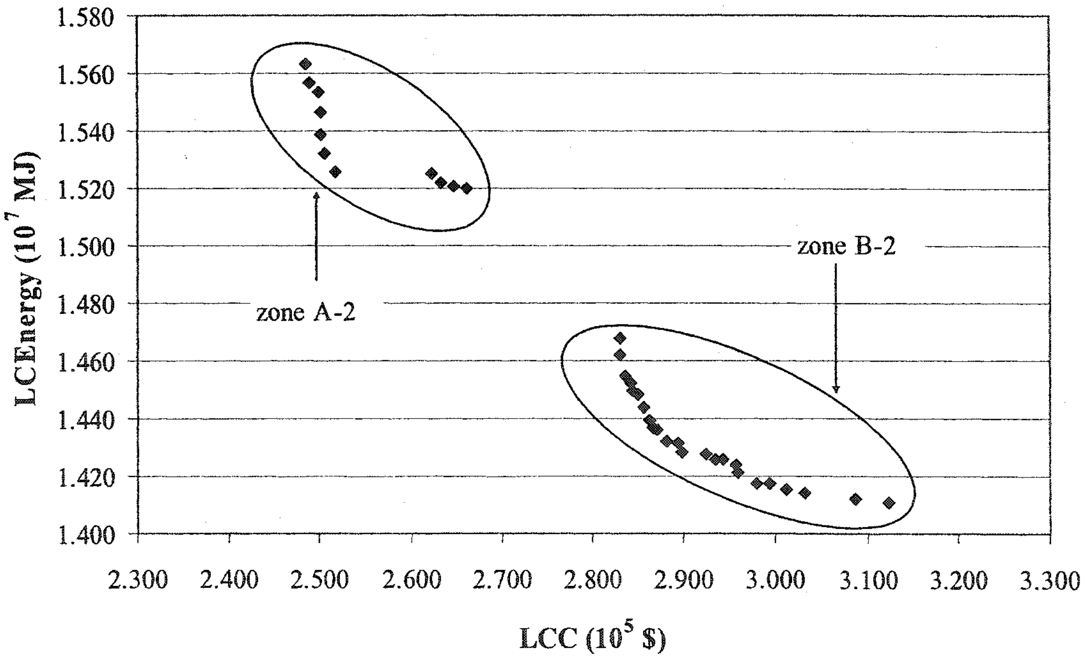


Figure 7.9 Pareto front for LCC and LCEnergy optimization

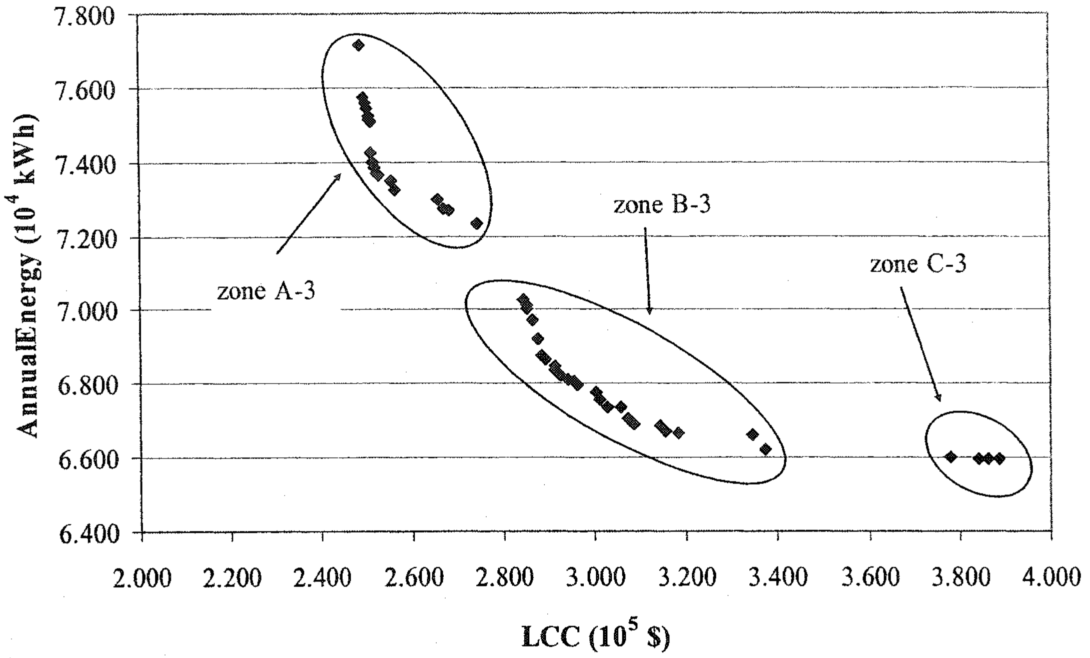


Figure 7.10 Pareto front for LCC and AnnualEnergy optimization

The three investigated cases can be compared with the reference case by depicting the non-dominated solutions in the same LCC-LCEI performance space, as shown in Figures 7.11, 7.12, and 7.13.

- When the IC and LCEI are used as the performance criteria, the non-dominated solutions have a more wide range of LCC and LCEI than those solutions obtained in the reference case (see Figure 7.11). This is mainly because compared with the LCC, the IC leads to a more diverse set of Pareto solutions in terms of insulation levels. For example, some solutions near the upper left edge of the Pareto front have located 25 mm and 51 mm EPS as the optimal rigid insulation level of the steel-stud wall; however, these insulations are not used by the Pareto solutions of the reference case.

- When the LCC and LCEnergy are used as the performance criteria, the Pareto front is close to that obtained in the reference case (see Figure 7.12). This indicates that in this case study, the optimal solutions are almost the same for LCEI and LCEnergy.

- When the LCC and AnnualEnergy are used as the performance criteria, some solutions are far away from the Pareto front obtained in the reference case (see Figure 7.13). This is because the variables may take different optimal values for the minimization of AnnualEnergy and LCEI. Since the pre-operation stage is not considered in optimizing annual operating energy consumption, building materials with high embodied environmental impacts may be used in the third investigated case. For example, several solutions far away from the Pareto front have identified the concrete block wall as the optimal wall type; however, this wall type is not used by the Pareto solution in the reference case.

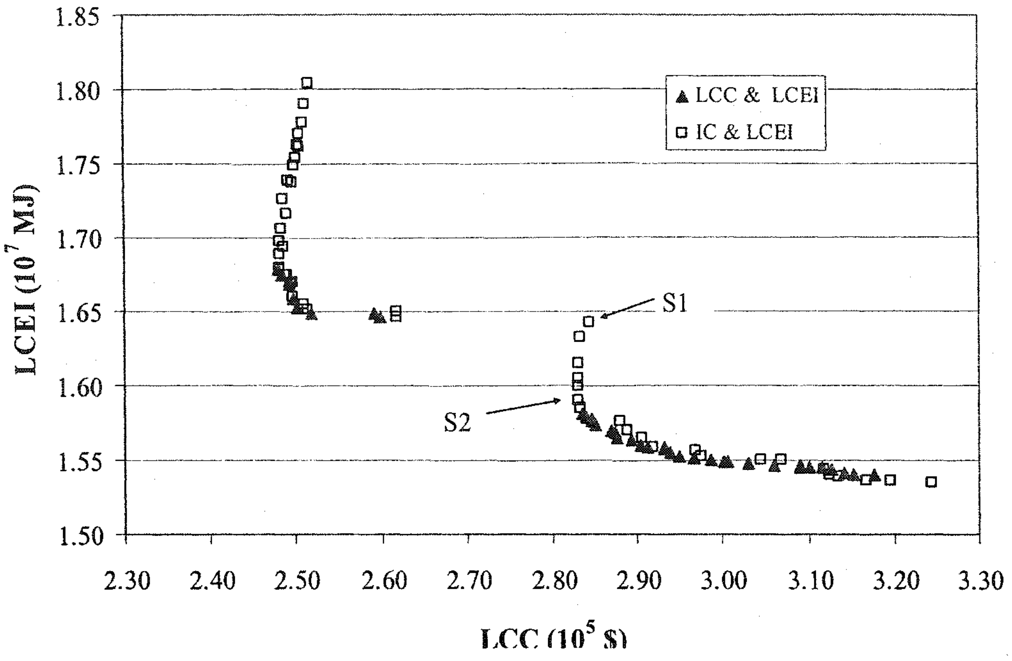


Figure 7.11 Comparison of non-dominated solutions in the LCC-LCEI performance space between the first case and the reference case

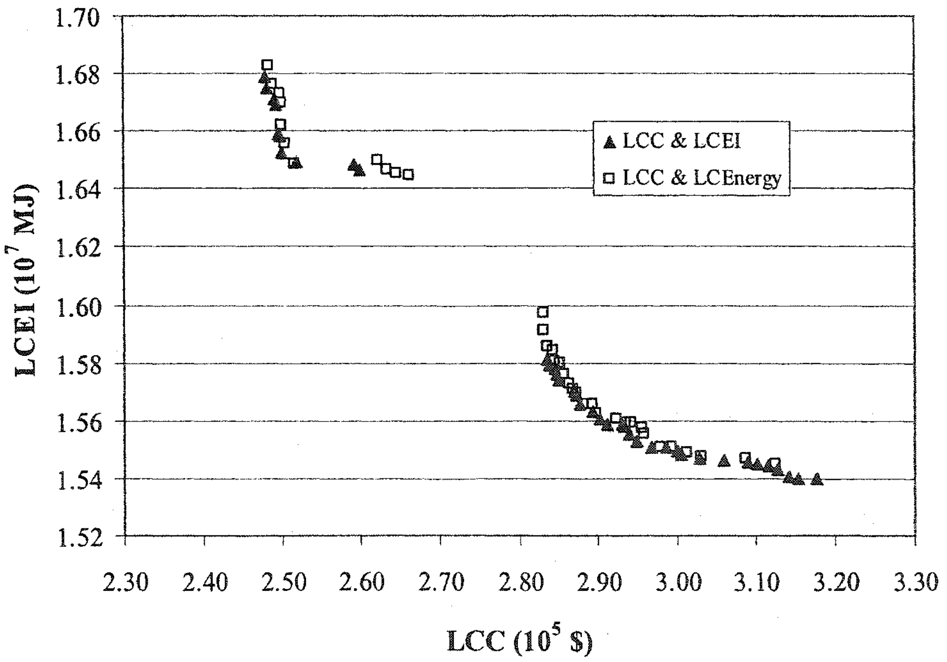


Figure 7.12 Comparison of non-dominated solutions in the LCC-LCEI performance space between the second case and the reference case

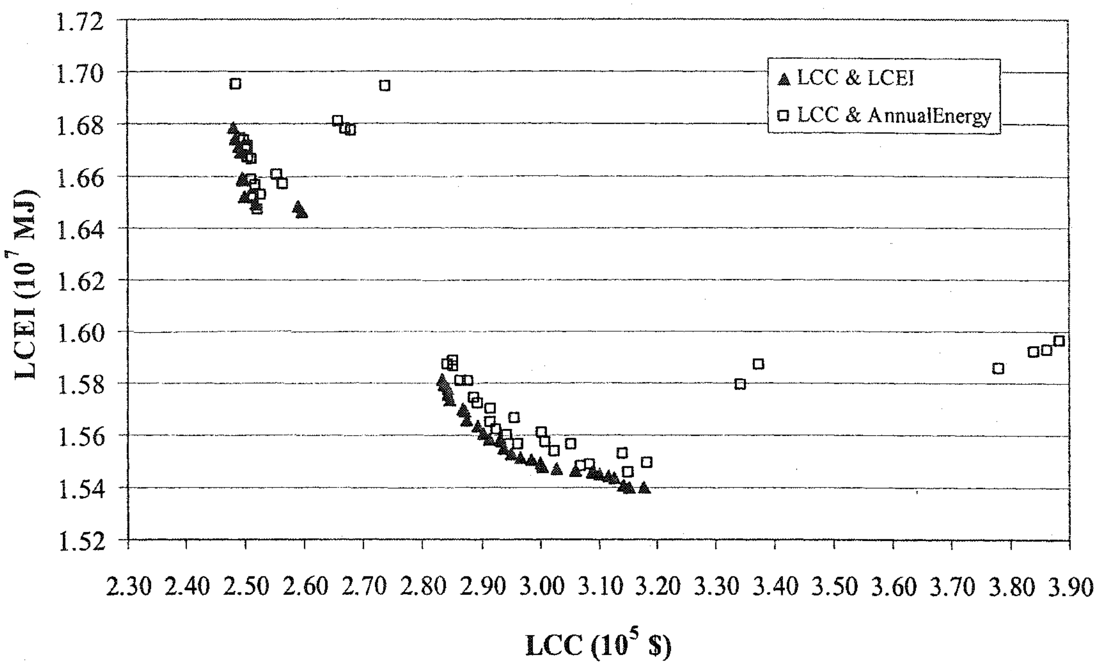


Figure 7.13 Comparison of non-dominated solutions in the LCC-LCEI performance space between the third case and the reference case

The investigation study demonstrates that non-dominated solutions obtained with IC or AnnualEnergy as one of the multiple performance criteria may not keep their non-dominated relationship in view of the life-cycle analysis. In Figure 7.8, for example, the solution S_1 (IC=$1.982*10^4$ \$, LCEI=$1.590*10^7$ MJ) and the solution S_2 (IC=$1.947*10^4$ \$, LCEI=$1.643*10^7$ MJ) are non-dominated in the IC-LCEI performance space; however, S_1 (LCC=$2.831*10^4$ \$) dominates solution S_2 (LCC=$2.842*10^4$ \$) in the LCC-LCEI space. The same phenomenon is observed when the solutions from LCC-AnnualEnergy optimization are evaluated in terms of LCC and LCEI. Therefore, life-cycle optimization is important to avoid choosing partial design alternatives.

7.4 Summary

The case study has shown that the implemented optimization program is able to find the optimal or near optimal solutions from a large design space. The Pareto front obtained from the multi-objective genetic algorithm can be used in a number of ways in the decision-making process. First, it can be used to get information about the best values for each criterion. This information is useful to set a reasonable target or constraint with respect to selected criterion in the conceptual design phase. Second, with predefined constraints for one criterion, the Pareto front can be used to determine the optimal value for the other criterion. Third, the Pareto front can be used to investigate the trade-off relationship between the two criteria.

The case study has demonstrated that the multi-objective optimization can capture the complicated interactions among variables. For example, the overhang depth is related to the window ratio and orientation, and this relationship is clear when a group of Pareto solutions are compared. In addition, the variables have different converging situations: some variables such as building shape and orientation converge to almost the same values for all Pareto solutions; some others such as aspect ratio and insulation materials vary with different Pareto solutions.

It is important to seek for the optimal design in view of the whole life cycle of a building. The investigation study shows that solutions obtained with the initial construction cost and annual operating energy consumption may become dominated if they are evaluated in terms of the LCC and LCEI.

Chapter 8
Summary, Contributions, and Future Work

8.1 Summary

Improvements in the environmental performance of buildings play an important role in the pursuit of a sustainable society. Many decisions that highly influence building performance are made at the conceptual design phase when numerous competitive alternatives are generated and compared. The large quantity of potential design alternatives make it essential to have suitable tools that can assist designers in finding an optimal or near optimal design efficiently. GBOptimizer is such a tool developed in this research to carry out simulation-based optimization for green building design.

A prerequisite for building optimization is to define a reasonable system boundary so that the established optimization model is manageable while keeping the major interactions among different elements of a building design. In this research, the optimization scope focuses on the building envelope, which has considerable impacts on the performance of buildings. The life-cycle assessment methodology is employed to evaluate the environmental performance of a building. Impact categories considered include resource depletion, global warming, and acidification, all of which have global, regional, and long-lasting impacts on the environment. These categories are unified together with expanded cumulative exergy consumption (see Chapter 3), which is the sum of cumulative exergy consumption and abatement exergy consumption. Cumulative exergy consumption evaluates resource depletion including fuels and non-fuel materials. Abatement exergy consumption evaluates waste emissions, which include three major greenhouse gases (CO_2, CH_4, N_2O) for global warming and two major acidic gases (SO_x and NO_x) for acidification.

Within this research scope, the optimization model discussed in Chapter 3 includes the following major aspects:

- This optimization research considers geometry-related variables (e.g., orientation, building shape, and aspect ratio), envelope configuration related variables (e.g., window type, window ratio, wall type, and wall layers), overhang-related variables (e.g., overhang height and overhang depth), and structure-related variables (e.g., structural system). Some variables (e.g., orientation and overhang depth) can take either continuous or discrete values while some others (e.g., window type and wall layers) can take only discrete values. The concept of structured variable is employed to capture the interdependency between some variables such as the wall type and wall layers.

- This optimization research minimizes two major objective functions: the life-cycle cost and the life-cycle environmental impact. The former includes the initial construction cost and the operating cost; the latter includes the embodied environmental impacts in the pre-operation stage (i.e., natural resource acquisition, material production, on-site construction, and transportation) and the operating environmental impacts in the operation stage. Besides the two major objective functions, the optimization models also support several other commonly used objective functions for minimization: the annual operating energy consumption, the life-cycle energy, the initial cost, and the operating cost.

- This optimization research has three types of constraints: box constraints for continuous variables, selection constraints for discrete variables, and functional constraints involving more than one variable. All functional constraints are converted from the objective functions by stipulating right hand values.

Different combinations of variables, objective functions, and functional constraints can result in a variety of optimization problems, which can be defined and solved with the optimization system. The simulation-based optimization system is made of four components (Chapter 4):

- the text-based input and output, used to customize an optimization problem and to obtain results;
- the simulation programs, used to evaluate objective functions and functional constraints;
- the data files, used to store the data (e.g., initial cost and embodied energy) required by the simulation programs;
- the optimizer, used to find the optimal solutions in the design space.

Through the input text files, an optimization problem can be customized by defining the variables, the objective functions, the functional constraints, the optimization parameters, and the simulation parameters. The ASHRAE toolkit for building load calculations is employed in this research as the basis of the simulation programs. Based on the hourly building loads from the toolkit, extension work is carried out to derive the objective function values. The initial construction cost data are mainly from the R.S. Means cost data books, and the embodied environmental impact data such as embodied energy and CO_2 emissions are manually extracted from the commercial life-cycle assessment program Athena EIE. For the optimizer, genetic algorithms are employed as the optimization technique because of their advantages in dealing with the following particularities of this optimization research: (i) a large design space to explore; (ii) many variables are discrete; (iii) the problem can be single- or multi- objective optimization with or without functional constraints; (iv) the objective functions may be not continuous. The existence of structured variables leads to the use of a special version of genetic algorithms, the structured genetic algorithm, which has a hierarchical binary representation in this study.

There are various simulation-based building optimization problems in practice. A small change in problem formulation (e.g., adding a new variable) could involve major changes for a rigid optimization system, which is developed for a particular application with little consideration of future reuse and expansion. On the other hand, an object-oriented framework can facilitate the reuse of code and can be easily adapted to solve many similar problems. Therefore, an object-oriented framework is presented in Chapter 5 for simulation-based optimization with genetic algorithms. This framework consists of three modules:

- the variables module used to define and instantiate variables;
- the simulation module used to implement user-developed simulation programs or to call external simulation programs;
- the optimizer module used to implement the optimization programs.

The customization process of the object-oriented framework is illustrated with the development of the simulation-based optimization system GBOptimizer (see Chapter 5). The implemented system supports unconstrained single objective optimization, constrained single objective optimization, and unconstrained multi-objective optimization. The extended simulation programs are verified by hand calculations, and the optimizer is validated with several mathematical optimization problems as discussed in Chapter 6.

The case study in Chapter 7 demonstrated the use of the system in green building design. The formulated optimization problem in this case study has a large design space involving two alternative shapes (rectangular and L-shape), two structural systems (steel frame and concrete frame), two wall types (steel-stud wall and concrete block wall), two roof types (conventional roof and green roof), many possible layers, dimensions, etc. Life-cycle cost and life-cycle environmental impact are the two objective functions to be minimized simultaneously. This problem is solved with the multi-objective genetic algorithm implemented in the system. The results show that the Pareto solutions are clustered into two disconnected regions (i.e., Pareto zones) in the performance space. The steel frame structural system is used by the solutions in the Pareto zone with lower cost, whereas the concrete frame is used by the solutions in the other Pareto zone with lower environmental impacts. Solutions in each Pareto zone may take different optimal values for some variables such as aspect ratio and insulation levels. In terms of the constituents of the two considered performance criteria, the initial cost has a larger proportion than the operating cost. For the operating cost, the demand cost is a considerable component, and it may be higher than the energy consumption cost. The pre-operation stage contributes to about 15% of the life-cycle environmental impact, while the building operation stage takes the remaining 85%. For both the pre-operation and the operation stages, most environmental impacts are due to the resource depletion evaluated by the cumulative exergy consumption.

8.2 Contributions

The contribution of this research to the area of optimization for green building design can be summarized as follows.

- This research is distinctive from previous studies because it pursues life-cycle optimization for green building design. Life-cycle assessment is one of the most important principles to evaluate the environmental performance of a building. However, most previous relevant studies fall in one of the following three branches: (1) case studies, which apply the life-cycle assessment methodology step by step to document the environmental performance of a building; (2) simulation programs, which implement the procedures of the life-cycle assessment for buildings; (3) optimization studies, which usually use operating energy consumption or life-cycle cost to evaluate the environmental performance of building design alternatives. Compared with previous studies, this research integrates the three branches to automatically find optimal or near optimal design alternatives for a building with respect to its environmental performance incorporating expanded impact categories over most of its life-cycle stages.

- This research proposed the use of expanded cumulative exergy consumption as the indicator to evaluate the environmental performance of a building (see Chapter 3). Since the life-cycle optimization considers a variety of impact categories including resource depletion, global warming, and acidification, the integration of these impact categories is usually required to reduce the number of objective functions. Most available integration methods require normalizing different impacts and setting weights for them. Both normalization coefficients and weights are subjective and have remarkable impacts on the conclusion. In contrast, the indicator of expanded cumulative exergy consumption does not need normalizing coefficients and weights, thereby reducing the impact of subjectivity on the results.

- This research suggested an easy-to-follow method to calculate cumulative exergy consumption for a building. As discussed in Chapter 3, the cumulative exergy consumption includes fuel exergy and non-fuel exergy. Fuel exergy is converted from embodied energy in the pre-operation stage and operating energy in the operation stage. Non-fuel exergy is based on the chemical exergy and the quantity of non-fuel resources. The proposed method advocates the use of the data from life-cycle assessment programs to calculate cumulative exergy consumption.

Therefore, compared with other methods in previous studies, the method in this research has two advantages: (1) it is simpler than those methods that need detailed process analyses using exergy; and (2) it improves the accuracy of the estimation method based on cumulative energy consumption by taking into account the non-fuel exergy.

- This research differs from most previous studies in terms of its use of structured variables such as wall types and roof types for the optimization of green building design (see Chapter 3). There are prevalent hierarchical relationships among building elements; hence, the incorporation of structured variables is a significant step to achieve the whole building optimization in the future so that competitive design alternatives at different levels of the building hierarchy can be compared at the early design stage. In addition, this research studied the following issues about the implementation of structured variables in the genetic algorithm (see Chapter 4): the shortened binary representation scheme, the modified crossover operation, and the use of mating restriction.

- This research made a thorough analysis about the simulation-based optimization system for green building design. Based on the system analysis, an object-oriented framework is proposed in Chapter 5. This framework considers many important aspects such as the simulation-optimization interfaces, the data organization, the post-processing of simulation results, and the running error problems of simulation programs. The system framework design can be used as a starting point to solve many other simulation-based optimization problems.

- This research contributes a flexible optimization system for green building design. Most previous studies were devoted to solving fixed optimization problems. The lack of flexibility is a major obstacle to apply previous research in design practice. In contrast, the system in this research has increased flexibilities in the following aspects: (1) it supports several optimization models including unconstrained single objective optimization, constrained single objective optimization, and unconstrained multi-objective optimization; (2) it offers a variety of performance criteria and functional constraints; (3) it does not limit the number of those sub-level variables (e.g., wall types and wall layers) affiliated to the structural system; (4) it offers the system users some freedom to change the data requirement (e.g., the waste emissions) without programming (see Chapter 5).

- This research demonstrated the advantages of multi-objective optimization for green building design through a case study. With multiple Pareto solutions available at the end of optimization, designers can grasp the trade-off relationship between the economical and environmental performance criteria. The trade-off relationship helps designers to select a compromised solution according to the marginal improvement of the environmental performance with additional costs.

8.3 Recommendations for Future Work

The system can be improved in a number of ways to make the optimization system more powerful. Future research into the following directions is of interest:

- Metamodels could be integrated into the system to reduce the computation time. The major reason of the long computation time lies in the use of a non-trivial simulation program. Metamodeling techniques such as response surface methodology and neural networks could be employed to approximate a simplified function relationship between the simulation results and the input variables. The approximated function then replaces the costly simulation to obtain the function values for a vector of variables.

- Local search methods could be used together with the genetic algorithms to find the optimal solutions. Because genetic algorithms are not good at local tuning, it consumes much time for them to improve solutions at later generations. Hence, the performance of the optimizer would be improved if the genetic algorithms are combined with some numerical optimization methods such as the Hook-Jeeves pattern search method for single objective optimization and the weighted linear utility function method for multi-objective optimization.

- The mechanical system could be incorporated into the optimization scope. As mentioned in Chapter 3, the building envelope and mechanical system can affect the environmental performance of a building significantly. This study has focused on the building envelope. The positive roles of the optimization system would be more evident if many parameters of the mechanical system were treated as variables.

- Many aspects such as daylighting and moisture could be considered in the simulation program. For example, the current study regards electrical lighting as a constant following a given schedule. This can be changed if daylighting is considered; therefore, the interactions between daylighting and cooling and heating could be modeled.

- Multiple attribute decision making techniques could be implemented in the system in order to facilitate selecting one design alternative from the set of Pareto solutions at the end of multi-objective optimization.

- The user-friendliness of the system could be improved in the following aspects: the text file *environment.txt* could be replaced by a graphical user interface; the results could be processed automatically to present the evolution process with figures; the data files in the system could be upgraded to databases.

- The object-oriented framework could be expanded to support constrained multi-objective optimization and to include several abstract classes used for stopping criteria, mutation, crossover, and the distance metrics between individuals.

References

Adalberth, K. (1997). Energy use during the life cycle of single-unit dwellings: Examples. *Building and Environment, 32*(4), 321-329.

Al-Rabghi, O.M., and Hittle, D.C. (2001). Energy simulation in buildings: Overview and BLAST example. *Energy Conversion and Management, 42*(13), 1623-1635.

Al-Homoud, M.S. (1994). *Design optimization of energy conserving building envelopes.* Ph.D. thesis, Texas A&M University, U.S.A.

Analysis and Modeling Group. (1999). *Canada's emissions outlook: An update.* Natural Resources Canada, Ottawa.

Anink,D., Bonstra, C., and Mak, J. (1996). Handbook of sustainable building: An environmental preference method for selection of material for use in construction and refurbishment. London, UK: James& James Ltd.

Annex 31. (2001). Directory of tools: A survey of LCA tools, assessment frameworks, rating systems, technical guidelines, catalogues, checklists and certificates [Technical Report]. Annex 31: Energy-Related Environmental Impact of Buildings.

Annex 37. (2003). *Low exergy systems for heating and cooling of buildings.* Retrieved June 2004, from: www.vtt.fi/rte/projects/annex37

ASHRAE. (1999). *ASHRAE Standard 90.1-1999: Energy standard for buildings except low-rise residential buildings.* Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

ASHRAE. (2001). *ASHRAE handbook of fundamentals.* Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

ATHENA EIE Version3.0. (2003). The ATHENA sustainable materials institute, Ottawa, Canada.

Athienitis, A.K., and Akhniotis, E. (1992). Methodology for computer-aided design and analysis of passive solar buildings. *Computer Aided Design, 25*(4), 203-214.

Ayres, R.U., Ayres, L.W., and Martinas, K. (1998). Exergy, waste accounting, and life-cycle analysis. *Energy, 23*(5), 355-363.

Baker,N., and Steemers, K. (2000). *Energy and environment in architecture: A technical design guide*. New York: E. & FN. Spon.

Balcomb, J.D. (1984). *Passive solar heating analysis: A design manual.* Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

Bansal, N.K., Hauser, G., and Minke, G. (1994). *Passive building design: A handbook of natural climatic control.* Amsterdam, The Netherlands: Elsevier Science.

Baouendi, R. (2003). *Development of a prototype tool for the evaluation of the sustainability of Canadian houses*. Master thesis, Department of Building, Civil, and Environmental Engineering, Concordia University, Montreal, Canada.

Barnthouse, L., Fava, J., Humphreys, K., Hunt, R., Laibson, L., Noesen, S., Norris, G., Owens, J., Todd, J., Vigon, B., Weitz, K., and Young, J. (1998). *Life-cycle impact assessment: The state-of-the-art* (2nd ed.). Pensacola, FL: SETAC press.

Bass, B., and Baskaran, B. (2003). *Evaluating rooftop and vertical gardens as an adaptation strategy for urban areas* [Technical Report], NRCC-46737. National Research Council of Canada, Ottawa. Retrieved June 2004, from: http://irc.nrc-cnrc.gc.ca/fulltext/nrcc46737/nrcc46737.pdf

Booch G., Rumbaugh J., and Jacobson I. (1998). *The unified modeling language user guide*, Reading, MA: Addison-Wesley.

Bouchlaghem, N., and Letherman, K.M. (1990). Numerical optimization applied to the thermal design of buildings. *Building and Environment, 25*(2), 117-124.

British Research Establishment. (2003). *BREEAM for offices: Assessment predication checklist.* Retrieved June 2004, from http://www.bre.co.uk/pdf/BREEAMchecklist.pdf

British Research Establishment. (2004). *Envest 2*, Retrieved June 2004, from: http://envestv2.bre.co.uk/

Caldas, L.G., and Norford, L.K. (2002). A design optimization tool based on a genetic algorithm. *Automation in Construction, 11*(2), 173-184.

Cantu-Paz, E. (2001). Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics, 7*(4), 311–334

Carnejo, P., and Hittle, D. (1989). An expert system for the design of heating, ventilating, and air-conditioning systems. *ASHRAE Transactions, 95*(1), 379-386.

Chambers, N., Simmons, C., and Wackernagel, M. (2000). Sharing nature's interest: Ecological footprints as an indicator of sustainability. London, UK: Earthscan.

Chan, M.Y., Burnett, J., and Chow, W.K. (1998). Optimum overhang dimensions for energy saving. *Building and Environment, 33*(5), 303-314.

Chris, S., Keoleian, G.A., and Reppe, P. (2003). Life cycle energy and environmental performance of a new university building: Modeling challenges and design implications. *Energy and Buildings, 35*(10), 1049-1064.

Coello, C.A.C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems, 1*(3), 269-308.

Cofaigh, E.O., Fitzgerald, E., Alcock, R., McNicholl, A., Peltonen, V., and Marucco, A. (1999). *A green Vitruvius: Principles and practice of sustainable architecture design*. London, UK: James & James (Science Publishers) Ltd.

Cole, R.J. (1999). Building environmental assessment methods: Clarifying intensions. *Building Research & Information, 27*(4/5), 230-246.

Cole, R.J., and Kernan, P.C. (1996). Life-cycle energy use in office buildings. *Building and Environment, 31*(4), 307-317.

Cole, R.J., and Larsson, N. (2002). *GBTool user manual*, Green Building Challenge 2002.

Coley, D.A., and Schukat, S. (2002). Low-energy design: combining computer-based optimization and human judgment. *Building and Environment, 37*(12), 1241-1247.

Cornelissen R.L. (1997). *Thermodynamics and sustainable development: The use of exergy analysis and the reduction of irreversibility*. Ph.D. Thesis. Laboratory of Thermal Engineering, University of Twente, Enschede, The Netherlands.

Cornelissen, R.L., and Hirs, G.G. (2002). The value of exergetic life cycle assessment besides the LCA. *Energy Conversion and Management, 43*(9-12), 1417-1424.

Dasgupta, D., and McGregor, D.R. (1993). *sGA: A structured genetic algorithm* [Technical Report], Report no. IKBS-11-93. Department of Computer Science, University of Strathclyde, UK.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms.

Computer Methods in Applied Mechanics and Engineering, 186(2-4), 311-338.

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. Chichester, UK: John Wiley & Sons.

Dewulf, J., Langenhove, H. V., and Dirckx, J. (2001). Exergy analysis in the assessment of the sustainability of waste gas treatment systems. *Science of the Total Environment, 273*(1-3), 41-52.

Dewulf, J., Langenhove, V., Mulder, J., Berg, M.M.D., Kooi, H. J., and Arons, J., (2000). Illustrations towards quantifying the sustainability of technology. *Green Chemistry, 2*(3), 108-114.

Dickie, L., and Howard, N. (2000). Assessing environmental impacts of construction: Industry consensus – BREEAM and UK Ecopoints. *BRE Digest*, No.446.

Dincer, I., and Cengel, Y.A. (2001). Energy, entropy and exergy concepts and their roles in thermal engineering. *Entropy, 3*(3), 116-149.

ECOTEC Research & Consulting Limited. (2001). *Ecological footprinting*. Technical report, Luxembourg: European Parliament.

Editeur Officiel du Quebec. (1992). *Regulation respecting energy conservation in new buildings*. Quebec, Canada.

EE4 CODE Version 1.4. (2000). The Buildings Group, Natural Resources Canada, Ottawa.

El-Khawas, I.N. (1997). *The optimal design of buildings: A life-cycle approach to energy efficiency*. Ph.D. thesis, The Ohio State University, Columbus, Ohio, U.S.A.

Environment Canada. (2003). Canada's Greenhouse Gas Inventory 1990-2001. Retrieved June 2004, from http://www.ec.gc.ca/pdb/ghg/1990_01_report/executive_e.cfm

Eumorfopoulou, E., and Aravantinos, D. (1998). The contribution of a planted roof to the thermal protection of buildings in Greece. *Energy and Buildings, 27*(1), 29-36.

European Commission. (1997). ExternE, externalities of energy: Methodology annexes. Retrieved June 2004, from http://externe.jrc.es/append.pdf

Finnveden, G. (1999). *A critical review of operational valuation/weighting methods for life cycle assessment* [Technical Report], AFR-Report 253. Swedish Environmental Research Institute, Stockholm, Sweden.

Finnveden, G, and Ostlund, P. (1997). Exergies of natural resources in life-cycle assessment and other applications. *Energy, 22*(9), 923-931.

Fonseca, C.M., and Flemming, P.J (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms part 1: A unified formulation. *IEEE Transactions on Systems Man and Cybernetics, Part A, 28*(1), 26-37.

Froehlich, G., Hoover, H.J., Liu, L., and Sorenson, P.G. (1999). Designing object-oriented frameworks. In S. Zamir (ed.), *Handbook of object-oriented technology* (pp. 25-1 - 25-22). Boca Raton, FL: CRC Press.

Fu, M.C. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing, 14*(3), 192-215.

Gagnon, L., Belanger, C., and Uchiyama, Y. (2003). Life cycle assessment of electricity generation options: The status of research in year 2001. *Energy Policy, 30*(14), 1267-1278.

Gamma E., Helm R., Johnson R., and Vlissides, J. (1995). *Design patterns: Element of reusable object-oriented software*. Reading, MA: Addison-Wesley.

Gluch, P., and Baumann, H. (2004). The life cycle costing (LCC) approach: A conceptual discussion of its usefulness for environmental decision-making. *Building and Environment, 39*(5), 571-580.

Goedkoop, M., and Spriensma, R. (2001). *The Eco-indicator 99: A damage oriented method for life cycle impact assessment – Methodology annex* (3rd version). PRé Consultants, Amersfoort, The Netherlands.

Goldberg, D. (1989). Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley.

Gosavi A. (2003). *Simulation-based optimization: parametric optimization techniques and reinforcement learning*. Boston: Kluwer Academic Publishers.

Government of Canada. (2001). *Canada's third national report on climate change: Actions to meet commitments under the United Nations Framework Convention on Climate Change*. Ottawa, Canada.

Green Building Advisor. (2004). Retrieved June 2004, from: http://www.greenbuildingadvisor.com/index.html

Guinee, J.B. (2002). *Handbook on life-cycle assessment: Operational guide on the ISO standards*, Dordrecht, The Netherlands: Kluwer Academic Publishers.

Gupta, C.L. (1970). A systematic approach to optimal thermal sesign. *Building Science,* *5*(3), 165-174.

Gustafsson, S.I. (1998). Mixed integer linear programming and building retrofits. *Energy and Buildings, 28*(2), 191-196.

Haab, T.C., and McConnel, K.E. (2002). Valuing environmental and natural resources: The econometrics of non-market valuation. Cheltenham, UK: Edward Elgar Publishing.

Hauglustaine, J.M., and Azar, S. (2001) Interactive tool aiding to optimise the building envelope during the sketch design. In R. Lamberts, C.O.R. Negarao, and J. Hensen (eds.), *Proceedings of the seventh international IBPSA conference* (pp. 387-394), IBPSA.

Hertwich, E.G., Pease, W.S., and Koshland, C.P. (1997). Evaluating the environmental impact of products and production processes: A comparison of six methods. *Science of the Total Environment, 196*(1), 13-29.

Hien, W.N., Poh, L.K., and Feriadi, H. (2000). The use of performance-based simulation tools for building design and evaluation–A Singapore perspective. *Building and Environment, 35*(8), 709-736.

Hong, T., Chou, S.K., and Bong, T.Y. (2000). Building simulation: An overview of developments and information sources. *Building and Environment, 35*(4), 347-361.

HOT2000 Version 8.5. (2001). The Buildings Group, Natural Resources Canada, Ottawa.

Huang, C.F. (2002). *A study of mate selection in genetic algorithms*. Ph.D. Thesis, Department of Electrical Engineering, The University of Michigan, Ann Arbor, MI, U.S.A.

Hwang, C.L., and Masud, A.S.M. (1979). *Multiple objective decision making: Methods and applications*. Berlin, Germany: Springer-Verlag.

Hydro-Quebec. (2004). *Electricity Tariff*, effective January 1, 2004.

Iliescu, S. (2000). *A case-based reasoning approach to the designing of building envelopes*. Ph.D. thesis, Department of Building, Civil, and Environmental Engineering, Concordia University, Montreal, Canada.

ISO. (1997). *ISO 14040: 1997 Environmental management—Life cycle assessment—Principles and framework*. International Organization for Standardization.

IVAM. (2004). *Eco-Quantum*. Retrieved June 2004, from:

http://www.ivambv.uva.nl/uk/index.htm

Johnson, C.A., Besant, R.W., and Schoenau, G.J. (1990). Economic preferred window orientation and optimum fenestration design of a non-daylit and a daylit large office building for different climatic conditions and different billing structures. *ASHRAE Transactions, 96*(1), 23-33.

Kats, G., Alevantis, L., Berman, A., Mills, E., and Perlman, J. (2003). The costs and benefits of green buildings. Retrieved June 2004, from http://www.ciwmb.ca.gov/greenbuilding/Design/CostBenefit/Report.pdf

Khajehpour, S. (2001). *Optimal conceptual design of high-rise office buildings*. Ph.D. thesis, Department of Civil Engineering, University of Waterloo, Ontario, Canada.

Knowles, J.D., and Corne, D.W. (2000). Approximating the nondominated front using the Pareto archived evolutionary strategy. *Evolutionary Computation, 8*(2), 149-172.

Kolda, T.G., Lewis, R.M., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review, 45*(3), 385-484.

Kolodner, J. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.

Lechner, N. (2001). *Heating, cooling, lighting: Design methods for architects* (2nd ed.). New York: John Wiley & Sons.

Le Riche R., Gaudin, J., and Besson, J. (2003). An object-oriented simulation-optimization interface. *Computers and Structures, 81*(17), 1689-1701.

Lindeijer, E. (1996). Normalization and valuation. In H.A. Udo de Haes (Ed.), *Towards a methodology for life-cycle impact assessment* (pp. 75-93). Brussels, Belgium: SETAC-Europe.

Lippiatt, B.C. (2002). *BEES 3.0 building for environmental and economic sustainability* [User Manual]. National Institute of Standards and Technology, U.S.A

Liu, K. (2004). Measured temperatures on the green roof system, [personal communication], Institute for Research in Construction, National Research Council of Canada.

Liu, K., and Baskaran, B. (2003). Thermal performance of green roofs through field evaluation [Technical Report], NRCC-46412. National Research Council of Canada, Ottawa. Retrieved June 2004, from: http://irc.nrc-cnrc.gc.ca/fulltext/nrcc46412/nrcc46412.pdf

Lu, L., Cai, W., Xie, L., Li, S., and Soh, Y.C. (2005). HVAC system optimization–in building section. *Energy and Buildings*, 37(1), 11-22.

Mahdavi, A., and Mahattanatawe, P. (2003). Enclosure systems design and control support via dynamic simulation-assisted optimization. In G. Augenbroe and J. Hensen (eds.), *Proceedings of the eighth international IBPSA conference* (pp. 785-792), IBPSA.

Matthews, H.S., and Lave, L. (2000). Applications of environmental valuation for determining externality costs. *Environmental Science and Technology, 34*(8), 1390-1395.

Mattsson M. (1996). *Object-oriented frameworks: A survey of methodological issues.* Licentiate Thesis, Department of Computer Science, Lund University, Sweden.

Mayer, R., Degelman, L.O., Su, C.J., Keen, A., Griffith, P., Huang, J., Brown, D., and Kim, Y.S. (1991). A knowledge-aided design system for energy-efficient buildings. *ASHRAE Transactions, 97*(2), 479-494.

Menke, D.M., Davis, G.A., and Vigon, B.W. (1996). *Evaluation of life-cycle assessment tools* [Technical Report]. Center for Clean Products and Clean Technologies, The University of Tennessee, U.S.A.

Miettinen, K.M. (1999). *Nonlinear multiobjective optimization.* Boston: Kluwer Academic Publishers.

Miller, B.E. (1992). *Optimizing building design variables.* Master thesis, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO, U.S.A.

Mitrovich, G. (2004). Cost for aluminium overhang by Ametco Manufacturing Corporation, [personal communication].

Moran, M.J. (1982). *Availability analysis: A guide to efficient energy use.* Englewood Cliffs, NJ: Prentice-Hall.

Moran, M.J., and Shapiro, H.N. (1994). *Fundamentals of engineering thermodynamics* (4th ed.). New York: John Wiley & Sons.

Nassif, N., Kajl, S., and Sabourin, R. (2003). Two-objective online optimization of supervisory control strategy. In G. Augenbroe and J. Hensen (eds.), *Proceedings of the eighth international IBPSA conference* (pp. 927-934), IBPSA.

National Research Council. (1997). *Model national energy code of Canada for buildings.* Ottawa, Canada.

Natural Resources Canada, (2004). *Energy efficiency trends in Canada, 1990 to 2002* (Cat. No. M141-1/2002). Ottawa, Canada.

Nielsen, TR. (2002). *Optimization of buildings with respect to energy and indoor environment*. Ph.D. thesis. Department of Civil Engineering, Technical University of Denmark, Lgngby, Denmark.

Niachou, A., Papakonstantinou, K., Santamouris, M., Tsangrassoulis, A., and Mihalakakou G. (2001). Analysis of the green roof thermal properties and investigation of its energy performance. *Energy and Buildings, 33*(7), 719–29.

Onmura, S., Matsumoto, M., and Hokoi, S. (2001). Study on evaporative cooling effect of roof lawn gardens. *Energy and Buildings, 33*(7) 653-666.

Osyczka, A., and Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization, 10*(2), 94-99.

Papamichael, K. (1999). Application of information technologies in building design decisions. *Building Research & Information, 27*(1), 20-34.

Pedersen, C.O., Liesen, R.J., Strand, R.K., Fisher, D.E., Dong, L., and Ellis, P.G. (2000). *A toolkit for building load calculations* [User Manual]. Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

Peippo,K., Lund, P.D., and Vartiainen, E. (1999). Multivariate optimization of design trade-offs for solar low energy buildings. *Energy and Building, 29*(2), 189-205.

Peck, S., and Kuhn, M. (2001). *Design guidelines for green roofs* [Technical Report]. Retrieved June 2004, from: http://www.peck.ca/grhcc/index.html

Pennington, D.W., Potting, J., Finnveden, G., Lindeijer, E., Jolliet, O., Rydlberg, T., and Rebitzer, G. (2004). Life cycle assessment Part 2: Current impact assessment practice. *Environment International, 30*(5), 721-739.

Peuportier, B.L.P. (2001). Life cycle assessment applied to the comparative evaluation of single family houses in the French text. *Energy and Buildings, 33*(5), 443-450.

Qun, G. (1999). *A knowledge-based system for energy-efficient building design*. Master thesis, Department of Building, Civil, and Environmental Engineering, Concordia University, Montreal, Canada.

Quatrani, T. (2003). *Visual modeling with Rational Rose 2002 and UML*. Boston: Addison-Wesley.

Radford, A.D., and Gero, J.S. (1987). *Design by optimization in architecture, building and construction*. New York: Van Nostrand Reinhold Company.

Rafiq M.Y, Mathews J.D, Bullock G.N. (2003). Conceptual building design– Evolutionary approach. *Journal of Computing in Civil Engineering, 17*(3),150-158.

Rao, S.S. (1996). *Engineering optimization: Theory and practice* (3rd ed.). New York: John Wiley & Sons.

Reijnders, L. (2003). Policies influencing cleaner production: The role of prices and regulation. *Journal of Cleaner Production, 11*(3), 333-338.

RMIT University. (2004). *Greening the building life cycle: Life cycle assessment tools in building and construction*. Retrieved June 2004, from: http://buildlca.rmit.edu.au/links.html

Robin, C., Brau, J., and Roux, J.J. (1993). Integration of expert knowledge and simulation tools for the thermal design of buildings and energy systems. *Energy and Buildings, 20*(2), 167-175.

Rosen, M.A., and Dincer, I. (1999). Exergy analysis of waste emissions. *International Journal of Energy research, 23*(13), 1153-1163.

RS Means (2004a). *Assemblies cost data*. Kingston, MA, U.S.A.

RS Means (2004b). *Building construction cost data*. Kingston, MA, U.S.A.

Rudbeck, C., Nielsen, T.R., and Svendsen, S. (2001). Optimal design of building envelopes. In *Conference proceedings of performance of exterior envelopes of whole buildings VIII*: integration of building envelopes, Florida, U.S.A.

Ruegg, R.T., and Marshall, H.E. (1990). *Building economics : Theory and practice*. New York: Van Nostrand Reinhold.

Saaty, T.L. (1990). *Decision making for leaders: The analytical hierarchy process for decisions in a complex world* (2nd ed.). Pittsburgh, PA: University of Pittsburgh.

Saporito, A., Day,A.R., Karayiannis, T.G., and Parand, F. (2001). Multi-parameter building thermal analysis using the lattice method for global optimization. *Energy and Building, 33*(3), 267-274.

Seppo, J., and Arpad, H. (2003). Life-cycle environmental effects of an office building. *Journal of Infrastructure Systems, 9*(4), 157-166.

Shams, H., Nelson, R.M., and Maxwell, G.M. (1994). Development of a knowledge-based system for the selection of HVAC system types for small buildings. *ASHRAE Transactions, 100*(1), 211-218.

Shaviv, E., Yezioro, A., Capeluto, I.G., Pelerg, U.J., and Kalay, Y.E. (1996). Simulations and knowledge-based computer-aided architectural design (CAAD) systems for passive and low energy architecture. *Energy and Buildings, 23*(3), 257-269.

Spears, W.M. (1997). Recombination parameters. In T. B¨ack, D. B. Fogel, and Z. Michalewicz (Eds.), *The handbook of evolutionary computation* (pp. E1.3:1-1.3:13). Philadelphia, PA: IOP Publishing Ltd. and Oxford University Press.

Srinivas, N., and Deb, K. (1994). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation, 2*(3), 221-248.

Steen, B. (1999). A systematic approach to environmental priority strategies in product development (EPS), version 2000 – Models and data for the default method [Research Report]. Centre for Environmental Assessment of Products and Material Systems, Chalmers University of Technology, Göteborg, Sweden.

Steven Winter Associates (1998). *Passive solar design and construction handbook.* New York: John Wiley & Sons.

Stiller, H. (1999). Material intensity of advanced composite materials. *Wuppertal Papers*, No.90. Wuppertal Institute for Climate, Environment and Energy, Wuppertal, Germany.

Szargut, J. Morris, D.R., and Steward, F.R. (1988). *Exergy analysis of thermal, chemical, and metallurgical process.* New York: Hemisphere Publishing.

Tan, K.C., Lee, T.H., and Khor, E.F. (2002). Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons. *Artificial Intelligence Review, 17*(4), 251-290.

Udo de Haes, H.A. (1996). Discussion of general principles and guidelines for practical use. In H.A. Udo de Haes (Ed.), *Towards a methodology for life-cycle impact assessment* (pp. 7-30). Brussels, Belgium: SETAC-Europe.

U.S. Department of Energy. (1999). High-performance commercial buildings: A technology roadmap. Retrieved June 2004, from: http://www.eere.energy.gov/buildings/info/documents/pdfs/roadmap_lowres.pdf

U.S. Department of Energy. (2003). Input output reference: The encyclopedic reference to energyplus input and output. Retrieved June 2004, from: http://www.eere.energy.gov/buildings/energyplus/documentation.html

U.S. Department of Energy. (2004a). 2004 *Building energy databook.* Retrieved June 2004, from http://buildingsdatabook.eren.doe.gov

U.S. Department of Energy. (2004b). *Building energy software tools directory*. Retrieved June 2004, from http://www.eere.energy.gov/buildings/tools_directory

U.S. Green Building Council. (2003). Green building rating system for new construction & major renovations (LEED-NC), Version 2.1.

U.S. National Renewable Energy laboratory. (2004). *Energy-10 Software Homepage.* Retrieved June 2004, from: http://www.nrel.gov/buildings/energy10/

Veldhuizen, D.A.V., and Lamont, G.B. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation, 8*(2), 125-147.

Wall, G. (1977). *Exergy: A useful concept within resource accounting* [Technical Report], Chalmers University of Technology, Göteborg, Sweden.

Watson, R.T., and the Core Writing Team. (Eds.). (2001). *Climate Change 2001: Synthesis Report.* Cambridge, UK: Cambridge University Press.

Wetter, M. (2001). GenOpt—A generic optimization program. In R. Lamberts, C.O.R. Negarao, and J. Hensen (eds.), *Proceedings of the seventh international IBPSA conference* (pp. 601-608), IBPSA.

Wetter M. (2004). *GenOpt—A generic optimization program*, Version 2.0.0. [User Manual]. Simulation research group, Lawrence Berkeley National Laboratory, Berkley, CA, U.S.A.

Wetter, M., and Wright, J.A. (2004). A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment, 39* (8), 989-999.

Wilson, A.J., and Templeman, A.B. (1976). An approach to the optimum thermal design of office buildings. *Building and Environment, 11*(1), 39-50.

Woolley, T., Kimmins, S., Harrison, P., and Harrison, R. (1997). *Green building handbook.* London, UK: E & FN Spon.

Wright, J.A., Loosemore, H.A., and Farmani, R. (2002). Optimization of building thermal design and control by multi-criterion genetic algorithm. *Energy and Buildings, 34*(9), 959-972.

Yohanis,Y.G., and Norton, B. (2000). Early design model for prediction of energy and cost performance of building design options. *International Journal of Solar Energy,* *20*(3), 207-226.

Yohanis, Y.G., and Norton, B. (2002). Life-cycle operational and embodied energy for a generic single-story office building in the UK. *Energy, 27*(1), 77-92.

Zhang, M. (1995). *Analysis of energy conversion systems, including material and global warming aspects*. Ph.D. Thesis. Department of Mechanical Engineering, Oregon State University, Corvallis, OR, U.S.A.

Zitzler E, Laumanns, M., and Thiele, L. (2001). *SPEA2: Improving the strengthen Pareto evolutionary algorithm* [Technical Report], TIK Report 103. Department of Electrical Engineering, Swiss Federal Institute of Technology, Zurich, Swiss.

Zitzler, E., and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 2*(4), 257-271.

Appendix A

Simulation Data Files

A.1 Files for Envelope Layers

These files can be read with reference to Figure 4.3.

File *cladding.txt*

Standard Modular Brick Veneer, C1, 0.75, 0.93, 3, 15759, 13585, 0, 2676, 0, 2539, 0, 8095, 940, 4842, 289, 211, 29, 23, 14, 0, 0, 2165, 7, 0, 0, 2102, 0, 3.8, 0, 21.8, 7.9,;

Concrete Brick Veneer, C2, 0.75, 0.93, 3, 14510, 2630, 0, 4420, 0, 14928, 0, 7278, 1062, 1523, 289, 1967, 519, 71, 117, 139, 4175, 11283, 7, 0, 0, 2337, 0, 2.5, 0, 14.6, 6.4,;

Split-Faced Concrete Block, C3, 0.65, 0.63, 2, 10624, 2061, 0, 7736, 0, 36839, 0, 23787, 5821, 3980, 6742, 1750, 242, 35, 117, 0, 11915, 34542, 7, 0, 0, 7745, 0, 7.0, 0, 49.4, 29.6,;

3/4"*10" Wood Bevel Siding Cedar, C4, 0.65, 0.9, 5, 6717, 492, 13, 324, 5, 454, 348, 114, 16, 84, 277, 3, 0, 2, 0, 0, 0, 0, 1, 488, 0, 33, 35, 0.1, 0, 0.3, 0.2,;

Vinyl Cladding, C5, 0.65, 0.9 , 5, 2551, 526, 0, 282, 0, 4195, 0, 2180, 1246, 876, 4319, 3, 0, 4, 0, 0, 0, 0, 3, 0, 0, 714, 0, 2.0, 0, 5.6, 2.9,;

26 ga Commercial Steel Siding, C6, 0.65, 0.25, 5, 4327, 666, 0, 1340, 3, 5773, 0, 29596, 634, 5763, 281, 7, 0, 3, 0, 0, 0, 0, 1, 0, 0, 4491, 0, 4.9, 0, 24.0, 8.9,;

Stucco over Metal Mesh, C7, 0.65, 0.9, 3, 5597, 393, 0, 1521, 0, 2672, 0, 1490, 181, 643, 1671, 471, 126, 129, 27, 0, 134, 2020, 16, 0, 0, 708, 0, 0.5, 0, 2.9, 2.9,;

File finish.txt

1/2" Firerated Gypsum Board on Wall, F1, 0.65, 0.9, 5, 678, 277, 0, 389, 0, 4099, 0, 1130, 857, 460, 13, 0, 48, 1, 0, 840, 0, 0, 2, 0, 0, 256, 0, 0.5, 0, 3.4, 0.8,;

5/8" Firerated Gypsum Board on Wall, F2, 0.65, 0.9, 5, 710, 321, 0, 487, 0, 5014, 0, 1390, 1080, 560, 13, 0, 48, 1, 0, 1117, 0, 0, 2, 0, 0, 314, 0, 0.7, 0, 4.2, 1.0,;

1/2" Regular Gypsum Board on Wall, F3, 0.65, 0.9, 5, 667, 280, 0, 383, 0, 4105, 0, 1130, 852, 461, 13, 0, 48, 1, 0, 844, 0, 0, 2, 0, 0, 256, 0, 0.5, 0, 3.4, 0.8,;

5/8" Regular Gypsum Board on Wall, F4, 0.65, 0.9, 5, 700, 323, 0, 480, 0, 5011, 0, 1388, 1074, 560, 13, 0, 48, 1, 0, 1102, 0, 0, 2, 0, 0, 314, 0, 0.7, 0, 4.2, 1.0,;

1/2" Moisture Resistant Gypsum Board on Wall, F5, 0.65, 0.9, 5, 721, 299, 0, 426, 0, 4307, 0, 1189, 903, 487, 13, 0, 48, 1, 0, 928, 0, 0, 2, 0, 0, 269, 0, 0.6, 0, 3.6, 0.8,;

5/8" Moisture Resistant Gypsum Board on Wall, F6, 0.65, 0.9, 5, 764, 346, 0, 528, 0, 5242, 0, 1455, 1129, 590, 13, 0, 48, 1, 0, 1186, 0, 0, 2, 0, 0, 329, 0, 0.7, 0, 4.4, 1.0,;

1/2" Firerated Gypsum Board on Roof, F7, 0.65, 0.9, 5, 732, 277, 0, 389, 0, 4099, 0, 1130, 857, 460, 13, 0, 48, 1, 0, 840, 0, 0, 2, 0, 0, 256, 0, 0.5, 0, 3.4, 0.8,;

5/8" Firerated Gypsum Board on Roof, F8, 0.65, 0.9, 5, 764, 321, 0, 487, 0, 5014, 0, 1390, 1080, 560, 13, 0, 48, 1, 0, 1117, 0, 0, 2, 0, 0, 314, 0, 0.7, 0, 4.2, 1.0,;

1/2" Regular Gypsum Board on Roof, F9, 0.65, 0.9, 5, 721, 280, 0, 383, 0, 4105, 0, 1130, 852, 461, 13, 0, 48, 1, 0, 844, 0, 0, 2, 0, 0, 256, 0, 0.5, 0, 3.4, 0.8,;

5/8" Regular Gypsum Board on Roof, F10, 0.65, 0.9, 5, 754, 323, 0, 480, 0, 5011, 0, 1388, 1074, 560, 13, 0, 48, 1, 0, 1102, 0, 0, 2, 0, 0, 314, 0, 0.7, 0, 4.2, 1.0,;

1/2" Moisture Resistant Gypsum Board on Roof, F11, 0.65, 0.9, 5, 775, 299, 0, 426, 0, 4307, 0, 1189, 903, 487, 13, 0, 48, 1, 0, 928, 0, 0, 2, 0, 0, 269, 0, 0.6, 0, 3.6, 0.8,;

5/8" Moisture Resistant Gypsum Board on Roof, F12, 0.65, 0.9, 5, 818, 346, 0, 528, 0, 5242, 0, 1455, 1129, 590, 13, 0, 48, 1, 0, 1186, 0, 0, 2, 0, 0, 329, 0, 0.7, 0, 4.4, 1.0,;

5/8" Mineral Fiber Accoustical Tile on ceiling, F13, 0.65, 0.9, 5, 1216, 4, 0, 901, 0, 2293, 0, 4867, 129, 422, 0, 194, 0, 0, 0, 0, 0, 0, 0, 0, 647, 0, 0.9, 0, 4.8, 14.1,;

3/4" Mineral Fiber Accoustical Tile on ceiling, F14, 0.65, 0.9, 5, 2131, 5, 0, 1097, 0, 2792, 0, 5926, 157, 514, 0, 236, 0, 0, 0, 0, 0, 0, 0, 0, 787, 0, 1.1, 0, 5.9, 17.1,;

<u>File *insulation.txt*</u>

1" Expanded Polystyrene on Wall, I1, 0.65, 0.9, 4, 743, 81, 0, 29, 0, 241, 0, 355, 872, 121, 2774, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 187, 0, 0.2, 0, 1.1, 0.7,;

2" Expanded Polystyrene on Wall, I2, 0.65, 0.9, 4, 1076, 140, 0, 55, 0, 427, 0, 698, 1741, 237, 5473, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 368, 0, 0.4, 0, 2. 2, 1.3,;

3" Expanded Polystyrene on Wall, I3, 0.65, 0.9, 4, 1216, 198, 0, 79, 0, 612, 0, 1042, 2610, 353, 8171, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 549, 0, 0.6, 0, 3.3, 1.9,;

4" Expanded Polystyrene on Wall, I4, 0.65, 0.9, 4, 1959, 279, 0, 108, 0, 853, 0, 1397, 3482, 474, 10945, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 736, 0, 0.8, 0, 4.4, 2.6,;

5" Expanded Polystyrene on Wall, I5, 0.65, 0.9, 4, 2293, 338, 0, 134, 0, 1039, 0, 1740, 4351, 590, 13644, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 917, 0, 1.0, 0, 5.5, 3.2,;

6" Expanded Polystyrene on Wall, I6, 0.65, 0.9, 4, 2433, 396, 0, 158, 0, 1224, 0, 2084, 5220, 706, 16342, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1098, 0, 1.2, 0, 6.5, 3.9,;

7" Expanded Polystyrene on Wall, I7, 0.65, 0.9, 4, 3369, 454, 0, 184, 0, 1410, 0, 2428, 6089, 822, 19041, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1279, 0, 1.4, 0, 7.6, 4.5,;

1" Extruded Polystyrene on Wall, I8, 0.65, 0.9, 4, 980, 140, 0, 19, 0, 428, 0, 701, 1747, 238, 5491, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 369, 0, 0.4, 0, 2.2, 1.3,;

2" Extruded Polystyrene on Wall, I9, 0.65, 0.9, 4, 1442, 257, 0, 34, 0, 801, 0, 1391, 3492, 471, 10908, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 732, 0, 0.8, 0, 4.4, 2.6,;

3" Extruded Polystyrene on Wall, I10, 0.65, 0.9, 4, 1765, 374, 0, 50, 0, 1173, 0, 2080, 5236, 703, 16324, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 1095, 0, 1.2, 0, 6.5, 3.9,;

4" Extruded Polystyrene on Wall, I11, 0.65, 0.9, 4, 2745, 514, 0, 69, 0, 1601, 0, 2781, 6983, 941, 21815, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1464, 0, 1.6, 0, 8.7, 5.2,;

5" Extruded Polystyrene on Wall, I12, 0.65, 0.9, 4, 3208, 631, 0, 84, 0, 1974, 0, 3471, 8728, 1174, 27232, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1827, 0, 2.0, 0, 11.0, 6.5,;

6" Extruded Polystyrene on Wall, I13, 0.65, 0.9, 4, 3531, 748, 0, 100, 0, 2346, 0, 4160, 10472, 1406, 32648, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 2190, 0, 2.4, 0, 13.1, 7.8,;

7" Extruded Polystyrene on Wall, I14, 0.65, 0.9, 4, 4650, 866, 0, 115, 0, 2719, 0, 4850, 12216, 1639, 38064, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 2553, 0, 2.8, 0, 15.3, 9.1,;

1" Expanded Polystyrene on Roof, I15, 0.65, 0.9, 4, 506, 81, 0, 29, 0, 241, 0, 355, 872, 121, 2774, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 187, 0, 0.2, 0, 1.1, 0.7,;

2" Expanded Polystyrene on Roof, I16, 0.65, 0.9, 4, 775, 140, 0, 55, 0, 427, 0, 698, 1741, 237, 5473, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 368, 0, 0.4, 0, 2.2, 1.3,;

3" Expanded Polystyrene on Roof, I17, 0.65, 0.9, 4, 1033, 198, 0, 79, 0, 612, 0, 1042, 2610, 353, 8171, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 549, 0, 0.6, 0, 3.3, 1.9,;

4" Expanded Polystyrene on Roof, I18, 0.65, 0.9, 4, 1023, 279, 0, 108, 0, 853, 0, 1397, 3482, 474, 10945, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 736, 0, 0.8, 0, 4.4, 2.6,;

5" Expanded Polystyrene on Roof, I19, 0.65, 0.9, 4, 1195, 338, 0, 134, 0, 1039, 0, 1740, 4351, 590, 13644, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 917, 0, 1.0, 0, 5.5, 3.2,;

6" Expanded Polystyrene on Roof, I20, 0.65, 0.9, 4, 1335, 396, 0, 158, 0, 1224, 0, 2084, 5220, 706, 16342, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1098, 0, 1.2, 0, 6.5, 3.9,;

7" Expanded Polystyrene on Roof, I21, 0.65, 0.9, 4, 1841, 454, 0, 184, 0, 1410, 0, 2428, 6089, 822, 19041, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1279, 0, 1.4, 0, 7.6, 4.5,;

8" Expanded Polystyrene on Roof, I22, 0.65, 0.9, 4, 2110, 536, 0, 213, 0, 1651, 0, 2782, 6961, 943, 21815, 0, 0, 12, 0, 0, 0, 0, 12, 0, 0, 1466, 0, 1.6, 0, 8.7, 5.2,;

9" Expanded Polystyrene on Roof, I23, 0.65, 0.9, 4, 2368, 594, 0, 239, 0, 1837, 0, 3126, 7830, 1059, 24514, 0, 0, 12, 0, 0, 0, 0, 12, 0, 0, 1647, 0, 1.8, 0, 9.8, 5.8,;

1" Extruded Polystyrene on Roof, I24, 0.65, 0.9, 4, 775, 140, 0, 19, 0, 428, 0, 701, 1747, 238, 5491, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 369, 0, 0.4, 0, 2.2, 1.3,;

2" Extruded Polystyrene on Roof, I25, 0.65, 0.9, 4, 1303, 257, 0, 34, 0, 801, 0, 1391, 3492, 471, 10908, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 732, 0, 0.8, 0, 4.4, 2.6,;

3" Extruded Polystyrene on Roof, I26, 0.65, 0.9, 4, 1873, 374, 0, 50, 0, 1173, 0, 2080, 5236, 703, 16324, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 1095, 0, 1.2, 0, 6.5, 3.9,;

4" Extruded Polystyrene on Roof, I27, 0.65, 0.9, 4, 1819, 514, 0, 69, 0, 1601, 0, 2781, 6983, 941, 21815, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1464, 0, 1.6, 0, 8.7, 5.2,;

5" Extruded Polystyrene on Roof, I28, 0.65, 0.9, 4, 2594, 631, 0, 84, 0, 1974, 0, 3471, 8728, 1174, 27232, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 1827, 0, 2.0, 0, 11.0, 6.5,;

6" Extruded Polystyrene on Roof, I29, 0.65, 0.9, 4, 3122, 748, 0, 100, 0, 2346, 0, 4160, 10472, 1406, 32648, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 2190, 0, 2.4, 0, 13.1, 7.8,;

7" Extruded Polystyrene on Roof, I30, 0.65, 0.9, 4, 3692, 866, 0, 115, 0, 2719, 0, 4850, 12216, 1639, 38064, 0, 0, 8, 0, 0, 0, 0, 8, 0, 0, 2553, 0, 2.8, 0, 15.3, 9.1,;

8" Extruded Polystyrene on Roof, I31, 0.65, 0.9, 4, 4467, 1005, 0, 134, 0, 3147, 0, 5551, 13964, 1877, 43556, 0, 0, 12, 0, 0, 0, 0, 12, 0, 0, 2922, 0, 3.2, 0, 17.5, 10.4,;

9" Extruded Polystyrene on Roof, I32, 0.65, 0.9, 4, 4995, 1123, 0, 149, 0, 3520, 0, 6241, 15708, 2110, 48972, 0, 0, 12, 0, 0, 0, 0, 12, 0, 0, 3285, 0, 3.6, 0, 19.6, 11.7,;

1" Polyisocyanurate on Roof, I33, 0.65, 0.9, 4, 689, 122, 0, 53, 0, 2304, 0, 999, 366, 784, 3071, 5, 0, 4, 0, 0, 0, 0, 4, 0, 0, 623, 1, 0.5, 0, 2.9, 0.6,;

2" Polyisocyanurate on Roof, I34, 0.65, 0.9, 4, 947, 222, 0, 103, 0, 4555, 0, 1991, 740, 1563, 6099, 8, 0, 4, 0, 0, 0, 0, 4, 0, 0, 1242, 1, 1.0, 0, 5.8, 1.2,;

3" Polyisocyanurate on Roof, I35, 0.65, 0.9, 4, 1259, 322, 0, 152, 0, 6805, 0, 2983, 1114, 2342, 9127, 12, 0, 4, 0, 0, 0, 0, 4, 0, 0, 1861, 2, 1.5, 0, 8.7, 1.9,;

4" Polyisocyanurate on Roof, I36, 0.65, 0.9, 4, 1895, 444, 0, 205, 0, 9109, 0, 3982, 1480, 3126, 12198, 17, 0, 8, 0, 0, 0, 0, 8, 0, 0, 2484, 3, 2.0, 0, 11.6, 2.5,;

5" Polyisocyanurate on Roof, I37, 0.65, 0.9, 4, 2207, 544, 0, 255, 0, 11360, 0, 4974, 1854, 3905, 15226, 20, 0, 8, 0, 0, 0, 0, 8, 0, 0, 3103, 3, 2.6, 0, 14.4, 3.1,;

1/2" Mineral Fiberboard, I38, 0.65, 0.9, 4, 635, 2, 0, 510, 0, 1296, 0, 2751, 73, 239, 0, 110, 0, 0, 0, 0, 0, 0, 0, 0, 0, 365, 0, 0.5, 0, 2.7, 8.0,;

File *membrane.txt*

Single-Ply EPDM, M1, 0.65, 0.9, 4, 1060, 311, 0, 1108, 4, 6455, 113, 5845, 4367, 2027, 13098, 12, 16, 51, 0, 0, 0, 0, 9, 117, 0, 1190, 11, 1.7, 0, 14.9, 3.7,;

PVC Roofing Membrane, M2, 0.65,0.9, 5, 1518, 425, 0, 1166, 4, 15353, 113, 8892, 1131, 2967, 17826, 16, 0, 41, 0, 0, 0, 0, 9, 117, 0, 1339, 11, 4.8, 0, 25.0, 4.5,;

2-Ply Modified Bitumen Membrane, M3, 0.65, 0.9, 3, 2067, 2557, 5, 477, 5, 18533, 0, 15893, 34655, 5469, 72129, 175, 0, 72, 0, 0, 0, 0, 37, 0, 0, 3703, 0, 2.5, 0, 36.8, 8.1,;

4-Ply Buit-Up with Organic Felt, M4, 0.65, 0.9, 3, 2325, 1619, 10, 583, 10, 15667, 0, 16343, 36951, 5809, 50009, 19, 0, 166, 0, 0, 0, 0, 52, 0, 0, 3910, 0, 1.6, 0, 33.7, 8.2,;

Asphalt Sheathing Paper, M5, 0.65, 0.9, 4, 151, 123, 3, 55, 3, 792, 0, 202, 144, 75, 2370, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 47, 0, 0.1, 0, 0.7, 0.2,;

6 mil Polyethylene Vapor Barrier, M6, 0.65, 0.9, 6, 152, 13, 0, 18, 0, 471, 0, 248, 209, 133, 724, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 80, 0, 0.1, 0, 0.6, 0.3,;

Modified Bitumen Membrane, M7, 0.65, 0.9, 3, 1033, 1278, 2.5, 239, 2.5, 9267, 0, 7947, 17328, 2735, 36065, 88, 0, 36, 0, 0, 0, 0, 18, 0, 0, 1852, 0, 1.2, 0, 18.4, 4.0,;

File *other.txt*

Exterior Wall Air Space 25mm, OT1, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Ceiling Air Space 400mm, OT2, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Glazing Air Space 6 mm without Low-E, OT3, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Glazing Air Space 6 mm with Low-E, OT4, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Glazing Air Space 13mm without Low-E, OT5, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Glazing Air Space 13mm with Low-E, OT6, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,;

Green Roof, Grass, 0.6, 0.6, 2, 10400, 122, 0, 44, 0, 362, 0, 533, 1308, 182, 4161, 0, 0, 6, 0, 0, 0, 0, 6, 0, 0, 281, 0, 0.3, 0, 1.7, 1.0,;

File *sheathing.txt*

3/8" Plywood Sheathing on Wall, SH1, 0.65, 0.9, 5, 1238, 160, 0, 963, 17, 1822, 786, 346, 22, 112, 316, 0, 0, 0, 0, 0, 0, 0, 0, 472, 6, 100, 79, 0.2, 0, 1.3, 0.6,;

1/2" Plywood Sheathing on Wall, SH2, 0.65, 0.9, 5, 1249, 213, 0, 1284, 23, 2429, 1048, 461, 29, 149, 421, 0, 0, 0, 0, 0, 0, 0, 0, 629, 8, 133, 105, 0.3, 0, 1.8, 0.8,;

5/8" Plywood Sheathing on Wall, SH3, 0.65, 0.9, 5, 1378, 267, 0, 1605, 28, 3037, 1310, 577, 37, 187, 527, 0, 0, 0, 0, 0, 0, 0, 0, 787, 10, 167, 132, 0.4, 0, 2.2, 1.0,;

3/4" Plywood Sheathing on Wall, SH4, 0.65, 0.9, 5, 1582, 320, 0, 1926, 34, 3644, 1572, 692, 44, 224, 632, 0, 0, 0, 0, 0, 0, 0, 0, 944, 12, 200, 158, 0.5, 0, 2.6, 1.3,;

3/8" OSB Sheathing on Wall, SH5, 0.65, 0.9, 5, 1076, 654, 0, 954, 72, 533, 1799, 106, 6, 81, 982, 0, 0, 0, 0, 0, 0, 0, 0, 472, 12, 72, 180, 0.1, 0, 0.5, 1.1,;

1/2" OSB Sheathing on Wall, SH6, 0.65, 0.9, 5, 1184, 872, 0, 1272, 96, 711, 2399, 141, 8, 108, 1309, 0, 0, 0, 0, 0, 0, 0, 0, 629, 16, 96, 240, 0.1, 0, 0.6, 1.4,;

5/8" OSB Sheathing on Wall, SH7, 0.65, 0.9, 5, 1303, 1090, 0, 1590, 120, 888, 2998, 177, 10, 135, 1637, 0, 0, 0, 0, 0, 0, 0, 0, 787, 20, 120, 300, 0.1, 0, 0.8, 1.8,;

3/4" OSB Sheathing on Wall, SH8, 0.65, 0.9, 5, 1464, 1308, 0, 1908, 144, 1066, 3598, 212, 12, 162, 1964, 0, 0, 0, 0, 0, 0, 0, 0, 944, 24, 144, 360, 0.1, 0, 0.9, 2.1,;

1/2" Gypsum Sheathing on Wall, SH9, 0.65, 0.9, 5, 1120, 299, 0, 426, 0, 4307, 0, 1189, 903, 487, 13, 0, 48, 1, 0, 928, 0, 0, 2, 0, 0, 269, 0, 0.6, 0, 3.6, 0.8,;

3/8" Plywood Sheathing on Roof, SH10, 0.65, 0.9, 5, 1098, 160, 0, 963, 17, 1822, 786, 346, 22, 112, 316, 0, 0, 0, 0, 0, 0, 0, 0, 472, 6, 100, 79, 0.2, 0, 1.3, 0.6,;

1/2" Plywood Sheathing on Roof, SH11, 0.65, 0.9, 5, 1098, 213, 0, 1284, 23, 2429, 1048, 461, 29, 149, 421, 0, 0, 0, 0, 0, 0, 0, 0, 629, 8, 133, 105, 0.3, 0, 1.8, 0.8,;

5/8" Plywood Sheathing on Roof, SH12, 0.65, 0.9, 5, 1216, 267, 0, 1605, 28, 3037, 1310, 577, 37, 187, 527, 0, 0, 0, 0, 0, 0, 0, 0, 787, 10, 167, 132, 0.4, 0, 2.2, 1.0,;

3/4" Plywood Sheathing on Roof, SH13, 0.65, 0.9, 5, 1421, 320, 0, 1926, 34, 3644, 1572, 692, 44, 224, 632, 0, 0, 0, 0, 0, 0, 0, 0, 944, 12, 200, 158, 0.5, 0, 2.6, 1.3,;

3/8" OSB Sheathing on Roof, SH14, 0.65, 0.9, 5, 990, 654, 0, 954, 72, 533, 1799, 106, 6, 81, 982, 0, 0, 0, 0, 0, 0, 0, 0, 472, 12, 72, 180, 0.1, 0, 0.5, 1.1,;

1/2" OSB Sheathing on Roof, SH15, 0.65, 0.9, 5, 1098, 872, 0, 1272, 96, 711, 2399, 141, 8, 108, 1309, 0, 0, 0, 0, 0, 0, 0, 0, 629, 16, 96, 240, 0.1, 0, 0.6, 1.4,;

5/8" OSB Sheathing on Roof, SH16, 0.65, 0.9, 5, 1227, 1090, 0, 1590, 120, 888, 2998, 177, 10, 135, 1637, 0, 0, 0, 0, 0, 0, 0, 0, 787, 20, 120, 300, 0.1, 0, 0.8, 1.8,;

3/4" OSB Sheathing on Roof, SH17, 0.65, 0.9, 5, 1399, 1308, 0, 1908, 144, 1066, 3598, 212, 12, 162, 1964, 0, 0, 0, 0, 0, 0, 0, 0, 944, 24, 144, 360, 0.1, 0, 0.9, 2.1,;

File *structure.txt*

4" Concrete Block Back-Up Wall, ST1, 0.65, 0.9, 3, 5705, 3777, 0, 3489, 0, 14655, 0, 7178, 1519, 1821, 8390, 201, 22, 106, 11, 0, 30, 1657, 800, 0, 0, 2535, 0, 1.8, 0, 13.1, 9.8,;

8" Concrete Block Back-Up Wall, ST2, 0.65, 0.9, 3, 7051, 7554, 0, 6978, 1, 29310, 0, 14356, 3037, 3641, 16779, 402, 43, 211, 21, 0, 60, 3313, 1599, 0, 0, 5068, 0, 3.6, 0, 26.2, 19.6,;

Conventional OWSJ Roof with Gypsum Board, ST3, 0.65, 0.9, 4, 8439, 3102, 0, 2955, 5, 19045, 0, 7295, 2042, 6949, 25982, 203, 48, 1929, 0, 928, 0, 0, 807, 0, 0, 3445, 0, 1.5, 0, 14.4, 6.7,;

Green OWSJ Roof with Gypsum Board, ST4, 0.65, 0.9, 4, 9322, 3887, 0, 3366, 5, 23325, 0, 8783, 2318, 8336, 32820, 252, 48, 2378, 0, 928, 0, 0, 1072, 0, 0, 4186, 0, 1.6, 0, 16.7, 8.0,;

Conventional OWSJ Roof, ST5, 0.65, 0.9, 4, 7664, 2803, 0, 2529, 5, 14738, 0, 6106, 1139, 6462, 25969, 203, 0, 1928, 0, 0, 0, 0, 806, 0, 0, 3176, 0, 0.9, 0, 10.9, 5.9,;

Green OWSJ Roof, ST6, 0.65, 0.9, 4, 8547, 3588, 0, 2940, 5, 19018, 0, 7594, 1415, 7849, 32807, 252, 0, 2377, 0, 0, 0, 0, 1070, 0, 0, 3917, 0, 1.1, 0, 13.1, 7.1,;

Conventional CIP Flat Roof, ST7, 0.65, 0.9, 4, 14335, 6833, 0, 13149, 0, 16569, 0, 15650, 4891, 3082, 13303, 6701, 932, 216, 454, 0, 23321, 21380, 1268, 0, 0, 5873, 0, 2.6, 0, 24.0, 24.6,;

Green CIP Flat Roof, ST8, 0.65, 0.9, 4, 14813, 8006, 0, 14454, 0, 19526, 0, 18113, 5640, 3609, 16019, 7715, 1073, 257, 522, 0, 26838, 24604, 1527, 0, 0, 6754, 0, 3.0, 0, 27.7, 27.9,;

CIP Flat Floor, ST9, 0.65, 0.9, 4, 15125, 8006, 0, 14454, 0, 19526, 0, 18113, 5640, 3609, 16019, 7715, 1073, 257, 522, 0, 26838, 24604, 1527, 0, 0, 6754, 0, 3.0, 0, 27.7, 27.9,;

Conventional OWSJ Floor, ST10, 0.65, 0.9, 4, 12293, 5554, 0, 6170, 5, 24265, 0, 13889, 3389, 9470, 36093, 2923, 374, 2518, 181, 0, 9322, 8548, 1086, 0, 0, 6124, 0, 2.2, 0, 23.0, 15.3,;

Green OWSJ Floor, ST11, 0.65, 0.9, 4, 12293, 5612, 0, 6226, 5, 25008, 0, 14236, 3433, 9866, 37291, 2931, 374, 2573, 181, 0, 9322, 8548, 1320, 0, 0, 6247, 0, 2.2, 0, 23.5, 15.5,;

File *studInsulation.txt*

25 ga 4" wide steel stud 24" O.C. with Fiberglass in cavity, SI1, 0.65, 0.9, 3, 1335, 690, 0, 474, 1, 6182, 0, 1624, 258, 985, 2959, 124, 0, 212, 162, 0, 0, 0, 26, 0, 0, 681, 0, 1.7, 0, 5.6, 2.1,;

25 ga 6" wide steel stud 24" O.C. with Fiberglass in cavity, SI2, 0.65, 0.9, 3, 1625, 931, 0, 693, 1, 9147, 0, 2365, 367, 1388, 3987, 183, 0, 286, 244, 0, 0, 0, 33, 0, 0, 963, 0, 2.5, 0, 8.1, 3.0,;

25 ga 8" wide steel stud 24" O.C. with Fiberglass in cavity, SI3, 0.65, 0.9, 3, 2670, 1380, 0, 948, 2, 12364, 0, 3248, 516, 1970, 5918, 248, 0, 424, 324, 0, 0, 0, 52, 0, 0, 1362, 0, 3.4, 0, 11.1, 4.1,;

25 ga 10" wide steel stud 24" O.C. with Fiberglass in cavity, SI4, 0.65, 0.9, 3, 2960, 1621, 0, 1167, 2, 15329, 0, 3989, 625, 2373, 6946, 307, 0, 498, 406, 0, 0, 0, 59, 0, 0, 1644, 0, 4.2, 0, 13.7, 5.1,;

25 ga 4" wide steel stud 24" O.C. with Rockwool in cavity, SI5, 0.65, 0.9, 3, 1399, 280, 0, 1640, 1, 4592, 0, 8793, 355, 1319, 2959, 356, 0, 212, 0, 0, 0, 0, 26, 0, 0, 1468, 0, 1.7, 0, 9.7, 24.6,;

25 ga 6" wide steel stud 24" O.C. with Rockwool in cavity, SI6, 0.65, 0.9, 3, 1582, 315, 0, 2443, 1, 6761, 0, 13119, 513, 1888, 3987, 530, 0, 286, 0, 0, 0, 0, 33, 0, 0, 2143, 0, 2.5, 0, 14.3, 36.9,;

25 ga 8" wide steel stud 24" O.C. with Rockwool in cavity, SI7, 0.65, 0.9, 3, 2799, 560, 0, 3280, 2, 9184, 0, 17586, 710, 2638, 5918, 712, 0, 424, 0, 0, 0, 0, 52, 0, 0, 2936, 0, 3.3, 0, 19.4, 49.3,;

25 ga 10" wide steel stud 24" O.C. with Rockwool in cavity, SI8, 0.65, 0.9, 3, 2982, 595, 0, 4083,

```
2, 11353, 0, 21912, 868, 3207, 6946, 886, 0, 498, 0, 0, 0, 0, 59, 0, 0, 3611, 0, 4.1, 0, 24.0,
61.5,;
```

A.2 File for Window Types

This file can be read with reference to Figure 4.4.

<u>File *windowType.txt*</u>

```
1/4" Single Pane Standard Glazing, SinglePaneWindow, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.73,0.80,0.17,0,0,10,
  0,10,20,30,40,50,60,70,80,90,
  0.81,0.81,0.81,0.80,0.80,0.78,0.73,0.62,0.39,0,
  0.16,0.16,0.16,0.17,0.17,0.18,0.19,0.19,0.17,0,
  0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,
  0.88,0.88,0.88,0.87,0.87,0.85,0.80,0.69,0.43,0,
  12648, 4285, 0, 1155, 0, 3451, 0, 1435, 278, 590, 0, 403, 0, 0, 1025, 0, 0, 0, 0, 0, 0, 1649, 0,
  2.2, 0, 11.1, 11.1,;

Double Pane Standard Glazing, DoublePaneWindow, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1, 1, 0.60, 0.51, 0.19, 0.11, 0, 10,
  0,10,20,30,40,50,60,70,80,90,
  0.70,0.70,0.69,0.68,0.67,0.64,0.58,0.45,0.23,0,
  0.17,0.17,0.17,0.18,0.18,0.19,0.20,0.21,0.20,0,
  0.11,0.11,0.11,0.12,0.12,0.12,0.12,0.10,0.07,0,
  0,0,0,0,0,0,0,0,0,0,
  0.61,0.61,0.60,0.59,0.58,0.55,0.48,0.36,0.17,0,
  24758, 8569, 0, 2311, 0, 6903, 0, 2869, 556, 1180, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3299, 0,
  4.4, 0, 22.2, 22.3,;

Reflective Double Glazing, ReflectiveDoubleWindow, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1, 1, 0.25, 0.15, 0.61, 0.04, 0, 10,
  0,10,20,30,40,50,60,70,80,90,
  0.29, 0.29, 0.29, 0.28, 0.28, 0.27, 0.25, 0.20, 0.12, 0,
  0.64, 0.64, 0.64, 0.64, 0.64, 0.63, 0.62, 0.58, 0.43, 0,
  0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.03, 0.02, 0,
  0,0,0,0,0,0,0,0,0,0,
  0.18, 0.18, 0.18, 0.17, 0.17, 0.16, 0.14, 0.10, 0.05, 0,
  25834, 8569, 0, 2327, 0, 6920, 0, 2876, 556, 1181, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3301, 0,
  4.4, 0, 22.2, 22.3,;
```

```
Double Low-E 0.2 inside Glazing, DoublePaneLowETin1, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.56,0.43,0.19,0.16,0,10,
  0,10,20,30,40,50,60,70,80,90,
  0.65,0.65,0.64,0.64,0.63,0.60,0.54,0.42,0.21,0,
  0.17,0.17,0.17,0.18,0.19,0.20,0.21,0.22,0.22,0,
  0.17,0.17,0.17,0.17,0.17,0.17,0.15,0.13,0.07,0,
  0,0,0,0,0,0,0,0,0,0,
  0.51,0.51,0.50,0.49,0.48,0.46,0.41,0.30,0.14,0,
  29602, 8569, 0, 2327, 0, 6920, 0, 2876, 556, 1181, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3301, 0,
  4.4, 0, 22.2, 22.3,;

Double Low-E 0.1 inside Glazing, DoublePaneLowETin2, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.49,0.35,0.21,0.16, 0, 10,
  0,10,20,30,40,50,60,70,80,90,
  0.56, 0.56,0.56,0.55,0.55,0.52,0.48,0.38,0.2,0,
  0.19,0.19,0.19,0.2,0.2,0.21,0.22,0.23,0.22,0,
  0.16,0.16,0.16,0.17,0.17,0.17,0.17,0.16,0.1,0,
  0,0,0,0,0,0,0,0,0,0,
  0.42,0.42,0.42,0.41,0.4,0.37,0.32,0.24,0.11,0,
  29602, 8569, 0, 2327, 0, 6920, 0, 2876, 556, 1181, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3301, 0,
  4.4, 0, 22.2, 22.3,;

Double Low-E 0.2 outside Glazing, DoublePaneLowETin3, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.53,0.43,0.25,0.10,0,10,
  0,10,20,30,40,50,60,70,80,90,
  0.6, 0.6, 0.6, 0.6, 0.59, 0.57, 0.51, 0.4, 0.21, 0,
  0.26, 0.26, 0.26, 0.26, 0.26, 0.26, 0.26, 0.25, 0.19, 0,
  0.1, 0.1, 0.1, 0.11, 0.11, 0.11, 0.11, 0.1, 0.07, 0,
  0,0,0,0,0,0,0,0,0,0,
  0.51,0.51,0.50,0.49,0.48,0.46,0.41,0.30,0.14,0,
  29602, 8569, 0, 2327, 0, 6920, 0, 2876, 556, 1181, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3301, 0,
  4.4, 0, 22.2, 22.3,;

Double Low-E 0.1 outside Glazing, DoublePaneLowETin4, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.44,0.35,0.32,0.08, 0, 10,
  0,10,20,30,40,50,60,70,80,90,
  0.51, 0.51,0.51,0.50,0.49,0.47,0.42,0.32,0.17,0,
  0.3,0.3,0.3,0.31,0.32,0.32,0.33,0.35,0.29,0,
  0.08,0.08,0.08,0.09,0.09,0.09,0.09,0.08,0.05,0,
  0,0,0,0,0,0,0,0,0,0,
  0.42,0.42,0.42,0.41,0.4,0.37,0.32,0.24,0.11,0,
  29602, 8569, 0, 2327, 0, 6920, 0, 2876, 556, 1181, 0, 807, 0, 0, 2051, 0, 0, 0, 0, 0, 0, 3301, 0,
  4.4, 0, 22.2, 22.3,;
```

```
Triple Pane Standard Glazing, TriplePaneWindow, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1, 1, 0.51, 0.39, 0.19, 0.12, 0.08, 10,
  0,10,20,30,40,50,60,70,80,90,
  0.61,0.61, 0.60,0.59,0.58,0.55,0.48,0.35,0.16,0,
  0.17,0.17,0.18,0.18,0.19,0.20,0.21,0.22,0.21,0,
  0.12,0.12,0.12,0.13,0.13,0.13,0.13,0.12,0.08,0,
  0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.06,0.03,0,
  0.49,0.48,0.47,0.46,0.45,0.42,0.35,0.24,0.09,0,
  37406, 12854, 0, 3466, 0, 10354, 0, 4304, 834, 1770, 0, 1210, 0, 0, 3076, 0, 0, 0, 0, 0, 0, 4948,
  0, 6.5, 0, 33.2, 33.4,;

Triple Pane Low-E 0.2 inside Glazing, TriplePaneLowETin, 1.2, 0, 0, 0.9, 0.9, 6,
  0.1,1,0.47,0.30,0.19,0.13,0.14,10,
  0,10,20,30,40,50,60,70,80,90,
  0.56,0.56,0.55,0.54,0.53,0.50,0.44,0.32,0.15,0,
  0.17,0.17,0.18,0.18,0.19,0.20,0.21,0.22,0.22,0,
  0.13,0.13,0.13,0.13,0.14,0.14,0.14,0.13,0.10,0,
  0.15,0.15,0.15,0.16,0.16,0.15,0.14,0.12,0.05,0,
  0.39,0.39,0.38,0.37,0.36,0.33,0.27,0.17,0.06,0,
  42250, 12854, 0, 3482, 0, 10371, 0, 4311, 834, 1771, 0, 1210, 0, 0, 3076, 0, 0, 0, 0, 0, 0, 4951,
  0, 6.5, 0, 33.2, 33.4,;
```

A.3 File for Overhang Types

This file can be read with reference to Figure 4.5.

File *overhangType.txt*

```
NULL,;

aluminum Overhang, 0, 0, 0.08, 21528, 31460, 9, 1748, 0, 23021, 0, 9603, 1745, 1692, 13050,
    152, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1560, 0, 3.7, 0, 19.5, 18.0,;
```

Appendix B

System Documentation: Classes

Table B.1 Attributes and operations for the classes in the system

Attributes / Operations	**Brief Description**
Class: Variable	
- name: string	The name of a variable.
- filePath : string	The path for a data file, if applicable.
# binaryLength : int	The binary string length of this variable.
# accessDataStatus : bool	A switch to indicate whether a data file is required to define or to use a variable.
~Variable() {virtual}	Deconstructor function.
+ getName() const: string	To get the variable name.
+ setName(string)	To set the variable name.
+ setAccessDataStatus(bool)	To set a value to the member *accessDataStatus*.
+ getAccessDataStatus() const : bool	To get the value of *accessDataStatus*.
+ setFilePath(string)	To set the file path required by a variable.
+ getFilePath() const : string	To get the file path required by a variable.
+ geBinarytLength() const: int	To get the binary string length needed to represent a variable.
+ computeBinaryLength() {virtual}	To calculate the binary string length for a variable. Since the calculation depends on variable type, this operation is defined as a virtual function.
+convertToRealValue(vector<double>&, vector<bool>&, int&, int&) {virtual}	To convert the binary string to real values. The decoded real values are stored in the vector of the first parameter. The chromosome is transferred as the second parameter. The starting position of the variable in the chromosome is the 3rd parameter, and the index of the variable is the 4th parameter.
+ print(ofstream&) const {virtual}	To print out the information about a variable.
Class: ContinuousVariable	
- lowerBound : double	Lower bound value.
- upperBound : double	Upper bound value.
- precision : double	Required precision of a continuous variable.

Attributes / Operations	**Brief Description**
+ getLowerBound() const : double	To get the lower bound for a variable.
+ getUpperBound() const : double	To get the upper bound for a variable.
+ getPrecision() const : double	To get the precision.
+ setLowerBound(double)	To set the lower bound for a variable.
+ setUpperBound(double)	To set the upper bound for a variable.
+ setPrecision(double)	To set the precision.
+ computeBinaryLength() {virtual}	To calculate the required binary string length for a continuous variable.
+convertToRealValue(vector<double>&, vector<bool>&, int&, int&) {virtual}	To convert its binary string to real value for a continuous variable.
+ print(ofstream&) const {virtual}	To print out the information about a continuous variable.
Class: DiscreteVariable	
- discreteValueVec : vector<double>	A vector to store the discrete values of a discrete variable.
+ getTotalNumber() const : int	To get the total number of values for a discrete variable.
+ getValue(int) const : double	To get the value for a designated alternative.
+ addValue(double)	To add a value to vector *discreteValueVec*.
+ computeBinaryLength() {virtual}	To calculate the required binary string length for a discrete variable.
+convertToRealValue(vector<double>&, vector<bool>&, int&, int&) {virtual}	To convert its binary string to real value for a discrete variable.
+ print(ofstream&) const {virtual}	To print out the information about a discrete variable.
Class: StructuredVariable	
subVariableVec : vector<Variable*>	This vector stores all direct sub-level variables controlled by the structured variable.

Attributes / Operations	Brief Description
subVariableNumVec : vector <int>	This vector stores the number of direct sub-level variables affiliated to each alternative of the structured variable. Each alternative may have different number of sub-level variables. The number counts the direct sub-level variables only. This means that if the structured variable has a sub-structured variable, the variables controlled by that sub-structured variable are not counted.
totalSubVariableNumVec : vector <int>	The number counts all sub-level variables controlled by each alternative of a structured variable. This means that if the structured variable has a sub-structured variable, the variables controlled by that sub-structured variable are counted.
subStructuredIndexVec : vector<int>	The indices (subscripts in a vector) of direct sub-level structured.
totalSubChromosomeLengthVec : vector<int>	This vector stores the binary string length for each alternative of a structured variable. Here, the binary string length counts all sub-level variables.
+ addSubVariable(Variable*)	To add a *Variable* (pointer) to the vector *subVariableVec.*
+ getSubVariable() const : Variable*	To get a variable.
+ addSubVariableNumber(int)	To add a number to the vector *subVariableNumVec.*
+ getSubVariableNumber(int) const : int	To get the direct sub-level numbers for a specified alternative.
+ addSubStructuredIndex(int)	To add an index to the vector *subStructuredIndexVec.*
+ getSubStructuredIndex(int) const : int	To get the index of a sub-structured variable.
+ getSubStructuredNumber() const : int	To get the total number of direct sub-structured variables.
+ computeTotalSubVariableNumber()	To compute the total number of sub-level variables for each alternative of a structured variable.
+ computeTotalSubChromosomeLength()	To compute the total number of binary string length for each alternative of a structured variable.
+ getTotalSubVariableNumber(int) const : int	To get the total sub-level numbers for a specified alternative.

Attributes / Operations	Brief Description
+ getTotalSubChromosomeLength (int) const : int	To get the total binary string length for a specified alternative.
+ computeBinaryLength() {virtual}	To calculate the binary string length required by a structured variable.
+convertToRealValue(vector<double>&, vector<bool>&, int&, int&) {virtual}	To convert its binary string to real values for a structured variable. This function decodes a structured variable and all its sub-level variables.
+ print(ofstream&) const {virtual}	To print out the information about a structured variable.
Class: VariableSet	
- variableVec: vector<Variable*>	Vector is used because "insert" and "delete" operation are rarely applied. Pointers to class *Variable* must be used because dynamic binding technique is employed for variables. All the variables in this vector are at the top level.
- structuredIndexVec : vector<int>	The subscripts of structured variables in the vector *variableVec*.
- ptrVariableSet : VariableSet* {static}	The modifier "static" is used because this class is treated as a singleton pattern.
# VariableSet()	Constructor must be invisible to the outside due to the use of the singleton design pattern.
+ ~VariableSet()	To reclaim memory allocated in the vector *variableVec*.
+ instance() : VariableSet* {static}	To get the unique instance of this class.
+ defineVariables(const char*)	To define variables based on the data file (the parameter).
+ getVariable(int) const : Variable*	To get a variable at the specified position.
+ addStructuredIndex(int)	To add an index (subscript in a vector) of a structured variable at top level to the vector *structuredIndexVec*.
+ getStructuredIndex(int)	To get an index (subscript in a vector) of a structured variable at the specified position.
+ getStructuredNumber() const : int	To get the total number of structured variables at the top level.

Attributes / Operations	Brief Description
+ getTopVariableNumber() const : int	To get the total number of variables at the top level.
+ computeTotalChromosomeLength() const : int	To compute the total chromosome length to represent all variables (top and all sub-levels).
+ computeTotalVariableNumber() const : int	To compute the total number of variables (top and all sub-levels).
+ computeVariableInformation()	To compute the binary length of all variables and those special attributes for structured variables.
Class: Simulation	
# simulationName: string	The name of the simulation program.
# failureValue : double	Whenever some error happens in its running process, the *failureValue* is returned to the optimizer. It is necessary to have this attribute so that the program can continue even some run error happens.
# simulationCallNumber : int {static}	The number of simulation calls.
# variableTopIndexVec : vector<int> {static}	The indices of top variables according to a predefined order. The index in this vector does not consider sub-level variables.
# variableTotalIndexVec : vector<int> {static}	The indices of top variables according to a predefined order. The index in this vector considers sub-level variables.
+ ~Simulation() {virtual}	Deconstructor function.
+ evaluateFunction(const double*, vector<double> &) =0 {virtual}	To calculate objective function values for an optimization problem. The calculation process varies with simulation, so it is a virtual function.
+ executeSimulation(const double*) {virtual}	To run the primary simulation program based on the transferred values of variable.
+ scanVariableOrder() {virtual}	To determine the values in the two vectors *variableTopIndexVec* and *variableTotalIndexVec*. This operation is called only once before the optimization.
+ loadData(const char*) {virtual}	To read data from a text file.
+ getSimulationName() const : string	To get the name of the simulation program.
+ setSimulationName(string)	To assign a name to the simulation program.

Attributes / Operations	**Brief Description**
+ getFailureValue() const : double	To get the *failureValue.*
+ setFailureValue(double)	To set the *failureValue.*
+ getSimulationCall () : int {static}	To return the number of simulation calls.
Class: EnergySimulation	
# area : double	Floor area (m^2).
# floorHeight : double	Floor-to-floor height (m).
# serviceLife : int	Service life for the life-cycle analysis (yr).
# heatingSystemEfficiency : double	Heating system efficiency.
# coolingSystemEfficiency : double	Cooling system efficiency.
# fixedInputFile : string	Path of the file *fixedPart.idf.*
# applicationFile : string	Path of the file *toolkit.exe.*
# typicalHeatWeatherVec : vector<string>	Paths of weathers files used for heating energy consumption.
# typicalCoolWeatherVec : vector<string>	Paths of weathers files used for cooling energy consumption.

# totalWallArea : double {static} 	
# totalRoofArea : double {static} 	
# totalFloorArea : double {static} 	
# windowArea[4] : double {static} 	
	They are set as attributes because (1) they are used in many places in the simulation program; (2) their calculations need many procedures.

# monthlyEnergyVec : vector<double> {static}	This vector stores monthly energy consumption (kWh) calculated by the simulation program.
+ executeSimulation(const double*) {virtual}	To run the energy simulation based on the transferred values of variable. This operation is supported by a number of private functions.
+ evaluateFunction(const double*, vector<double> &) {virtual}	To calculate the annual operating energy consumption (kWh).
+ scanVariableOrder() {virtual}	To determine the values in the two vectors *variableTopIndexVec* and *variableTotalIndexVec.*
+ loadData(const char*)	To read energy simulation data from the *simulationData.txt* text file.

Attributes / Operations	Brief Description
+ prepareIntermediateIDF(const double*): bool	To prepare an intermediate file for the toolkit input. This intermediate file contains the *fixedPart.idf* and the envelope configuration, but does not include the data in the weather files. If this operation finds that the transferred variables cannot formulate a feasible design, it returns *false*.
- prepareFinalIDF(const double*) const	To establish the *toolkit.idf* file based on the intermediate file prepared by the operation *PrepareIntermediateIDF()* and the applicable weather file.
- executeToolkit(int&, vector<double>&)	To call the toolkit via the operating system. The first parameter indicates whether errors occur in executing the toolkit, and the second parameter contains hourly loads.
- checkErrorFile() const: bool	To check whether errors occur in executing the toolkit by examining the file *tookit.err*.
Class: CostSimulation	
- localCostFactor : double	Local cost factor.
- discountRate : double - fuelPriceEscalationRate : double	Discount rate and fuel price escalation rate. Both include inflation.
- demandStep : int - demandThresholdVec : vector<double> - demandPriceVec : vector<double>	Demand price structure.
- consumptionStep : int - consumptionThresholdVec : vector<double> - consumptionPriceVec : vector<double>	Energy consumption price structure.
- peakHeatWeatherVec : vector<string>	Paths of weather files used for heating peak loads.
- peakCoolWeatherVec : vector<string>	Paths of weather files used for cooling peak loads.
+ evaluateFunction(const double*, vector<double> &) {virtual}	To calculate cost-related objective function values.
+ loadData(const char*)	To read cost-related parameters from the file *simulationData.txt*.

Attributes / Operations	Brief Description
# calculateLCC(const double*, vector<double>&)	To calculate the life-cycle cost.
# calculateInitialCost(const double*, vector<double>&)	To calculate the initial cost.
# calculateOperatingCost(const double*, vector<double>&)	To calculate the operating cost.
Class: CostConstraint	
- rightHandValue	Right hand value for the cost-related constraints.
+ evaluateFunction(const double*, vector<double> &) {virtual}	To calculate the difference between the right hand value and the corresponding cost-related objective function value.
Class: ImpactSimulation	
- alphaEmbodiedFuelVec : vector<double>	The α values for the fuel types considered in embodied energy.
- materialExergyVec : vector<double>	The chemical exergy of all considered non-fuel materials.
- alphaOperatingFuelVec : vector<double>	The α values for the fuel types considered in operating energy.
- etaOperatingFuelVec : vector<double>	The η values for the fuel types considered in operating energy.
- emissionFactorVec : vector<double>	The emission factors for the considered operating fuel type.
- abatementExergyVec : vector<double>	The abatement exergy for all considered waste emissions.
+ evaluateFunction(const double*, vector<double> &) {virtual}	To calculate environmental impact related objective function values.
+ loadData(const char*)	To read environmental impact related parameters from the file *simulationData.txt*.
# calculateLCEI(const double*, vector<double> &)	To calculate the life-cycle environmental impact.
# calculateLCEnergy(const double*, vector<double> &)	To calculate the life-cycle energy.

Attributes / Operations	Brief Description
- calculateEmbodiedImpact(const double*, vector<double> &)	To calculate the environmental impacts in the pre-operation stage.
- calculateOperatingImpact(const double*, vector<double> &)	To calculate the environmental impacts in the operation stage.
- calculateEmbodiedEnergy(const double*, vector<double> &)	To calculate the embodied energy for a building design alternative.
Class: ImpactConstraint	
- rightHandValue	Right hand value for the environement-related constraints.
+ evaluateFunction(const double*, vector<double> &) {virtual}	To calculate the difference between the right hand value and the corresponding environment-related objective function value.
Class: SimulationSet	
- criterionVec: vector<Simulation*>	This vector stores the simulations used to evaluate objective functions.
- constraintVec: vector <Simulation*>	This vector stores the simulations used to evaluate functional constraints.
- primarySimulationVec: vector <Simulation*>	This vector stores the primary simulations, which are characterized as: (1) their results are usually used by several other criteria or constraints; (2) they take much computation time.
- ptrSimulationDefinitionStrategy : SimulationDefinitionStrategy*	This class maintains a pointer to the class *SimulationDefinitionStrategy* to define simulations.
- ptrSimulationSet: SimulationSet* {static}	The modifier "static" is used because this class is treated as the singleton pattern.
# SimulationSet()	Constructor must be invisible to the outside due to the use of the singleton pattern.
+ ~SimulationSet()	Deconstruction function to release dynamic memories for the members in the three simulation vectors.
+ instance() : SimulationSet* {static}	To get the unique instance of this class.
+ defineSimulation(const char*)	To define the simulation programs based on the file *environment.txt,* if applicable.

Attributes / Operations	Brief Description
+ setSimulationDefinitionStrategy (SimulationDefinitionStrategy*)	To set the appropriate simulation definition strategy.
+ addCriterion(Simulation*)	To add a criterion to the vector *criterionVec*.
+ addConstraint(Simulation*)	To add a constraint to the vector *constraintVec*.
+ addPrimarySimulation(Simulation*)	To add a primary simulation to the vector *primarySimulationVec*.
+ getCriterion(int) : Simulation*	To get a simulation for criterion.
+ getConstraint(int) : Simulation*	To get a simulation for constraint.
+ getPrimarySimulation(int) : Simulation*	To get a primary simulation.
+ getTotalCriterionNumber() const : int	To get the total number of criteria.
+ getTotalConstraintNumber() const : int	To get the total number of constraints.
+ getTotalPrimarySimulationNumber() const : int	To get the total number of primary simulations.
Class: SimulationDefinitionStrategy	
+ defineSimulation(const char*, SimulationSet*) {virtual}	To define simulation programs. The definition may be based on a file, which is indicated as the first parameter. A pointer to the class *SimulationSet* is transferred as the second parameter to use its member functions.
Class: EnergyDefinitionStrategy	
+ defineSimulation(const char*, SimulationSet*) {virtual}	To define simulation programs for the system. This operation is based on the file *environment.txt*.
Class: Individual	
- fitness : double	The fitness value of an individual.
- failureStatus : bool	The switch to indicate whether the individual cause simulation failure.
- feasibleStatus: bool	The switch to indicate whether the individual violates functional constraints.

Attributes / Operations	**Brief Description**
- evaluationStatus : bool	The switch to indicate whether the function values are valid. It is used to determine whether to call the simulation to get the values for the objective functions and functional constraints.
- chromosomeVec : vector<bool>	This vector stores the binary string.
- realValueVec : vector<double>	This vector stores the decoded real values for variables.
- criterionValueVec : vector<double>	This vector stores the objective function values for an individual.
- constraintValueVec : vector<double>	This vector stores the constraint values for an individual.
+ Individual()	The constructor allocates spaces for the vectors in this class.

+ setFitness(double)
+ setFailureStatus(bool)
+ setFeasibleStatus(bool)
+ setEvaluationStatus(bool)
	Set functions.

+ getFitness() const: double + getFailureStatus() const: bool + getFeasibleStatus() const: bool + getEvaluationStatus() const: bool	Get functions.
+ initialize()	To initialize the binary string of an individual randomly.
+ decode()	To convert the chromosome to real values.
+ callSimulation()	To call the simulation program and calculate the objective functions and constraints.
+ mutate(double)	To carry out the bit-wise mutation based on transferred mutation probability.
+ crossover(const Individual&, Individual&, Individual&)	To swap the binary strings of this individual and another one (the first parameter). The two produced individuals are returned with the second and the third parameters.

Attributes / Operations	**Brief Description**
+ computeEuclideanDistance(const Individual&, const double *) const: double	To calculate the normalized distance between this individual and another one (the first parameter). The second parameter is used for normalization.
+ operator = = (const Individual&) const: bool	To compare two individuals to check whether their binary strings are the same.
+ checkDominance (const Individual&) const: bool	To check whether the individual dominates another one (the parameter).
+ checkEqualOrDominance (const Individual&) const: bool	To check whether the individual is equal to or dominates another one (the parameter).
+ print(ofstream&) const	To output the individual information including the chromosome, the real values, the objective function values, and the constraint values to a data file.
- calculateEffectiveLength (const Individual*) : int	To calculate the effective crossing length for two individuals. The effective length refers to the chromosome length not counting the inactive structured variables for both individuals. This function is designed for structured GAs in order to produce potentially different children.
- calculateActualCrossPoint (const Individual*, int) : int	To calculate the actual crossover point based on the point selected from the effective length.
Class: IndividualSet	
# individualVec : vector<Individual>	This vector stores individuals in the evolution process. The individuals may be elites in an external population or those individuals participating in crossover and mutation in an active population.
# populationSize : int	The maximum number of individuals in a population.
# generationNumber : int	The current generation number.
# bestIndexVec : vector <int>	This vector stores the indices of those individuals that have the best (minimum) objective function values.
# worstIndexVec : vector <int>	This vector stores the indices of those individuals that have the worst (maximum) objective function values.
# averageCriterionVec : vector <double>	This vector stores the averages of all feasible individuals for each criterion.

Attributes / Operations	**Brief Description**
# deviationCriterionVec : vector <double>	This vector stores the standard deviations of all feasible individuals for each criterion.
+ setPopulationSize(int) {virtual}	To set a value to the *populationSize*. The population size is not necessarily equal to the number of individuals in the vector *individuallVec* for an external population.
+ getPopulationSize() const : int	To get the number of individuals in the vector *individualVec.*
+ getPopulationCapacity() const : int	To get the value of member *populationSize.*
+ getGenerationNumber() : int	To get the value of member *generationNumber*.
+ getIndividual(int) : Individual*	To get the individual at the specified position.
+ replaceIndividual(const Individual&, int)	To replace the individual at the specified position (the second parameter) by a new one (the first parameter).
+ addIndividual(const Individual*)	To add an individual to *individualVec.*
+ emptyIndividual()	To remove all individuals in the *individualVec* and resize the vector as 0.
+ getBestIndex(int =0) const: int	To get the index of the individual with the minimum function values for the specified criterion (the default is the first criterion, applicable to single objective optimization).
+ getWorstIndex(int =0) const: int	To get the index of the individual with the maximum function values for a specified criterion (the default is the first criterion, applicable to single objective optimization).
+ getAverageCriteria(int =0) const: double	To get the average for the specified criterion.
+ getDeviationCriteria(int =0) const: double	To get the standard deviation for the specified criterion.
+ computeStatistics()	To calculate the statistics for the generation. The statistics include the best and the worst individuals, the average and the standard deviation for each criterion.
+ findParetoFront(vector<int> &)	To find the Pareto front of the population. The indices of Pareto solutions are stored in the parameter.

Attributes / Operations	Brief Description
+ determineRepresentative(vector<int> &)	To find a representative for a group of individuals. The representative is the one closest to the centroid.
+ print(ostream &) const {virtual}	To output the population to a data file. This operation outputs detailed information including all individuals.
+ printConcise(ostream &) const {virtual}	To output the population to a data file. This operation outputs brief information including the objective function values for all individuals.
Class: ActivePopulation	
- crossoverProbability : double	Crossover probability.
- mutationProbability : double	Mutation probability.
+ setPopulationSize(int) {virtual}	To assign value to *populationSize* and to allocate space to *individualVec*

+ setCrossoverProbability(double)
+ setMutationProbability(double)
	Set functions.

+ initialize()	To initialize the population.
+ comparePopulation(ActivePopulation&)	To compare the individuals in this population and those in the population of the previous generation (the parameter) in order to find those that have identical genes. The purpose of this operation is to save time by reducing simulation calls.
+ evaluateIndividuals()	To calculate criterion values and constraint values for each individual in the population.
+ matePopulation(ActivePopulation&, bool =false)	To carry out the mating procedure. The population after crossover is stored in the second parameter. Mating can be done with or without restriction, which is determined by the second parameter.
+ mutatePopulation()	To mutate each individual in the population.
+ keepBoundary(ActivePopulation&)	To keep the boundary individuals from loss in the evolution process. The previous population is transferred as the parameter.
+ print(ostream &) {virtual}	To output the population to a data file.
Class: Optimizer	
+ ~Optimizer() {virtual}	

Attributes / Operations	**Brief Description**
+ defineOptimizationParameter(const char*) {virtual}	To define optimization parameters based on the data file *(environment.txt)*.
+ optimize(ofstream&) =0 {virtual}	To run the optimization algorithm. The intermediate results may be output to an data file (the parameter) for analysis.
Class: GAOptimizer	
# maximumGeneration: int	The maximum generations to terminate the GA program.
# mateRestrictionGeneration: int	The number of generations from which mating restriction applies.
# ptrExternalPopulation: IndividualSet*	The external population used to store elites. Individuals in the external population usually do not crossover and mutate.
# ptrActivePopulation: ActivePopulation*	The usual GA population that has crossover and mutation operations.
# ptrFitnessStrategy: FitnessAssignmentStrategy*	A pointer to the base class for fitness assignment.
# ptrSelectionStrategy: SelectionStrategy*	A pointer to the base class for GA selection.
# ptrElitistStrategy: ElitistStrategy*	A pointer to the base class for GA elitism.
+ ~GAOptimizer() {virtual}	To release dynamic memories allocated to the members.
+ optimize() {virtual}	To run the genetic algorithm. Depending on the number of objective functions, this function will call the appropriate private functions.
+ defineOptimizationParameter(const char*) {virtual}	To define optimization parameters based on the data file *environment.txt*.
+ getActivePopulation(): ActivePopulation* const	To get the handle for the active population.
+ getExternalPopulation() : IndividualSet* const	To get the handle for the external population.
- optimizeSingleObjective(ActivePopulation*, ofstream&)	To run the single objective GA.

Attributes / Operations	Brief Description
- optimizeMultiObjective(ActivePopulation*, ofstream&)	To run the multi-objective GA.
Class: FitnessAssignmentStrategy	
+ assignFitness(ActivePopulation&, IndividualSet* =0) {virtual}	The public interface to implement fitness assignment methods. The first parameter indicates the population whose individuals are assigned fitness. The external population is transferred as the second parameter because some fitness assignment methods require the information of the external population.
Class: MaxSOGAFitness	
+ assignFitness(ActivePopulation&, IndividualSet* =0)	To implement the fitness assignment method for unconstrained single objective optimization. This operation regards the difference between an individual's objective function value and the maximum objective function value in the population as the individual's fitness.
Class: DebConstrainedSOGAFitness	
+ assignFitness(ActivePopulation&, IndividualSet* =0)	To implement the fitness assignment method for constrained single objective optimization. This operation gives feasible individuals higher fitness values than those infeasible ones. Infeasible individuals are assigned fitness values according to their violation of constraints.
Class: RankMOGAFitness	
+ assignFitness(ActivePopulation&, IndividualSet* =0)	To implement the rank-based fitness assignment method for unconstrained multi-objective optimization.
Class: FrontMOGAFitness	
assignFitness(ActivePopulation&, IndividualSet* =0)	To implement the front-based fitness assignment method for unconstrained multi-objective optimization.

Attributes / Operations	Brief Description
Class: SelectionStrategy	
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	The public interface to implement selection methods. The first parameter is the population from which individuals are selected. The second parameter is the population produced after selection. The third parameter is the external population that may be used by some selection strategies.
Class: Tournament	
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	To implement the binary tournament selection method.
Class: RWS	
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	To implement the roulette wheel selection method.
Class: SUS	
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	To implement the stochastic universal selection method.
Class: SRS	
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	To implement the stochastic remainder selection method.
Class: InsertTournament	
- immigratedEliteNumber: int	The maximum number of elites immigrated from the external population into the mating pool (the second parameter of the operation *doSelection*).
+ InsertTournament(int)	Constructor function to assign a value to the attribute *immigratedEliteNumber*

Attributes / Operations	Brief Description
+ doSelection(ActivePopulation&, ActivePopulation&, IndividualSet* =0) {virtual}	To implement the insertTournament selection method. This operation is carried out in two steps. First, at most immigratedEliteNumber of elites are randomly chosen from the external population and put into the mating pool. Then, the remaining number of individuals are selected with the tournament method.
Class: ElitistStrategy	
+ preserveElite(ActivePopulation&, IndividualSet*, ActivePopulation* =0) {virtual}	The public interface to protect robust individuals in a population. The first parameter is the genetic population from which elites are picked. The second parameter is the external population to store elites. The third parameter is the previous population that may be used by some algorithms.
Class: ConstrainedSOGAElitism	
- preservedEliteNumber: int	The number of robust individuals to be preserved.
+ ConstrainedSOGAElitism(int)	Constructor function to assign a value to the attribute *preservedEliteNumber*.
+ preserveElite(ActivePopulation&, IndividualSet*, ActivePopulation* =0)	To implement the elitist strategy for constrained single objective optimization. After a new generation is formed, the *preservedEliteNumber* of worst individuals are replaced with the same number of best individuals from the previous generation if the latter is more robust than the former.
Class: ClusterMOGAElitism	
+ preserveElite(ActivePopulation&, IndividualSet*, ActivePopulation* =0)	To implement the elitist strategy for multi-objective optimization. For each generation, the non-dominated individuals are copied to the external population, and the dominated ones in the external population are removed. If the external population reaches its capacity, clustering is used to remove some individuals in crowded regions.
- compactByCluster(IndividualSet&, vector<int>&, vector<int>&)	To carry out the clustering algorithm. The second parameter indicates the indices of those non-dominated solutions in the individual set (the first parameter). The indices of those individuals after clustering are stored in the third parameter.

Attributes / Operations	**Brief Description**
Class: SystemControl	
- ptrVariableSet : VariableSet*	A pointer to the class *VariableSet*, from which the system accesses all the information about variables.
- ptrOptimizer : Optimizer*	A pointer to the class *Optimizer*, from which the system calls the optimization algorithm.
- ptrSimulationSet : SimulationSet*	A pointer to the class *SimulationSet*, from which the system accesses all the information about the simulation programs.
+ ~IntegratedSystem()	To release dynamic memories allocated to the members.
+ systemRun()	To run the system.
Class: Utility	
+ inString(ifstream&, char): string	To read a string from a data file. The second parameter indicated the delimiter character.
+ inInteger(ifstream&, char): int	To read an integer from a data file. The second parameter indicated the delimiter character.
+ inDouble(ifstream&, char): double	To read a real value from a data file. The second parameter indicated the delimiter character.
+ peekNonWhiteSpaceChar(ifstream&): char	To peek the next non-white space character in a data file.
+ compareString(string&, string): bool	To compare two strings. The comparison is not case-sensitive.
+ randomInteger(int, int): int	To produce a random integer between two boundary values.
+ randomBit(): bool	To produce a random value which is equal to 1 or 0.
+ randomReal(double, double): double	To produce a random real number between two boundary values.