Minimizing Electricity Costs with an Auxiliary Generator Using Stochastic Programming

by

Paul Rafiuly

B.Arch., Architecture
University of Texas at Austin, 1998

Submitted to the Department of Architecture and the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Building Technology
and
Master of Science of in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
September 2000

Signature of Author...
Department of Architecture
August 4, 2000

Certified by...
Leslie K. Norford
Associate Professor of Building Technology
Thesis Supervisor

Certified by...
John N. Tsitsiklis
Professor of Electrical Engineering
Thesis Supervisor

Accepted by...
Stanford Anderson
Chairman, Departmental Committee on Graduate Students
Department of Architecture

Accepted by...
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Minimizing Electricity Costs with an Auxiliary Generator Using Stochastic Programming

by

Paul Rafiuly

Submitted to the Department of Architecture and the Department of Electrical Engineering and Computer Science on August 4, 2000 in partial fulfillment of the requirements for the Degrees of Master of Science in Building Technology and Master of Science of Electrical Engineering and Computer Science

ABSTRACT

This thesis addresses the problem of minimizing a facility's electricity costs by generating optimal responses using an auxiliary generator as the parameter of the control systems. The goal of the thesis is to find an optimization method that can cope with the uncertainty in the building load while also handles the complex electricity price structures. The building load is the random factor of the stochastic problem and is composed of the weather load and the occupancy load.

Several optimization techniques such as Dynamic Programming and Linear Programming (deterministic optimization) are looked at. Stochastic Programming using Nested Bender's Decomposition method is chosen and studied to solve the optimization problem. Stochastic Programming, which is a hybrid of Dynamic Programming and Linear Programming, is used because it can cope with the complex electricity price structures and the uncertainty of the building load while avoiding an explosion in the number of states.

However, the method is not suitable for our problem, which is a Mixed Integer Programming problem. Moreover, the random sampling adds some limitations on the method. In addition, the high memory requirement and the extensive computational time prohibit the method from being used for a long planning period. Hence, a new control system, which is the combination of Stochastic Programming and Linear Programming, is proposed. The key of this proposed method is the reduction of the problem into a two-stage stochastic problem.

Thesis Committee:

Thesis Supervisor: Leslie K. Norford
Title: Associate Professor of Building Technology, Department of Architecture

Thesis Supervisor: John N. Tsitsiklis
Title: Professor of Electrical Engineering

Acknowledgement

First, I want to thank my advisor, Prof. Leslie K. Norford, who has played an instrumental role in educating and guiding me throughout my years in MIT. He has been very understanding and supportive, and without his vision this concept would have never been evolved into the thesis. I would also like to thank Prof. John T. Tsitsiklis, whose insights and great patience are invaluable to the completion of this thesis. Further, I want to thank Prof. Alvin W. Drake who has always been very kind. Had it not been for his trust and encouragement, all of this would have never happened.

In addition, I would like to thank all people whose names I did not mention, who have contributed not only to the completion of this thesis, but also to my life.

Most importantly, to my parents and God for their endless love and supports in everything that I do, I thank them. To them, this work is dedicated.

Table of Contents

Chapter I: Introduction

1.1 Motivation

1.1.1 Background

The rapid transitions occurring in the utilities industries and ongoing deregulation give rise to the Real Time Pricing (RTP) Structure (the electricity price varies on hourly basis), which has evolved from a Time of Use Pricing (TOU) Structure (the electricity price varies on a block of hours basis). On the top of that, some schedules levy a demand charge that is based on the maximum instantaneous demand recorded over a month. This demand charge can sometimes span its effect over a year period.

Further, tools to reduce the electricity cost for buildings have been developed, for example the Automated Real Time Pricing Control System currently implemented by Honeywell (Shavit *et al.* 1998). However, this tool is not adequate because it actually does not avoid the peak-hour tariff and demand charges. It only reduces the load by lowering the comfort level to its lowest limit and by utilizing the automated lighting control. Thus, the system can be further improved by using another device (for example a generator or a thermal storage) to shift loads. The control logic for this automated control system is revealed in the following example: "If the RTP Price is greater than 10c kWh", and "If the air handling system AC-2 is in its occupied period", and "If the space temperature is not over 80 F (27 C)", then "Command system AC-2 off". The cost saving with the current control strategy in the Marriott Marquis Hotel, New York City, was US$1 million over a four-year period.

1.1.2 Objectives

The thesis seeks to construct a set of algorithms that can be used to minimize the cost of electricity by using an auxiliary generator as the control strategy. This answers the need for the building energy management system: a unified tool that can respond to varied tariff schedules. 'Respond' means the tool can be used to discriminate between utilities based on the net electricity cost. The 'unified tool' is what distinguishes the thesis from other research in the area. It attempts to include and reduce the effect of uncertainty in the optimization model, combining it with a complex pricing structure and generator model.

A generator is commonly used as a source of emergency power in most buildings. In this thesis, a generator is a single component that helps to absorb some of the load during peak-hour to avoid high price and charges. In many cases, a generator is permitted to operate for a limited number of hours per year, to minimize local air and noise pollution. Moreover, it is also characterized with several transition stages: start-up, ramp-up, synchronization, ramp-down, and shut-down.

1.1.3 Related Work

Some work has been done in this field. The closest one is Shanbag's work (Shanbag 1998). Shanbag considered three different systems in minimizing the energy costs in buildings: an auxiliary generator, a thermal storage, and lighting control. However, his work assumes that the building manager has perfect knowledge of the building's load and more importantly, it does not consider the generator operating-hour constraint nor the seasonality aspects of the building load (affected directly by weather).

The stochastic load of the building is the main focus of the thesis. The stochastic load requires a totally different algorithm than the one with perfect knowledge of the load. Furthermore, it has a wide range of implications in solving the diverse pricing issues and in modeling the actual complex generator behavior.

1.2 Problems Description

1.2.1 Optimization under Uncertainty

The uncertainty in load forecasting is one of the factors that is most often forgotten in a building energy optimization tool. Future loads are very hard to calculate and depend on unpredictable external factors: weather, occupancy type, building structure, infiltration (building deteriorates through time), etc. The accuracy of the load prediction raises the question of the success of the optimization itself. This thesis tries to find the optimization method that is the most appropriate (by considering accuracy and running time) to minimize the building load using an auxiliary generator.

The goal for this thesis is to create an algorithm that combines the load forecasting model with the generator model and price structure. This tool will try to optimize the use of the auxiliary generators to minimize the electricity cost. The generator has an operating-hour constraint within a year. Moreover, the hours of the generator are used mainly to avoid an increase in the demand charges (that could have an effect for a year in the future) or peak hour charges. The tool will answer whether a building operator should use the hour of the generator for the next hour given: the predetermined control strategies, the remaining generator operation hours (taking into account the future load prediction), the updated weather data, and the last hour's electrical price.

1.2.2 Building Thermal Load

Building load is divided into thermal load and non-thermal load. We consider the non-thermal load as fixed, and focus more on the variability of the thermal load. Thermal load is affected by several factors:

- Weather: the forecasting of weather itself is another sub-problem worth studying more closely.
- Thermal Comfort: this looks at the criteria from which the load is to be determined. Some papers have used PMV (Predicted Mean Vote) as a basis (Braun and Keeney 1996), however, for the purpose of this thesis, it is more practical to use the dry bulb temperature of the internal ambient air than PMV as the set point control. A thermal model based on a 2R1C circuit analogy can be used to illustrate the connection between the outdoor air temperature and the internal ambient air temperature.
- Building Thermal Data: we can use the generic numbers from the ASHRAE handbook for typical office building.
- Historical Load Data: this is a common weighing factor for predicting the seasonality of the building. Moreover, by using past data, the expected load and its PMF during a particular month can be derived.

1.2.3 Cases

During this planning stage several questions should be answered: how do we allocate the limited generator operating-hours based on what happen earlier in the month and is expected to happen later in the month. For example if during the last week an unexpected cold front is coming, should we allocate the generator now as planned or save it for the last week of the month? Will the strategy differ if we have a demand charge levied on the top of the RTP charge? How different will the strategy be if the demand charge varies from hour to hour? What should we do if we are expecting a prolonged heat wave? If the load shrinks as the month goes by, should we save the generator hour for the future months? Those are several extreme cases upon which the different optimization algorithms will be tested.

1.3 Research Approach

1.3.1 Optimization Methods

Several different optimization methods have been considered: Linear Programming, Dynamic Programming, and Stochastic Programming. In approaching the problem, Stochastic Programming (also called Dual Dynamic Programming) using Nested Bender's Decomposition is used. This method is a combination of Linear Programming and Dynamic Programming (Pereira and Pinto 1991). All of the constraints are laid up in linear programming forms. Moreover, this method breaks down the time horizon into stages, with several states on each one branching out from the previous states, forming sub-problems. These sub-problems will then be solved using Bender's Decomposition method recursively, analogous to the Dynamic Programming Method. This recursion, which is done in the dual space (resulting in a lower bound), is combined with forward simulation, which is done in the primal space (resulting in an upper bound). This is done in several iterations until it converges to a tolerable error value.

This hybrid method has some advantages compared to pure Linear Programming or pure Dynamic Programming. First, the method avoids discretization and the combinatorial explosion with the number of states, unlike Dynamic Programming. Because of that, the method can solve a richer set of problems with more complex constraints than a regular dynamic programming method. Second, unlike its counterpart, the Linear Programming method, the Dual Dynamic Programming method can handle the stochastic nature of the problem.

1.3.2 Optimization Tools

So far several tools have been tried. The first one is *Matlab 5.3 Optimization Solver*. The attractive feature of Matlab is that it is easy to use and it has its own dynamic data structure. The iteration script can be written in Matlab calling the integrated optimization solver function. However, the Matlab Solver is unreliable when it encounters sparse matrices, even for a small problem. Example 6.1.1 and Example 6.1.2 are simulated in Matlab, however, the computed dual variables are wrong (the results are compared to those of CPLEX). The error can be significant and thus skews the final solutions.

The second tool tried is *OPL* by ILOG. OPL is reliable even for a huge problem with a lot of variables. Moreover, it can solve integer programming and combinatorial problems. Unfortunately, this tool does not have a callable library, which is necessary to run the algorithm. The other downside is that the tool does not seem to be able to display the optimal dual variables, which are crucial in the backward recursion process.

The third one used is *CPLEX* by ILOG. CPLEX is very reliable and normally used for large-scale optimization problems. It also has the callable library in C/C++ that allows users to write scripts for their algorithms. CPLEX provides options for the optimization algorithm used to solve LP problems, such as *primal simplex method*, *dual simplex method*, *interior point method*, etc. Moreover, CPLEX provides the Mixed Integer Programming Solver that gives different options of solver algorithms. In all simulations, *Branch and Bound* technique, which is the default MIP solver of CPLEX, is used.

1.3.3. Different Time and Pricing Models

The first model works for RTP electricity charge. It handles different hourly rates assumed to be known ahead (given by the utility companies). In this first model, the random factor will be the building load, which also varies in an hourly basis.

The second model will implement one demand charge period consisting of several days with some consideration to seasonality. Each month has its own building load transition probabilities (based on the weather transition probabilities) to capture the seasonality effect. Moreover, besides minimizing the energy charges, its other goal is to minimize demand charges of that month without further complication of effects on the past and on the future. There are two types of demand charges. The first one is a Flat Demand Charge, which has a fixed demand charge within the demand charge period. The second one is a Time Varying Demand Charge, which has a varying demand charge from time to time.

For each price structure, several cases are tested. The first ones are the deterministic examples, which usually consist of short planning horizon period (three stages or less) examples and long planning horizon period examples. The second ones are the stochastic examples, which just like the deterministic case consist of long and short planning horizon examples.

Chapter II: Building Load

There are two main factors that influence the building load. The first one is the room temperature control, which is removing or supplying heat for the room to maintain a set-point temperature. The second factor is the occupation type, which will be explained in greater detail later. Thus, building load can be modeled as:

$$L_{q\bullet p} = U_{q\bullet p} + OC_{s\bullet r} \quad ; \quad \begin{array}{l} \forall q = \{1,...,24\}; \forall s = \{8,...,18\}; \\ \forall p = \{1,...,7\}; \forall r = \{1,...,5\}; \end{array} \qquad (2.1)$$

q, s = index for hours in a day

p, r = index for days in a week

$L_{q\bullet p}$ = Building Load (KWh) at time $k{*}t$ (=24 hours/day and 7 days/week)

$U_{q\bullet p}$ = Heat removed or supplied (KWh) at time $k{*}t$ (=24 hours/day and 7 days/week)

$OC_{s\bullet r}$ = Occupancy Load (KWh) at time $s{*}r$ (= (8am to 6pm)/day and (Monday to Friday)/week)

Occupancy load is assumed to be during regular working hours for a typical office building, which is used as the example. The subscript '*s*' represent a set of working hours, and '*r*' represents a set of working days.

2.1 Room Temperature Control

Room temperature control is part of the building load, and it is derived based on the heat transfer characteristics of the building with response to outside temperature. The objective is to maintain the ambient temperature within a tolerable human comfort zone, which is between 20 C (lower limit) to 27 C (upper limit) (ASHRAE Handbook of Fundamentals 1997), against the temperature swing outside. Thus, an ambient temperature is the primary variable of interest.

There are several strategies currently used to maintain an ambient temperature. First, by utilizing the range within human comfort zones, one can simply relax the ambient temperature. For example, the set point at which the temperature should be maintained can be fixed at the upper limit of the human comfort zone during peak hours (assuming that cooling is needed) and at the lower limit during off-peak hours (Norford *et al.* 1996). In contrast, during the winter, we maintain the temperature at the lower limit of the human comfort zone during peak hours.

Second, by utilizing the building's thermal mass, one can shift some of the load from the peak hours to the off-peak hours. Thermal mass is a building's capacity to store energy with some time-latency during heat transfer process. This is analogous to a capacitor in an RC circuit. The capacity also depends on the structure, the materials, and the design of the building (a lot of opening or glass surfaces, etc). A space can be preheated or pre-cooled during off-peak hours so that the conventional heating or cooling can be reduced during peak hours (Braun *et al* 1996).

2.2 Heat Transfer Model

A 2R1C model is used since it captures two most important aspects of building load: heat transfer properties of a building and the outside weather. The weather is the stochastic element of the building load and predicting the building load is analogous to predicting the weather with the building's thermal mass as the buffer. This model is studied extensively in Daryanian's dissertation (Daryanian 1989) and is also used in Shanbag's thesis (Shanbag 1998).

There are several assumptions regarding this model. First, the heat transfer relations to the external temperature, the ambient temperature, and the thermal mass are discretized into time blocks (hours). By time discretization and through some algebraic manipulation, the energy needed in the future to remove or to supply heat (the future building load) can be written in a dynamic equation involving all the variables in the present time (Equation 2.4). Second, the thermal mass of air and furniture inside the house has been assumed to be negligible in comparison with that of the walls.

2.2.1 Diagram and Formulation

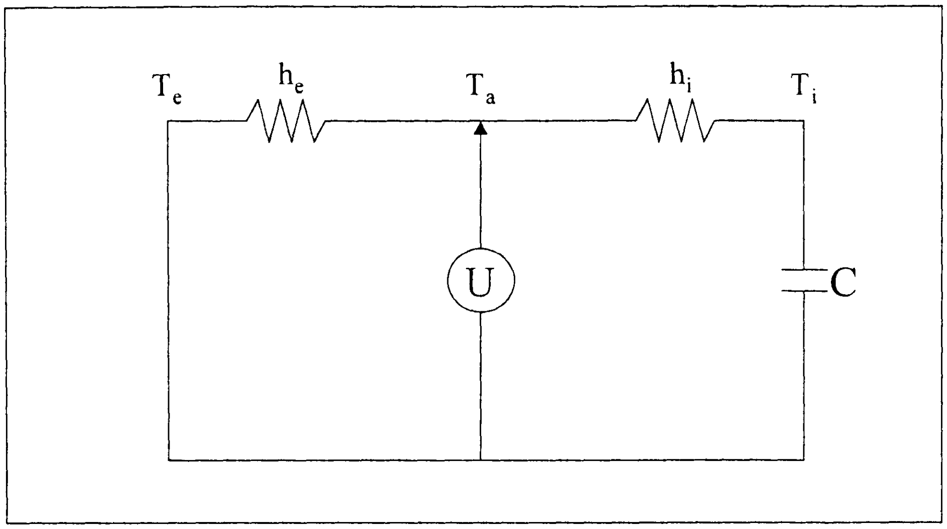


Figure 2.1 2R1C Circuit Analogy (Daryanian 1989)

T_e = External Temperature (^{0}K)

T_a = Ambient Temperature (^{0}K)

T_i = Internal Mass Temperature (^{0}K)

U = Heat Removed/Supplied (kWh)

C_i = Building Thermal Mass (kWh/^{0}K)

h_e = Heat Transfer Coefficient between Air and Outside Surface (kW/^{0}K)

h_i = Heat Transfer Coefficient between Air and Inside Surface (kW/^{0}K)

The objective of this model is to maintain T_a within a tolerable human comfort zone. Thus, the thermal comfort constraint:

$$T_{lower} \leq T_a(k) \leq T_{upper} \quad ;\forall k$$

where

k = index for time (hour)

T_{lower} = Lower limit of the human comfort zone (^{0}K)

T_{upper} = Upper limit of the human comfort zone (^{0}K)

The 2R1C diagram above can be translated into a dynamic heat transfer equation relating the ambient temperature with the external temperature (Shanbag 1998), which is the stochastic factor in our problem:

$$U_k = h_i\Delta t((T_a(k) - T_i(k)) + h_e\Delta t(T_a(k) - T_e(k)) \tag{2.1}$$

$$0 = C_i(T_i(k+1) - T_i(k)) + h_i\Delta t(T_i(k) - T_a(k)) \tag{2.2}$$

Equations (2.1) and (2.2) can be rearranged to show the dynamics of the ambient temperature as a factor of previous stage and current stage ambient temperatures and external temperatures. This algebraic manipulation is necessary to eliminate dependency on internal mass temperature, which is hard to calculate.

$$T_a(k+1) = k_1T_a(k) + k_2U(k+1) + k_4T_e(k+1) + k_5T_e(k) \tag{2.3}$$

$$U(k+1) = \frac{1}{k_2}\{T_a(k+1) - (k_1T_a(k) + k_3U(k) + k_4T_e(k+1) + k_5T_e(k))\} \tag{2.4}$$

where

$$k_1 = 1 - \frac{h_1\Delta t}{C_i} + \frac{h_i\Delta t}{C_1(h_1 + h_e)}$$

$$k_2 = \frac{1}{\Delta t(h_1 + h_e)}$$

$$k_3 = (\frac{h_i}{(h_1 + h_e)})(\frac{h_i}{C_1(h_1 + h_e)}) - (1 - \frac{h_i\Delta t}{C_1(h_1 + h_e)})(\frac{1}{\Delta t(h_1 + h_e)})$$

$$k_4 = \frac{h_e}{(h_1 + h_e)}$$

$$k_5 = (\frac{h_i}{(h_1 + h_e)})(\frac{h_ih_e\Delta t}{C_1(h_1 + h_e)}) - (1 - \frac{h_i\Delta t}{C_1(h_1 + h_e)})(\frac{h_e}{(h_1 + h_e)})$$

2.2.2 Example of Building Coefficients

A typical office building floor is used for the example. The wall is made of concrete and the fenestration is 40% of the wall. The thermal mass of the fenestration (glass) is negligible. Further, the floor is located in the middle of a medium rise building, and the floor above and below are assumed to have the same room temperature and hence, there is no heat transfer between floors. The thickness of the concrete wall is 0.4 m, and the gypsum board and the insulation have a negligible thermal mass.

Further, for the model to be meaningful, there is an upper bound on the time step. The time step of one hour, which falls well within the bound, is used.

Parameter			Total Value	Units
h_i =	7.3	W/m^2K	3.796	kW/K
h_e =	8.4	W/m^2K	4.368	kW/K
Density of the wall =	2100	kg/m^3		
c =	880	J/kg K		
Time Step =	1	hr		
Floor Area =	1000	m^2		
Wall Area =	520	m^2		
Volume of Opaque Wall =	124.8	m^3		
C =			64.064	kWh/K
k_1 =			0.9682976	
k_2 =			0.122489	
k_3 =			-0.115231	
k_4 =			0.5350318	
k_5 =			-0.503329	

Table 2.1 2R1C Example Typical Masonry Office Building Coefficents

By definition:

$$C = c \times \rho \times V$$

C = capacity or thermal mass (J/K)

c = constant-volume specific heat (J/kg K)

ρ = density (kg/m^3)

V = volume (m^3)

2.3 Occupancy Type

Occupancy type is another important stochastic aspect of building load. There are three main components of the load due to occupancy. First, water vapor and heat produced by occupants (depending on their activity) count for a significant portion of the cooling load. The vapor needs to be removed to maintain the humidity in the room within comfort level. The body heat needs to be taken into account since it can be a big factor affecting ambient temperature. Second, equipment, such as computers, machines, etc, dissipate heat and play a big role in affecting the ambient temperature. Third, lighting is also an important factor in the building load, because it dissipates heat and consumes energy.

For the simplicity of the problem, occupancy load will be considered as a fixed load during the day-time period from Monday to Friday. The model can be written as:

$$OC_{s\bullet r} = \textit{Total Lighting Load} + \textit{Total Occupants Load} + \textit{Total Equipment Load}$$

There are other load sources in the building, however, their fraction is negligible compared to the three loads above.

Below is the example of the typical value of each occupancy load's components for a typical office environment (ASHRAE Handbook). The occupancy load assumes a moderately active office work. Moreover, the latent heat represents the humidity as another human's comfort factor.

Parameter			Total Value	Units
Lighting Load =	20	W/m^2	20000	W
Occupants Load = (latent and sensible)	130	W/person	2600	W
Equipment Load = (copier, computer, etc.)	1500	W/person	30000	W
Number of people	20	persons		
Total Occupancy Load =			52.6	kWh

Table 2.2 Occupancy Load

2.4 Weather Forecasting

There are several weather predicting methods considered (Grunenfelder *et al.* 1985). The first one uses the expected temperature weather forecast for the day for the whole 24-hour period. The variation of this method is to use the high temperature for hours between 7.00 am to 7.00 pm and the low temperature for hours between 7.00 pm to 7.00 am. The second method is to say "next hour temperature equals this hour temperature:"

$$E[T_e^{k+1}] = T_e^k$$

The third method uses transition probabilities, where $\Pr(T_e^{k+1} = j \mid T_e^k = i)$ is the transition probability from state i to state j. The transition probabilities are determined by taking a regression analysis over the measured weather data from the previous years. The transition probabilities are different for each month.

i/j	1	2	3	4
1	0.37	0.29	0.25	0.09
2	0.29	0.32	0.25	0.14
3	0.24	0.29	0.27	0.2
4	0.14	0.14	0.23	0.5

Table 2.3 Transition Probabilities Pr for April in Zurich (Grunenfulder *et al* 1985)

i = temperature of this hour

j = predicted temperature of the next hour

The third method is used in this thesis for several reasons. First, it fits well in our Stochastic Programming framework. The Stochastic Programming Method solves a tree-like structure with each node branching into several scenarios, and the state transitions from the parent node to its children nodes are described with transition probabilities. Second, the method captures the seasonality changes. Each month will have its own transition probability table, which reflects the temperature characteristic of that month (for example: in summer the temperature is high). Third, this method gave the best result for weather forecasting in the framework of Dynamic Programming optimization (Grunenfulder *et al.* 1985).

Since, in the optimization problem formulation, the random factor is the building load, the outside temperature PMF needs to be translated into the building load. This translation can be done by using the scenario tree. First, all possible future outside temperatures given the current temperature are laid out (one at each node) in a tree form. Then, the external temperatures at each node can be translated into the weather load by using Equation 2.4. Since weather loads are not independent (depend on the previous hour temperature and thus, weather load) and since there is no one-to-one mapping of external temperatures to weather loads, weather load's PMF will be different from external temperature's PMF.

2.5 Limitations of the Building's Load Model

The building load model has some limitations. First, it does not consider solar heat gain, which consists of direct and indirect solar gain. Solar heat gain is a predominant factor in calculating the building load. Hence, its absence creates a discrepancy between the calculated load and the actual load.

Second, the model leaves out the physical context of the building, such as the condition of its surrounding, its orientation, its geometry, etc. For example, if there is a high-rise south of the building, and the building is located in the U.S. (where south is the predominant facade that is exposed to direct solar radiation), we must take into account the shadow effect. Likewise, if there is a pool of water south of the building, then the indirect solar gain from the reflection will be high. Building's orientation also adds wind as another complicating factor (wind factor). Cold wind normally comes from the North during winter, but in general, wind direction tends to vary on an hourly basis.

Chapter III: Auxiliary Generator

The auxiliary generator is an alternative supply of power that may be fueled by diesel oil or natural gas or some other fuel. The generator is used to absorb some of the building load during peak hours to reduce the base cost and to avoid the demand charge. The decision as to whether to use the generator will be governed by several major factors: marginal cost to operate the generator, Real Time Price (RTP) of purchased electricity, generator maximum operating-hour constraint governed by the state, and expectation of building loads, which will be explained later in the thesis.

3.1 Basic Generator Model

3.1.1 Assumptions

The basic model tries to capture the essence of the problem, but it does not represent all of an actual generator's behavior. The basic model takes into account the operating hour constraint, the capacity of the generator, and the efficiency of the generator, but not ramp-up, ramp-down, and start-up (transient behaviors).

3.1.2 Model

The objective of the basic generator model is to minimize costs for operation and maintenance:

$$Min \sum_{k=0}^{n} (p_k^{gen} \times u_k) \tag{3.1}$$

where the hourly generator price embodies the maintenance, the fuel costs, and the efficiency of the generator:

$$p_k^{gen} = \frac{p_{fuel} + M_{gen}}{\eta_{gen}} \quad \text{(efficiency constraints)} \tag{3.2}$$

Moreover, the generator is constrained by the limited capacity and the maximum allowable generator operating hours for each year.

$$u_k \le z_k C_{gen}$$

$$y_k \le gen\_max\_OH$$

The constraints:

$$z_k \leq z\_period$$

$$z_k \in \{0,1\}$$

seem to be redundant. However, this redundancy has a purpose as explained in Section 6.1.4.2. The generator model can be formed into linear programming statements in the basic main problem:

$$Min \sum_{k=0}^{n} P_k \bullet X_k \text{ ; which is equivalent to:}$$

$$Min \sum_{k=0}^{n} (p\_gen_k \times u_k + p\_rtp_k \times x_k) \tag{3.3}$$

$$\begin{aligned} \text{s.t.} \quad & x_k + u_k \geq L_k \\ & u_k \leq z_k C_{gen} \\ & y_k \leq gen\_max\_OH \\ & z_k \leq z\_period \\ & y_{k-1} + z_k = y_k \\ & z_k \in \{0,1\} \\ & x_k, u_k, y_k \geq 0 \end{aligned}$$

P = Cost vector

X = Variable vector to be optimized

p_gen_k = price of using the generator to generate electricity at time k (\$/kWh)

p_rtp_k = price of purchasing electricity from the utility company at time k (\$/kWh)

n = planning horizon (number of stages)

u_k = electricity generated at time k (kWh)

x_k = electricity purchased at time k (kWh)

L_k = building load at time k (kWh)

z_k = indicator whether generator is used (1) or not (0)

y_k = number of generator operating-hours used until time k

C_{gen} = maximum capacity of the generator (KWh)

gen_max_OH = generator operating-hour limit

z_period = the number of hours in one stage (for RTP, $z\_period = 1$)

p_{fuel} = price of generator's fuel (\$/KWh)

M_{gen} = generator's maintenance cost (\$/KWh)

η_{gen} = efficiency of the generator $\{0,\ldots,1\}$

3.2 Start-up, Shut-down, Ramp-up, and Ramp-down Constraints

In the actual implementation, a generator is constrained by some physical characteristics. The first one is the *Start-up phase* when the generator is first turned on. During this period the generator consumes fuel, but it does not produce any energy and thus takes away the otherwise useful fraction of the limited generator operating hours.

Once the generator is synchronized, it begins the second phase, the *Ramp-up phase*. In this period the generator starts contributing to the electrical load and the output capacity keeps increasing until it reaches its *Lower Operating Limit (LOL)*. After the second phase, the generator can then work up to its maximum capacity.

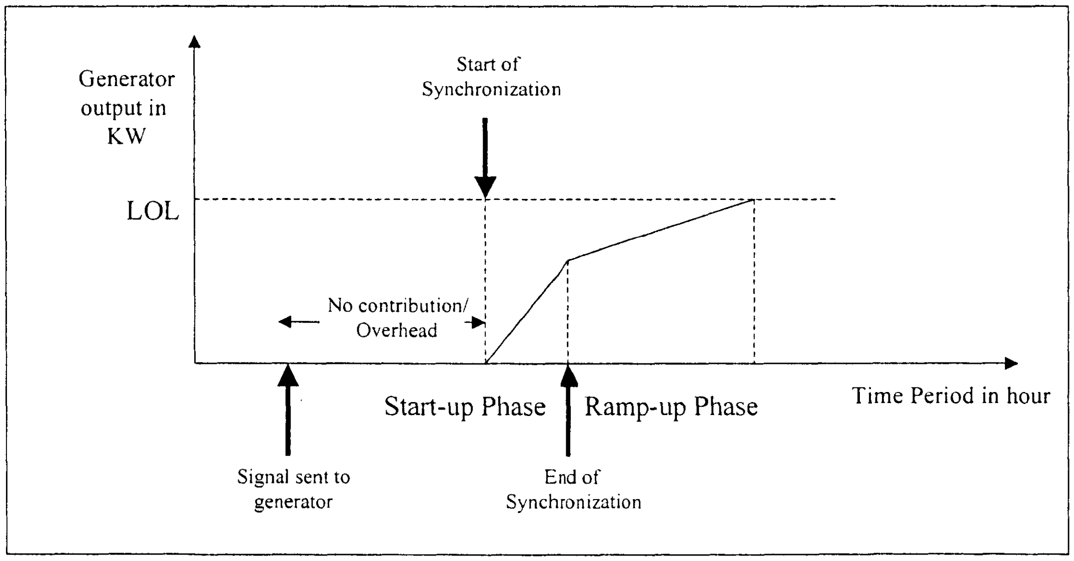


Figure 3.1 Generator Transition Behavior

The cycle repeats when the generator is switched off, it will go through the *Ramp-down phase* and then the *Shut-down phase*.

The total time for this transition period is less than one hour and depends on the size and the type of the generators used (Cogdell 1999). These transient phases take some portion of the allocated generator hour. Hence, it will be very inefficient if a generator is only used for an hour, since it is possible that the large portion of the time is used for this transition period. To avoid this inefficient generator-hour allocation, the constraint for the minimum contiguous generator-hour allocation (n) can be added into the problem formulation:

$$z_k \geq \max\{z_{k-1} - z_{k-2}, z_{k-2} - z_{k-3}, \ldots, z_{k-n+1} - z_{k-n}\} \tag{3.4}$$

The value of n needs to be studied further to find the optimal point between having a fragmented generator-hour allocation versus the contiguous one.

In addition, since the generator does not contribute the maximum power as assumed in the basic model, the adjustment on the power generated (u_k) needs to be made. This adjustment is critical if the generator is only used for the minimum generator operating-hour allocation (n), because the effect of generator transition behaviors will be significant. This adjusted-generator model will not be covered in the thesis and needs further study.

Chapter IV: Stochastic Programming with Recourse

4.1 Introduction

Stochastic Programming, also known as Dual Dynamic Programming (Pereira and Pinto 1991), combines Linear Programming with Dynamic Programming. It is based on an approximation of expected cost-to-go functions of Dynamic Programming in the form of piecewise linear functions. These approximate functions are obtained from dual solutions of the optimization problem at each stage and are equivalent to Bender's Decomposition cuts in linear programming. By using this method, state discretization, which is necessary in dynamic programming, is avoided, preventing the exponential explosion of the number of states.

A recourse problem occurs when a decision maker has to consider tradeoffs between cost of the '*here-and-now*' decision and the expected cost of the recourse decision. This problem is inherent in any stochastic problem with incomplete information where the decision maker has to choose a course of action that will meet his/her objectives and minimize his/her expected utility. The decision maker is said to have taken his/her recourse decision and move to the next stage after s/he has implemented the solution for the first stage problem.

4.2 Dynamic Programming Formulation of the Problem

It is assumed that the probability distribution function is known ahead of time (this will be the assumption throughout the thesis). The stochastic multi-stage case can be formulated in Dynamic Programming equation:

$$Q_t(X_t) = \min_{z_t \in Z_t(X_t)} \underset{w_t \in W_t}{E} \{g_t(L_t, z_t) + Q_{t+1}(X_{t+1})\} \tag{4.1}$$

state $X_t = \begin{bmatrix} y_t \\ L_t \end{bmatrix}$, and state transition: $X_{t+1} = \begin{bmatrix} y_{t+1} \\ L_{t+1} \end{bmatrix} = \begin{bmatrix} y_t + z_t \\ L_t + w_t \end{bmatrix}$

$Q_t(X_t)$ = the optimum expected cost at time t

$g_t(L_t, z_t)$ = given L_t and z_t determines u_t and thus, x_t; the cost function at time t then becomes:

$$(L_t - u_t) \times p\_rtp_t + u_t \times p\_gen$$

w_t = Independent random factor (the changes in building's load at time t)

L_t = the building's load at time t (Chapter 2)

u_t = the electricity generated at time t (Chapter 3)

p_rtp_t = the price of electricity purchased at time t (Chapter 3)

p_gen = the price of electricity generated at time t (Chapter 3)

y_t = number of generator operating hours used at time t (Chapter 3)

z_t = the control at time t (whether generating electricity z=1 or purchasing z=0, Chapter 3)

The main problem with dynamic programming is the explosion of states after several stages as explained in Chapter 1. Assume w_{max} is discretized into p components and thus there are p realizations at each stage. Moreover, if L_{max} is discretized into m states, and y_{max} into n state, then there will be $O(mn)$ possible states at each realization at each stage. Hence, there are $O((mnp)^T)$ subproblems (or possible paths) that need to be solved.

For a prototypical office space L_{max} =1000 KW, y_{max} =500hr, and w_{max} =500 KW will already create a gigantic optimization problem provided that the value is finely discretized. This states explosion is one of the main motivations behind studying the Stochastic Programming method to solve the problem. Using Stochastic Programming (Nested Bender Decomposition method), only w needs to be discretized; thus the total subproblems that need to be solved are $O(p^T)$.

4.3 Derivation of Dual Dynamic Programming

The concept of Dual Dynamic Programming in two stages is equivalent to the *Bender's Decomposition method*, and is illustrated with the linear programming form of the problem:

$$\begin{aligned} Min \quad & C_1 X_1 + C_2 X_2 (\xi) \\ s.t \quad & A_1 X_1 \leq b_1 \\ & A_2 X_2 (\xi) \leq b_2 (\xi) \\ & G_1 X_1 + F_2 X_2 (\xi) = d_2 \\ & X_1, X_2 (\xi) \geq 0 \end{aligned} \tag{4.1}$$

In general, the parameter ξ corresponds to the Stochastic Programming's random factor. In our case, the parameter ξ corresponds to different possible building loads at the second stage (Chapter 5). Here X_2 represents the second stage variable to be optimized. Vector b_2 has the stochastic elements, i.e. the building's load, and is also called the random factor.

The problem can then be broken up into a first-stage problem and a second stage problem:

$$\begin{aligned}
&\textit{First Stage:} && Min && C_1 X_1 + Q_2(X_{1,}\xi) && (4.2)\\
& && s.t && A_1 X_1 \le b_1 &&\\
& && && X_1 \ge 0 &&\\
&\textit{Second Stage:} && Q_2(X_1,\xi) = Min && C_2 X_2\ (\xi) &&\\
& && s.t && A_2 X_2(\xi) \le b_2\ (\xi) &&\\
& && && G_1 X_1 + F_2 X_2(\xi) = d_2 &&\\
& && && X_2\ (\xi) \ge 0 &&
\end{aligned}$$

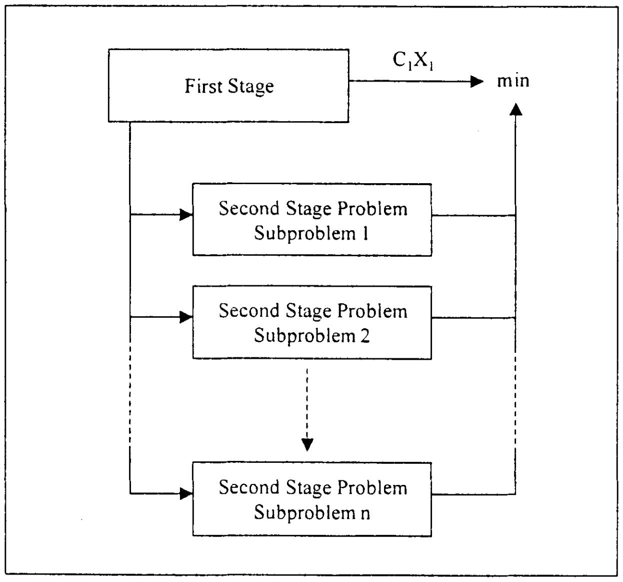


Figure 4.1 Diagram of Two-stage Stochastic Decision Process

In dynamic programming, $Q_2(X_1,\xi)$ represents the cost-to-go function. Here, Q_2 depends on the value of X_1 from the previous stage. Moreover, C_1X_1 is the present cost function. As we will see this dynamic programming structure will help in expanding the Bender's Decomposition to solve the multi-stage problem.

In order to transform the problem into an LP formulation that is solvable in the first stage, the dependency of the second stage problem's constraint on the first stage solution needs to be eliminated. Hence, the second stage problem can be transformed into its dual to move the decision variable X_1 up into the objective function.

$$Q_2(X_1,\xi) = \quad Max\ \pi_1(\xi)b_2(\xi) + \pi_2(d_2 - G_1X_1) \tag{4.3}$$
$$s.t. \quad \pi_1(\xi)A_2 + \pi_2(\xi)F_2 \le C_2$$

Let $\prod = \{\pi_i^1(\xi), \pi_i^2(\xi), \ldots, \pi_i^v(\xi)\}$ represent the set of all vertices of the dual constraint set. From the Resolution Theorem of Linear Programming (Bertsimas and Tsitsiklis 1997), the problem can be transformed into:

$$Q_2(X_1,\xi) = \quad Max\ \pi_1^i(\xi)\,b_2 + \pi_2^i(\xi)\ (d_2 - G_1X_1) \quad ; \forall i = 1,\ldots,v \tag{4.4}$$

Problem (4.4) can then be transformed into a Linear Programming problem:

$$Q_2(X_1,\xi) = \quad Min\ \alpha \tag{4.5}$$
$$s.t. \quad \alpha \ge \pi_1^1(\xi)\,b_2(\xi) + \pi_2^1(\xi)\,(d_2 - G_1X_1)$$
$$|$$
$$|$$
$$\alpha \ge \pi_1^v(\xi)\,b_2(\xi) + \pi_2^v(\xi)\,(d_2 - G_1X_1)$$

From the Duality Theorem in linear programming, the optimal costs of the dual problem and primal problem coincide (Bertsimas and Tsitsiklis 1997).. More importantly, the dual formulation (Equation 4.5) has shown that the cost to go function is a *piecewise linear function* of X_1 as will be proven later in this chapter.

Finally, the dynamic formulation of the first stage problem can be written in a linear programming form:

$$Q_1 = \text{Min } C_1X_1 + \sum_{\xi} p(\xi)Q_2(X_1,\xi) \tag{4.6}$$
$$s.t \quad A_1X_1 \le b_1$$
$$\pi_2^i(\xi)G_1X_1 + \theta \ge \pi_1^i(\xi)b_2 + \pi_2^i(\xi)d_2 \quad ; \forall(\xi); \forall i = 1,\ldots,v$$
$$\pi_1^i(\xi) \le 0 \quad ; \forall(\xi); \forall i = 1,\ldots,v$$

$p(\xi)$ = transition probabilities that correspond to different building loads

Since calculating all of the vertices is very difficult, only a subset of the vertices are going to be calculated. The chosen approach is to calculate the dual vertices from a set of trial values of X_1.

$$Q_2(X_1, \xi) = Min\ C_2 X_2\ (\xi) \tag{4.7}$$
$$s.t \qquad A_2 X_2(\xi) \leq b_2\ (\xi)$$
$$F_2 X_2(\xi) = d_2 - G_1 \hat{X}_1$$
$$X_2\ (\xi) \geq 0$$

It is worth noting that since problem 4.7 has only the subset of the actual problem constraints (Equation 4.5), the backward recursion using the dual variable as additional hyperplane constraints will result in a lower bound for the optimal cost. This property will be exploited later in the algorithm.

Without changing the result of the new hyperplane constraint, Equation 4.1 can be simplified into:

$$Min \quad C_1 X_1 + C_2 X_2\ (\xi) \tag{4.8}$$
$$s.t \quad A_1 X_1 \leq b_1$$
$$T_1 X_1 + W_2 X_2(\xi) \leq h_2(\xi)$$
$$X_1, X_2(\xi) \geq 0$$

With $T_1 = \begin{bmatrix} 0 \\ G_1 \end{bmatrix}$; $W_2 = \begin{bmatrix} A_2 \\ F_2 \end{bmatrix}$; $h_2(\xi) = \begin{bmatrix} \leq b_2(\xi) \\ = d_2 \end{bmatrix}$, and X_i is a vector of variables to be optimized.

The new hyperplane constraint can then be formed with equation:

$$E_1 X_1 + \alpha_1 \geq e_1(\xi) \tag{4.9}$$
$$E_1 = \sum_{\xi} p(\xi) \pi^i(\xi) T_1 \qquad ;\forall(\xi) \tag{4.10}$$
$$e_1 = \sum_{\xi} p(\xi) \{\pi_1^i(\xi) b_2(\xi) + \pi_2^i(\xi) d_2\} = \sum_{\xi} p(\xi) \pi^i(\xi) h_2(\xi) \quad ;\forall(\xi) \tag{4.11}$$

Matrix W_i is called the recourse matrix, and matrix T_i is called a *technology matrix* (Birge and Louveaux 1997). Our problem can be categorized as a *fixed recourse model.* A fixed recourse model is a stochastic programming with recourse that has a *recourse matrix* (W_i) fixed through all the stages. The fixed recourse model also has some basic properties, which will be discussed later.

This concept of Dual Dynamic Programming (Bender's Decomposition method) can be extended easily to solve multi-stage stochastic problems without fundamental changes as will be shown later in Chapter 7.

4.4 Basic Properties

The basic properties gave an insight into the problem, whether the problem is tractable or not. It is desirable to be able to check if a particular first-stage decision X_1 leads to a finite second-stage value without having to compute that value. To understand the problem's tractability, we need to know whether the solution space is bounded and well defined or not. The theorems and their proofs, which are explained in Chapter 3 of Birge and Louveaux (1997), will help answer the feasibility questions. Let us start with some definitions:

$$K_1 = \{ X_1 | A_1 X_1 = b_1, X_1 \geq 0 \}$$

$$K_2(\xi) = \{ X_1 | Q_2(X_2, \xi) < +\infty \}$$

$$K_2^P = \{ X_1 | \forall \xi \in \Xi, \exists X_2 \geq 0 \ s.t. \ W_2 X_2 = h_2 - T_1 X_1 \}$$

$$= \bigcap\nolimits_{\xi \in \Xi} K_2(\xi)$$

K_1 = the set determined by the fixed constraints, namely those that do not depend on the particular realizations of the random factor

$K_2(\xi)$ = second-stage feasibility sets that depends on the random factor ξ

K_2^P = defining the possibility interpretation of second-stage feasibility sets

Theorem 1 (Theorem 4, p.88, Birge and Louveaux 1997):

For a stochastic problem with fixed recourse, when W matrix is fixed and ξ has finite second moments:

a. $K_2(\xi)$ is closed and convex.

b. If T is fixed $K_2(\xi)$ is polyhedral.

Our problem has a fixed W matrix (fixed recourse problem), and the second moment of ξ is:

$$E[\xi^2] = \sum_{i=0}^{\max} L_i^2 p_i$$

where $0 \leq p_i \leq 1$ is a probability distribution of the building load in scenario i (KWh)

max = index of the maximum building load ever (KWh)

L_i = the building load in scenario i (KWh)

Since *max* and L are finite and p_i will have a Probability Mass Function (PMF) similar to Binomial PMF (depending on the external temperature PMF), the ξ has finite second moments. Thus, $K_2(\xi)$ of our problem is a closed and convex polyhedron.

Theorem 2 (Theorem 1, p.86, Birge and Louveaux 1997):

a. For each $\xi \in \Xi$, the elementary feasibility set is a closed convex polyhedron, hence, the set K_2^P is closed and convex.

b. When Ξ is finite, then K_2^P is also polyhedral and coincides with $K_2(\xi)$

From Theorem 1, we know that $K_2(\xi)$ of our problem is a closed and convex polyhedron satisfying the elementary feasibility set. ξ represents random factor (building load). Because building load is discretized (for example into very low, low, medium, high, very high) and has a finite range (finite maximum and has to be larger than 0), ξ is finite. Hence, Ξ is finite for our problem, which means K_2^P is also polyhedral and coincides with $K_2(\xi)$. In other words, this means that our problem will always have a solution space of a polyhedron.

Theorem 3 (Theorem 5, p.89, Birge and Louveaux 1997):

Assuming $Q_2(X_2, \xi)$ is not $-\infty$, for a stochastic program with fixed recourse, $Q_2(X_2, \xi)$ is :

a. a piecewise linear convex function in (h_2, T_1);

b. a piecewise linear concave function in C_2;

c. a piecewise linear convex function in X_1 for all X_1 in $K = K_1 \cap K_2$.

In our case, $Q_2(X_2, \xi) \geq 0$ since $C_2, X_2 \geq 0$. Thus, it is known beforehand that the cost function has a piecewise linear behavior relative to X_1. This result confirms our observation earlier in Section 4.2 that the cost to go function is a piecewise linear function.

The theorems are written in a two-stage problem context. However, since the multi-stage case is just a (recursively) nested form of the two-stage case, by induction, the theorems can be extended to cover the multi-stage case.

Chapter V: Two-Stage Solution

5.1 Introduction

The two-stage stochastic problem can be generalized with the formulation below:

$$
\begin{array}{lll}
Min & C_1 X_1 + \sum_{i=1}^{m} p^i (C_2^i X_2^i) & (4.2) \\
s.t & A_1 X_1 \geqslant b_1 & \\
 & T_1 X_1 + W_2^1 X_2^1 \geq h_2^1 & \\
 & T_1 X_1 + \qquad W_2^2 X_2^2 \geq h_2^2 & \\
 & \quad | & \\
 & \quad | & \\
 & T_1 X_1 + \qquad W_2^m X_2^m \geq h_2^m &
\end{array}
$$

As explained in the previous chapter we need to optimize the second stage problem before optimizing the first stage problem. We will evaluate all possibilities for the second stage and use the Bender's Decomposition method to recourse back to the first stage.

5.2 Problem Formulation

Equation 5.1 above can be translated into the linear programming statements of the problem below (Equation 5.2). The two-stage problem formulation is similar to the basic generator model (Section 3.1.2), the difference is that the formulation below captures the stochastic nature of the building load in the future. Here, $0 \leq p_i \leq 1$ is a probability distribution of the building load at scheme i (each path branches to m schemes). The variable definitions and notations are the same as those in Section 3.1.2.

$$Min\ (p\_gen_1 \times u_1 + p\_rtp_1 \times x_1) + \sum_{i=0}^{m} p^i (p\_gen_2^i \times u_2 + p\_rtp_2^i \times x_2) \qquad (5.1)$$

s.t. $x_1 + u_1 \geq L_1$

$u_1 \leq z_1 C_{gen}$

$y_k \leq gen\_max\_OH$

$z_1 \leq z\_period$

$-y_1 + z_1 = 0$

$x_2^i + u_2^i \geq L_2^i$; $\forall i = 1,...,m$ (Constraint 1)

$u_2^i - z_2^i C_{gen} \leq 0$; $\forall i = 1,...,m$ (Constraint 2)

$y_2^i \leq gen\_max\_OH$; $\forall i = 1,...,m$ (Constraint 3)

$z_2^i \leq z\_period$; $\forall i = 1,...,m$ (Constraint 4)

$y_1 - y_2^i + z_2^i = 0$; $\forall i = 1,...,m$ (Constraint 5)

$z_1, z_2^i \in \{0,1\}$; $\forall i = 1,...,m$

$x_1, u_1, y_1, x_2^i, u_2^i, y_2^i \geq 0$; $\forall i = 1,...,m$

m = number of scenarios

Constraints 1 to 4 make up constraint matrices $A_t X_t^i \leq b_t^i$ (refer to Section 4.3):

$$A_t = \begin{bmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -C \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } b_t^i = \begin{bmatrix} -L_2^i \\ 0 \\ OH \\ z\_period \end{bmatrix}$$

Constraint 5, which is the state transition constraint, forms constraint matrices

$F_t X_t^i \leq d_t^i - G_{t-1} X_{t-1}^j$ (index j represent the parent node of scenario i):

$F_t = \begin{bmatrix} 0 & 0 & -1 & 1 \end{bmatrix}$ and $G_{t-1} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$

Combining the two constraint matrices together:

W_2 represents the 'present' coefficient matrix: $\begin{bmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -C \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$;

T_1 (Technology Matrix) is represented by the coefficient matrix of the past stage problem:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix};$$

while h_2^i represents the right hand side matrix: $\begin{bmatrix} -L_2^i \\ 0 \\ OH \\ z\_period \\ 0 \end{bmatrix}$;

and $X_2^i = \begin{bmatrix} u_2^i \\ x_2^i \\ y_2^i \\ z_2^i \end{bmatrix}$

5.3 Algorithm

This two-stage algorithm is also known as *Bender's Decomposition method* or *Two-stage Stochastic Dual Dynamic Programming* (Pereira and Pinto 1991) with a heuristic method to find the vertices of the dual space.

In order to find the optimal vertices, the temporary optimal solution from the forward pass will be fed into the backward recursion to produce another set of temporary optimal vertices, which is used to build new constraints for the prior stage.

Step (a) Initialize: approximate future cost function, $Q_t(X_t, \xi)=0$

Set upper bound, $\theta_{upper} = \infty$

Set number of vertices $n=0$

Step (b) Solve the first stage problem:

$$Q_1 = \text{Min } C_1X_1 + Q_2(X_2, \xi) \qquad (5.2)$$

$$s.t \qquad A_1X_1 \le b_1$$

Problem (5.2) can be transformed to problem (4.5).

Let $\hat{X}_1$ be the optimal solution.

Step (c) Calculate the lower bound, $\theta_{lower} = Q_1$

If $error \ge \theta_{upper} - \theta_{lower}$ then stop else go to step (d)

Step (d) Solve second stage problem for each scenario (subproblem), i.e. find the optimal dual multipliers:

$$Q_2(X_2, \xi) = MinC_2X_2(\xi) \qquad (5.3)$$

$$s.t \qquad W_2^iX_2^i(\xi) \le h_2^i(\xi) - T_1\hat{X}_1$$

Let the dual multipliers from step (d) be $\pi^i(\xi)$;

Step (e) Increment the number of vertices $n=n+1$;

Construct the approximate cost function using the n vertices:

$$Q_2(X_2, \xi) = \quad Min\ \alpha \qquad (5.4)$$

$$s.t. \qquad \alpha \ge \pi_1^i(\xi)\, b_2(\xi) + \pi_2^i(\xi)\,(d_2 - G_1\hat{X}_1) \qquad ;\forall i = 1,\dots,n$$

The approximate cost function is the new constraint for Stage 1 (Equation 4.8 to 4.10)

Add this new constraint into the set of constraints of Stage 1 problem.

Step (f) $\theta_{upper} = C_1\hat{X}_1 + \sum_{\xi} p(\xi)(C_2X_2(\xi))$. Go to Step (b).

5.4 Sample Case

Example 5.4.1

Given:

Stage (Stg)	Scenario (Sub)	L (kW)	p_gen (\$/kWh)	p_rtp (\$/kWh)	OH	C_{gen}
1		500	1	1.5	1	500
2	1 (p_1=0.5)	450	1	2	1	500
	2 (p_2=0.5)	600	1	2	1	500

Table 5.1 Given Case of Example 5.4.1

Iter#	Step	Stg/Sub	X_{opt}'	Dual (P')	OptCost	Add. Constraint	error	$Q_{upper/lower}$
1	a		Initialize $Q_1(X_{1,}j)$ = 0 for all schemes					
	b	1	[500 0 1 1 0]		500			
	c						inf	Q_{lower}=500
	d	2-Sub1		[-2 -1 -500 -500]		E=[0 0 -500 0]		
		2-Sub2		[-2 -1 -500 -500]		e=0.5(400 + 700)		
						=550		
	e					Q_2-500y_1 >= 550		
	f							Q_{upper}=1500
2	b	1	[0 500 0 0 1]		1250			
	c						250	Q_{lower}=1250
	d	2-Sub1		[-2 -1 -500 -500]				
		2-Sub2		[-2 -1 -500 -500]				
	e					the same as above		
	f							Q_{upper}=1250
3	b						0	

Table 5.2 Step by Step Procedure Running the Algorithm

$Q_{lower} = \theta_{lower}$

$Q_{upper} = \theta_{upper}$

$$\text{Matrix } X = \begin{bmatrix} u_i \\ x_i \\ y_i \\ z_i \\ \alpha_i = Q_i \end{bmatrix}$$

The result can be verified with a calculation of the expected value:

Scheme 1 (allocate the generator hour at stage 1):

$$E[\alpha] = 500x1 + 0.5(900 + 1200) = 1550$$

Scheme 2 (allocate the generator hour at stage 2):

$$E[\alpha] = 500x1.5 + 0.5(450 + 600) = 1275*$$

The optimal solution is to allocate the generator hour at stage 2 with optimal cost of 1250.

5.5 Limitation on the Algorithm

Notice that problem (5.1) is a *mixed-integer programming problem (MIP)*. An integer programming problem does not have a dual solution. Thus, in our case we will solve the problem as a regular linear programming problem when we move backward in the algorithm, and treat it as a mixed-integer programming problem when we run the forward simulation.

If MIP solver is used during the forward pass, there is no guarantee that the solution will be optimal. So far during the simulations for small-stages problems, the algorithm with MIP in the forward pass always yields the optimal solutions. However, as the number of stages increases and the complexity of the problem increases, there are certain problem structures that could make the algorithm yield suboptimal solutions (Chapter 7).

Chapter VI: Multi-Stage Solution

6.1 Deterministic Case

Before exploring the Multi Stage Stochastic Problem, we will start with a simpler model of the Multi-Stage Deterministic case. The deterministic model of the basic generator model is analogous to a '*knapsack*' problem, whose goal is to maximize the value of a sack given a set of widgets with different values and weights to take and given a limited sack capacity (Bertsekas 1995). The knapsack problem is also an Integer Programming problem. The difference is in the state transition from one stage to another. The multi-stage formulation has the Technology matrix (T_i) as the link between one stage to the next. This link can be interpreted as the state transition equation in dynamic programming. In our particular problem, the state that links one stage to another is y_i, which is the number of generator hours used so far. This transition state is necessary for the problem to be able to be decomposed into independent stages, and still have the global constraints, the generator operating-hour limit (*gen_max_OH*).

6.1.1 Problem Formulation

In the deterministic case each stage only has one scenario:

$$
\begin{aligned}
Min \quad & C_1 X_1 + C_2 X_2 + C_3 X_3 + \ldots + C_T X_T \\
s.t \quad & A_1 X_1 \geq b_1 \\
& T_1 X_1 + W_2 X_2 \geq h_2 \\
& \qquad T_2 X_2 + W_3 X_3 \geq h_3 \\
& \qquad\qquad | \\
& \qquad\qquad | \\
& \qquad\qquad T_{T-1} X_{T-1} + W_T X_T \geq h_T
\end{aligned}
\qquad (6.1)
$$

The definition of T_i, X_i, W_i, and h_i are the same as explained in the previous chapter.

6.1.2 Algorithm

The algorithm is known as (the deterministic version of) *Nested Bender's Decomposition method* or *Dual Dynamic Programming* (Perreira and Pinto 1991). The termination procedure using the upper limit and lower limit is proposed by Pereira and Pinto (1991). It works analogous to the one for two-stage case, except the cost-to-go functions are calculated recursively from one stage to another in the forward and backward passes.

Step (a) Initialize: Let T be the planning horizon

Approximate future cost function, $Q_t(X_t)=0$ for $t=1,\dots,T$

Set upper bound, $\theta_{upper} = \infty$

Set number of vertices $n=0$

Step (b) Solve the first stage problem:

$$Q_1 = \text{Min } C_1X_1 + Q_2(X_2) \qquad (6.2)$$
$$s.t \quad A_1X_1 \leq b_1$$

Problem (5.2) can be transformed to problem (4.5).

Let $\hat{X}_1$ be the optimal solution.

Step (c) Calculate the lower bound, $\theta_{lower} = Q_1$

If $error \geq \theta_{upper} - \theta_{lower}$ then stop else go to step (d)

Step (d) Repeat *for $t=2,\dots,T$ (forward simulation)*

Solve the optimization problem for each stage t given a trial decision $\hat{X}_{t-1}$:

$$Q_t(X_t) = Min\ C_t X_t + Q_{t+1}(X_{t+1}) \qquad (6.3)$$
$$s.t \quad W_t X_t \geq h_t - T_{t-1} \hat{X}_{t-1}$$

Store the solution as $\hat{X}_t$

Step (e) Calculate the upper bound: $\theta_{upper} = \sum_{t=1}^{T} C_t \hat{X}_t$

Step (f) Repeat *for $t=T, T-1, \dots, 2$ (backward recursion)*

Solve the optimization problem for stage t, trial decision $\hat{X}_{t-1}$:

$$Q_t(X_t) = Min\ C_t X_t + Q_{t+1}(X_{t+1})$$

$$s.t \quad W_t X_t \geq h_t - T_{t-1} \hat{X}_{t-1}$$

Increment the number of vertices $n=n+1$;

Let the dual multipliers from step (f) be π_t'';

Step (g) Construct the additional constraints for the previous stage with n vertices:

$$Q_t(X_t) = Min\ \alpha_t \tag{6.4}$$

$$s.t. \quad \alpha_t \geq \pi_1^i b_t + \pi_2^i\ (d_t - G_{t-1} \hat{X}_{t-1}) \qquad ; \forall i = 1,...,n$$

The approximate cost function is the new constraint for Stage t-1 (Equation 4.8 to 4.10, without the random factor ξ)

Add this new constraint into the set of constraints of Stage t-1 problem.

Step (h) Go to Step (b).

6.1.3 Sample Cases

Example 6.1.1:

This is a very simple example where the optimal cost is obvious, to allocate the generator hour at the peak hour (at Stage 3).

Stage (Stg)	L (kW)	p_gen ($/kWh)	p_rtp ($/kWh)	OH	C_{gen}
1	200	1	1.5	1	300
2	400	1	2	1	300
3	300	1	3	1	300

Table 6.1 Given Case of Example 6.1.1

$Q_{lower} = \theta_{lower}$

$Q_{upper} = \theta_{upper}$

$$\text{Matrix } X = \begin{bmatrix} u_i \\ x_i \\ y_i \\ z_i \\ \alpha_i = Q_i \end{bmatrix}$$

Iter#	Step	Stg	X_{opt}'	Dual (P')	OptCost	New Constraint	err	$Q_{upper/lower}$
1	a		Initialize $Q_1(X_1,j) = 0$ for all schemes					
	b	1	[200 0 1 1 0]		200			
	c						inf	Q_{lower}=200
	d	2	[0 400 1 0 0]		800			
		3	[0 300 1 0 0]		900			
	e							Q_{upper}=1900
	f	3		[-3 -2 -600 -600]		E_2=[0 0 -600 0 0]		
						e_2=-300x-3-600x1		
						e_2=300		
						Q_3-600y_2 >= 300		
		2		[-2 -1 0 -1 -600]		E_1=[0 0 –600 0 0]		
						e_1=-400x-2-1x-300		
						e_1=1100		
						Q_2-600y_1 >= 1100		
2	b	1	[0 200 0 0 1100]		1400			Q_{lower}=1400
	c						500	
	d	2	[0 400 0 0 300]		1100			
		3	[300 0 1 1 0]		300			
	e							Q_{upper}=1400
	f	3		[-1 0 0 0]		E_2=[0 0 0 0 0]		
						e_2=300		
						Q_3 >= 300		
		2		[-2 -1 0 -1 0]		E_1=[0 0 0 0 0]		
						e_1=1100		
						Q_2 >= 1100		
3	b	1	[0 200 0 0 1100]		1400			Q_{lower}=1400
	c						0	

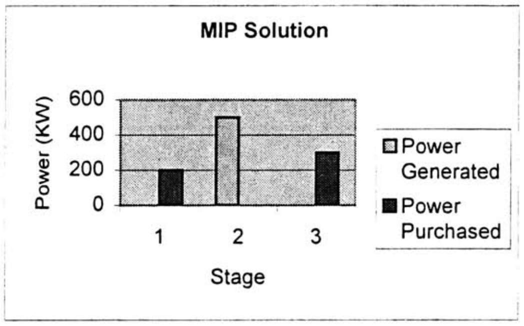
Table 6.2 Step by Step Procedure Running the Algorithm of Example 6.1.1

Example 6.1.2:

Stage (Stg)	L (kW)	p_gen (\$/kWh)	p_rtp (\$/kWh)	OH	C_{gen}
1	200	1	1.5	1	500
2	500	1	2.5	1	500
3	300	1	3	1	500

Table 6.3 Given Case of Example 6.1.2

In this second example the generator capacity (C_{gen}) is larger than all building loads, and thus, the optimal solution is not just allocating the generator operating-hour to the peak RTP charge hour like in Example 6.1.1. As the consequence, the second example has a more complex structure than the first, in the sense that the LP solution and the MIP solution are different (figure 6.1, figure 6.2).



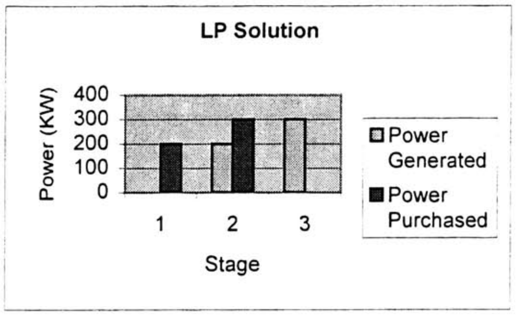


Figure 6.1 MIP Solution of Example 6.1.2 Figure 6.2 LP solution of Example 6.1.2

Iter#	Step	Stg	X_{opt}'	Dual (P')	OptCost	New Constraint	err	$Q_{upper/lower}$
1	a		Initialize $Q_1(X_1,j) = 0$ for all schemes					
	b	1	[200 0 1 1 0]		200			
	c						inf	$Q_{lower}=200$
	d	2	[0 500 1 0 0]		1250			
		3	[0 300 1 0 0]		900			
	e							$Q_{upper}=2250$
	f	3		[-3 -2 -1000 -1000]		E_2=[0 0 -1000 0 0]		
						$e_2=-300x-3-1000x1$		
						$e_2=-100$		
						$Q_3-1000y_2 >= -100$		
		2		[-2.5 -2 0 -1 -1000]		E_1=[0 0 -1000 0 0]		
						$e_1=-500x-2.5-1x100$		
						$e_1=1150$		
						$Q_2-1000y_1 >= 1150$		
2	b	1	[0 200 0 0 1150]		1450			
	c						800	$Q_{lower}=1450$

	d	2	[0 500 0 0 0]		1250			
		3	[300 0 1 1 0]		300			
	e							Q_{upper}=1850
	f	3		[-1 0 0 0]		E_2=[0 0 0 0 0]		
						e_2=300		
						Q_3 >= 300		
		2		[-2.5 -1.5 0 -0.75 -0.25 -750]		E_1=[0 0 –750 0 0]		
						e_1=1250-75+75		
						e_1=1250		
						Q_2-750y_1 >= 1250		
3	b	1	[0 200 0 0 1250]		1550			
	c						300	Q_{lower}=1550
	d	2	[500 0 1 1 900]		1400			
		3	[0 300 1 0 0]		900			
	e							Q_{upper}=1700
	f	3		[-3 -2 -1000 -1000]		E_2=[0 0 -1000 0 0]		
						e_2=-300x-3-1000x1		
						e_2=-100		
						Q_3-1000y_2 >= -100		
		2		[-2.5 -1.5 0 -0.75 -0.25 -750]		E_1=[0 0 -750 0 0]		
						e_1=1250-75+75		
						e_1=1250		
						Q_2-750y_1 >= 1250		
4	b	1	[0 200 0 0 1400]		1550			
	c						0	Q_{lower}=1550

Table 6.4 Step by Step Procedure Running the Algorithm of Example 6.1.2

This difference in solution is due to our LP formulation of the problem (Chapter 5). The optimal solution for the constraint:

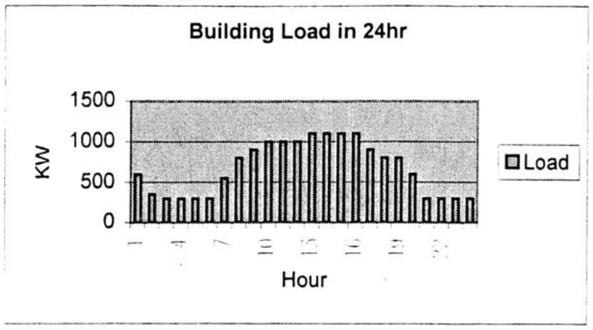
$$u_t - z_t C_{gen} \leq 0$$

is to use the generator up to its maximum capacity for each generator operating-hour. This problem does not happen in the MIP formulation where *z* is constraint to be either 1 or 0 at each stage and avoid a portion of *z-value* to be carried over to the next stage. This side effect of the LP formulation skewed the interpretation of the problem where the generator capacity functions just like the generator operating-hour. More importantly, this example shows the importance of the constraint:

$$z_t \in \{0,1\}$$

Hence, running LP-only formulation may give misleading results. The difference between MIP solution and LP solution consequences will be covered in Chapter 7.

Example 6.1.3



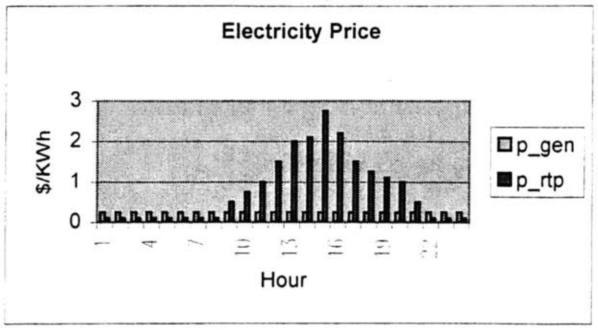


Figure 6.3 Load Profile in Example 6.1.3

Figure 6.4 Electricity Price Profile in Example 6.1.3

In this example, a 24-hour/stages planning horizon was run (figure 6.3 and 6.4). The answer given is to allocate the generator operating hours at the peak hours (Stage 13-16). The algorithm yields the expected answer (figure 6.5), and it converges in five iterations.

Generator Capacity	600kW
Operating Hour Limit	4hr
Cost without generator	$18,075.00
Optimized Cost	$13,245.00
Saving	**$ 4,830.00**

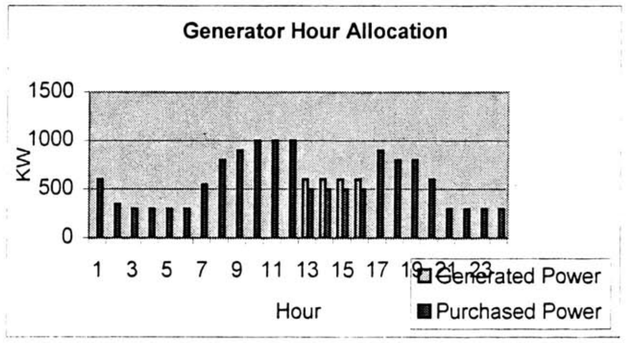


Figure 6.5 Optimal Dispatch Profile of Example 6.1.3

6.1.4 Analysis for the Deterministic Solution

6.1.4.1 Generated Cuts in the Objective Function

To illustrate how the method works and to provide insight into performance of the Nested Bender's Decomposition method, the behavior of the first stage objective function will be studied more closely. The objective function of the first stage (Q_1) of Example 6.1.2 can be written as a function of z_1:

Iteration 0 :

$$Q_1^0 = p\_gen * z_1 \min\{L_1, C\} + p\_rtp_1 * z_1 \max\{L_1 - z_1 \min\{L_1, C\}, 0\}$$
$$+ p\_rtp_1(1 - z_1)L_1 + Q_2 \qquad (6.5)$$

Q_1^i = optimal cost at Stage 1 iteration I

Q_2 = the cost-to-go function

In Equation 6.5, the left multiplication is to calculate the cost if a generator is used, the middle one is to calculate the cost if the capacity of the generator is less than the load and thus the facility is forced to buy some electricity form the grid, and the one on the right is to calculate the cost of purchasing all electricity from the grid. Moreover, at iteration 0, the cost-to-go function is zero.

Iteration 1 : $1000y_1 + 1150 \le Q_2$ *(the first cut)* (6.6)

$$Q_1^1 \ge Q_1^0 + 1000y_1 + 1150 = Q_1^0 + 1000z_1 + 1150 \qquad (6.7)$$

Equation 6.6 is the cut at iteration 1, which was produced by the backward recursion of Stage 2 (table 6.4). Stage 1 is the initial stage, thus $z_1 = y_1$. Hence, y_1 in Equation 6.6 can be replaced with z_1. Moreover, the cost-to-go function (Q_2) in Equation 6.5 can be substituted with the cost-to-go function in Equation 6.6 to form the new objective cost function (Equation 6.7). At the next iteration, the same procedure happens (Equation 6.8 and 6.9).

Iteration 2 : $750y_1 + 1250 \le Q_2$ *(the second cut)* (6.8)

$$Q_1^2 \ge Q_1^0 + 750y_1 + 1250 = Q_1^0 + 750z_1 + 1250 \qquad (6.9)$$

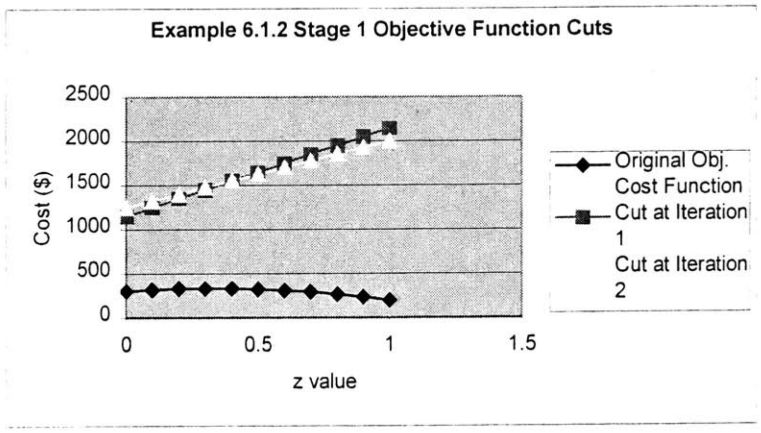


Figure 6.6 Objective Function Cuts at Stage 1 of Example 6.1.2

From figure 6.6, our objective cost function is a nonlinear function of z_1 due to the quadratic original cost function (Equation 6.5). The algorithm can be extended to the nonlinear cases (Perreira and Pinto 1989). Moreover, since the objective function (relative to z_1) is concave and our problem is a minimization problem, the nonlinearity, in this case, does not affect the result; the optimal solution is always be at one of the endpoints (0 or 1).

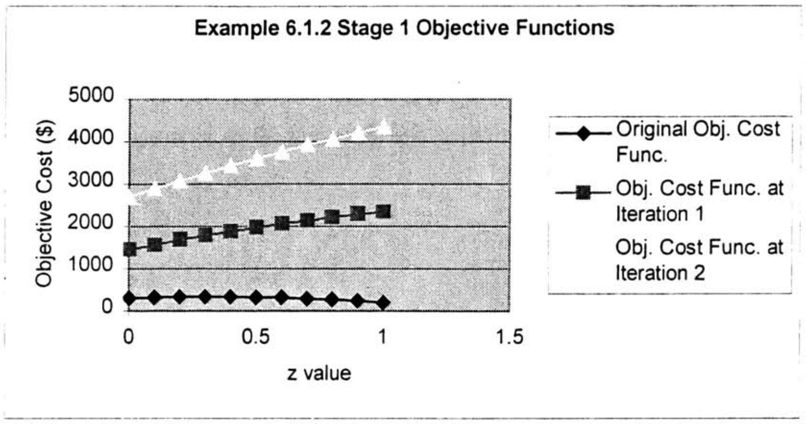


Figure 6.7 Stage 1 Objective Functions of Example 6.1.2 at Different Iterations

As seen in figure 6.7, in iteration 1 it is more attractive to allocate the generator hour at Stage 1. However, in iteration 1, the new cut causes the allocation of the generator hour at Stage 1 to be more costly. Furthermore, the cut in iteration 3 confirms the previous cut. In iteration 3 the solution becomes stable and thus, no more cuts are generated. Hence, the algorithm gives a recommendation of not using the generator operating hour at Stage 1.

6.1.4.2 Correctness

The correctness of the LP-version of the algorithm is presumed from the proof of Bender's Decomposition method and from tests done on the algorithm (Perreira and Pinto 1991, Birge and Louveraux 1997). There is no guarantee that the MIP-version algorithm will always achieve optimal solution as will be shown in the next Chapter.

Another important aspect of correctness is the resemblance between the LP formulation and the MIP formulation. From Equation 3.3, there are redundant constraints:

$$z_k \leq z\_period \qquad (6.10)$$

$$z_k \in \{0,1\} \qquad (6.11)$$

In MIP, Equation 6.11 is sufficient without Equation 6.10. However, the purpose of this redundancy is to make the LP terms resemble those of MIP as much as possible, so the MIP solution and LP solution does not differ (the effect of MIP and LP solutions being different will be explained in greater detail in the next chapter). Significant error can occur if constraint in Equation 6.10 is taken out (figure 6.8). What happen is that the LP solution from the backward pass does not resemble the MIP (allows allocation of more than *z=1* at each hour) (figure 6.9) and thus, causes the algorithm to yield the wrong objective cost function cuts, which are used in the forward pass. The optimal solution is achieved after adding the constraint in Equation 6.10 and 6.11 (figure 6.10).

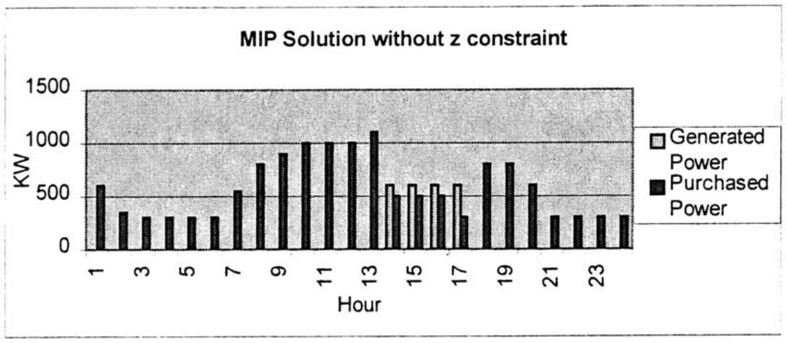


Figure 6.8 MIP Solution w/o z constraint of Example 6.1.3

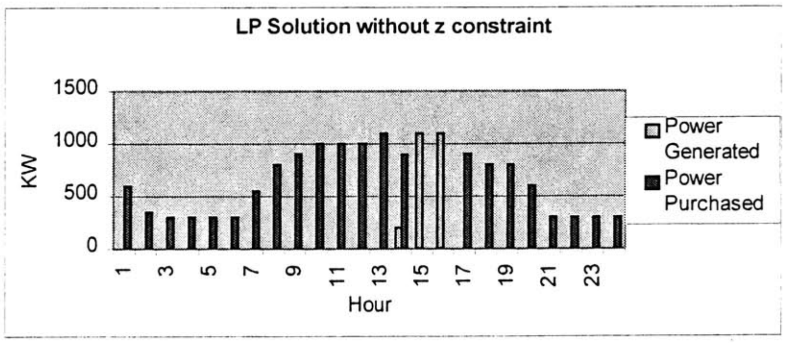


Figure 6.9 LP Solution w/o z constraint of Example 6.1.3

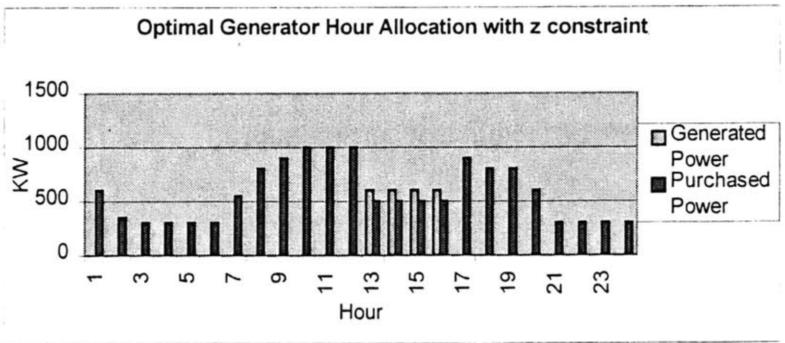


Figure 6.10 Optimal Solution of Example 6.1.3

6.1.4.3 Convergence

Since the problem is deterministic, the algorithm will achieve a stable solution in some finite number of iterations. The number of iterations depends on the complexity of the problem, which is determined by several factors: the dimension of the variables, the number of constraints, the structure of the problem, and the number of stages. The complexity of the problem governs the number of vertices of the dual space solution and also the way the algorithm traverses through the vertices before arriving at the optimal solution. Hence, in the worst case, the number of iterations is equal to the number of vertices of the dual space (v).

Example 6.1.2 has a more complex problem structure (the LP solution and the MIP solution are different) than Example 6.1.1, and thus the algorithm takes more iterations to achieve the optimal solution in Example 6.1.2 than in Example 6.1.1. Moreover, the upper bound, which is calculated in the forward pass, and the lower bound, which is calculated in the backward pass from the dual space, will never coincide (figure 6.11 and 6.12), because the upper bound results in the optimal MIP value, while the lower bound results in the optimal LP value. Moreover, the optimal LP value will always lower than optimal MIP value (Bertsimas and Tsitsiklis 1997).

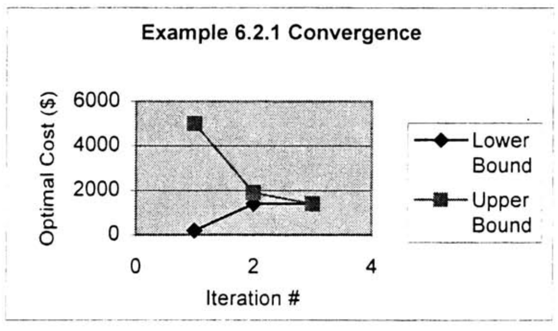


Figure 6.11 Convergence of Example 6.1.1

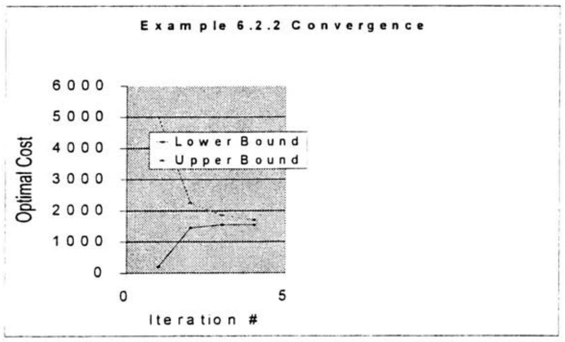


Figure 6.12 Convergence of Example 6.1.2

6.1.4.4 Computational Efficiency

The running time in the deterministic case algorithm depends on the number of iterations (*iteration*), the number of stages (T), and CPLEX's Branch and Bound algorithm running time (*mipopt*). Further, *mipopt* depends on the size of the problem (number of columns and number of rows of the problem matrices). The number of columns (*NUMCOLS*) represents the dimension of the problem space, while the number of rows (*NUMROWS*) represents the number of constraints. Both of them affect the number of vertices (v) and thus, the number of iterations.

Unlike an LP problem, our problem is a Zero One (Mixed) Integer Problem (*ZOIP*), which is part of the NP-hard family (Bertsimas and Tsitsiklis 1997). The problem is considered as ZOIP, because at each stage the control decision is to whether to allocate the generator hour ($z=1$) or not ($z=0$). NP-hard is the complexity for a problem that shows strong evidence of not being polynomially solvable. The computational efficiency of the algorithm:

	Deterministic Algorithm
Best Case	*O(T*mipopt)*
Worst Case	*O(v*T*mipopt)*
Memory	*O(T*NUMROWS*NUMCOLS)*

Table 6.5 Deterministic Algorithm Computational Performance

There are other algorithms that can solve this deterministic problem faster than the Nested Bender's Decomposition method. For example, Dynamic Programming can solve this type of knapsack problem in $O(T^2 * C_{max})$ (C_{max} = the maximum from all possible *p_rtp* and *p_gen* values) (Bertsimas and Tsitsiklis 1997). However, as the complexity of the problems increases, Dynamic Programming is impractical.

The relationship between *T*, *iteration*, and *running time* are shown below (figure 6.9, 6.10). All simulations are done in a PC with AMD K-6 200 MHz processor and 64 Mbyte of RAM.

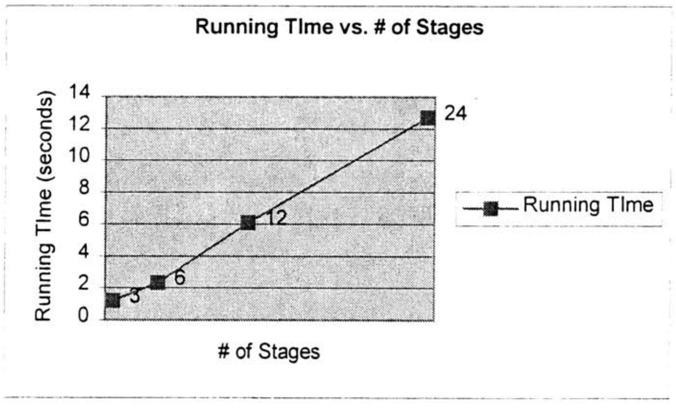


Figure 6.13 Running Time vs. Number of Stages of Example 6.1.3

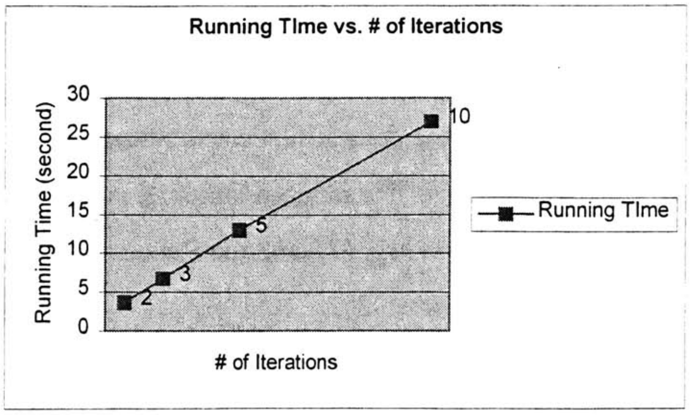


Figure 6.14 Running Time vs. Number of Iterations of Example 6.1.3

The results verify the computational efficiencies stated above. The graphs (figure 6.13 and 6.14) show the positive linear relationships between the running time and number of stages and between the running time and number of iterations.

6.2 Stochastic Case

6.2.1 Problem Formulation

The stochastic multi-stage problem is an extension of the deterministic multi-stage problem with an added complication of the nested subproblems. The random factor, ξ is the building load (L_i), which is an element of h_i. The Technology matrix (T_i) and the Recourse matrix (W_i) are fixed for all stages and scenarios (refer to Section 5.2).

$$\begin{aligned}
&Min \quad C_1X_1 + (\sum_{i=1}^{n} p_2^i C_2^i X_2^i + (\sum_{i=1}^{n^2} p_3^i C_3^i X_3^i + (\ldots + (\sum_{i=1}^{n^{T-1}} p_T^i C_T^i X_T^i)))) && (6.12)\\
&s.t \quad A_1X_1 \geq b_1 &&\\
&\quad T_1^j X_1^j + W_2^i X_2^i \geq h_2^i && ;\ \forall i = 1,\ldots,n\\
&\qquad T_2^j X_2^j + W_3^i X_3^i \geq h_3^i && ;\ \forall i = 1,\ldots,n^2\\
&\qquad\qquad | &&\\
&\qquad\qquad | &&\\
&\qquad\qquad T_{T-1}^j X_{T-1}^j + W_T^i X_T^i \geq h_T^i && ;\ \forall i = 1,\ldots,n^{T-1}
\end{aligned}$$

Note: j is the parent index of i from the previous stage

6.2.2 Algorithm

The algorithm used is known as *Nested Bender's Decomposition method/Nested L-Shaped method* (Birge and Louveaux 1997) or *Stochastic Dual Dynamic Programming* (Pereira and Pinto 1991). Since the combinations of scenarios increase exponentially with the stage and become unmanageable after a short period, a heuristic method of sampling is used. *Monte Carlo Simulation* is used in the algorithm to set trial decisions X_i, and to reduce the problem into a tractable one.

```
Step (a) Find a set of trial decisions: $\{ \hat{X}_t^i$ for $i=1,\dots,n^{t-1}$; $t=1,\dots,T\}$
        $i$ = index of the trial decision (number of trial decisions = number of samples)
        $j$ = index of children under trial decision $\hat{X}_t^i$, where $s$ = number of children
Step (b) Repeat for $t = T, T-1, \dots, 2$ (backward recursion)
        Repeat for each trial decision $\hat{X}_t^i$, $i=1,\dots,n$
            Repeat for each scenario under trial decision $i$, $h_t^{i,j}$, $j=1,\dots,s$
                Solve the optimization problem for $t$ given $h_t^{i,j}$ and trial decision $\hat{X}_t^i$
                    $Min\ \{ C_t X_t + Q_{t+1}(X_{t+1}, \xi) \}$        (6.13)
                    $s.t \quad W_t X_t^{i,j} \ge h_t^{i,j} - T_{t-1} \hat{X}_{t-1}^i$
                Let $\pi_t^{i,j}$ be the optimal dual multiplier of problem (6.7) and save it.
            Use the saved dual multipliers (there are $m$ of them) to construct an additional
            hyperplane of the approximate expected future cost function for Stage $t-1$
            (derivation from Equation 4.8 to 4.10):
                $Q_t(X_t, \xi) = Min\ \alpha_t^i$        (6.14)
                $s.t.\ \alpha_t^i \ge \sum_{j=1}^{m} p_t^{i,j} \pi_t^{i,j} h_t^{i,j} - \sum_{j=1}^{m} p_t^{i,j} \pi_t^{i,j} T_{t-1} X_{t-1}^i$
            In Equation 6.10, the dual multipliers are weighed based on their probabilities.
            The approximate cost-to-go function for Stage $t-1$ can then be written as:
                $E_{t-1}^i X_{t-1}^i + \alpha_t^i \ge e_t^i$        (6.15)
            Add this new constraint into the set of constraints of Stage $t-1$ problem.
Step (c) Go to Step (a)
```

In order to find the set of trial decisions in Step (a), Monte Carlo Simulation is used, and the steps are described below.

```
Step (a) Initialize for the very first time (afterward go to Step (b) directly):
        Let $T$ be the planning horizon
        Approximate future cost function, $Q_t(X_t, \xi)=0$ for $t=1,\dots,T$
        Set upper bound, $\theta_{upper} = \infty$
        Set number of vertices $n=0$
Step (b) Solve the first stage problem
        $Q_1 = \text{Min } C_1X_1 + Q_2(X_2, \xi)$                                  (6.16)
        $s.t \quad A_1X_1 \geq b_1$
    Let $\hat{X}_1$ be the optimal solution
    Let $\theta_{lower} = Q_1$
    Initialize $X_1^i = \hat{X}_1 ;\forall i = 1,\dots,n$
Step (c) Repeat for $t=2,\dots,T$
    Repeat for $i=1,\dots,n$
        Sample a vector $h_t^{i,j}$ from the set $\{h_t^{i,j}; j=1,\dots,s\}$
        Solve the optimization problem for stage $t$, sample $i$:
            $Min\ \{ C_t X_t + Q_{t+1}(X_{t+1}, \xi) \}$                            (6.17)
            $s.t \quad W_t X_t^{i,j} \geq h_t^{i,j} - T_{t-1} \hat{X}_{t-1}^i$
        Store the optimal solution as $\hat{X}_t^i$
Step (d) Calculate Upper Bound
    The upper bound is estimated from the Monte Carlo simulation results for all stages and
    scenarios at this iteration:
    $\theta_{upper} = C_1X_1 + \sum_{t=2}^{T} \frac{1}{n} \sum_{i=0}^{n-1} C_t^i X_t^i$                  (6.18)
```

6.2.3 Sample Cases

Example 6.2.1:

Stage (Stg)	Scenario	L (kW)	p_gen ($/kWh)	p_rtp ($/kWh)	OH	C_{gen}
1		300	1	2	1	500
2	1 (p_1=0.5)	400	1	2.8	1	500
	2 (p_2=0.5)	510	1	2.8	1	500
3	1 (p_1=0.25)	300	1	3	1	500
	2 (p_2=0.25)	450	1	3	1	500
	3 (p_3=0.25)	350	1	3	1	500
	4 (p_4=0.25)	550	1	3	1	500

Table 6.6 Case of Example 6.2.1

This is a problem where each internal node branches into two scenarios (number of children = *num_of_children* = 2). The solution is not to allocate the generator operating hour (the limit, *gen_max_OH* = 1, in this case) at the first stage (present hour), but to allocate it at the second stage if scenario 2 happens, otherwise allocate it at the third stage.

Stage (Stg)	Scenario	Decision to use OH
1		0
2	1	0
	2	1
3	1	1
	2	1
	3	0
	4	0

Table 6.7 Solution of Example 6.2.1

Table 6.7 shows the typical Dynamic Programming output when all the possible paths are evaluated. The output interpretation will be different if sampling is done (sampling is done by Monte Carlo Simulation whose algorithm is shown in Section 6.2.2).

This example illustrates the strength of the Stochastic Programming where the method can give different solutions depending on what path will later be chosen (similar to Dynamic Programming solutions). As we shall see later, this method yields a better optimal solution and cost than a deterministic optimization (Section 6.2.4.1).

Example 6.2.2

In this example, we assume a binomial building load transition Probabilities Mass Function (PMF) from one stage to another (different scenarios), instead of a uniform one as in the previous example. It is worth noted that the building load PMF is different than the external temperature PMF (Section 2.4).

i/j	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800
100	0.5	0.5													
150	0.25	0.5	0.25												
200		0.25	0.5	0.25											
250			0.25	0.5	0.25										
300				0.25	0.5	0.25									
350					0.25	0.5	0.25								
400						0.25	0.5	0.25							
450							0.25	0.5	0.25						
500								0.25	0.5	0.25					
550									0.25	0.5	0.25				
600										0.25	0.5	0.25			
650											0.25	0.5	0.25		
700												0.25	0.5	0.25	
750													0.25	0.5	0.25
800														0.5	0.5

Table 6.8 Building Load PMF of Example 6.2.2

i = the current building load from weather (kW)

j = the future building load from weather (kW)

The building load is now divided into two parts: Occupation Load and Weather Load (Chapter 2) (figure 6.15). In this example, the expected occupation loads are used. From the binomial PMF in Table 6.8, the expected weather load at all stages will be equal to the weather load initial value, in this case 600 KW.

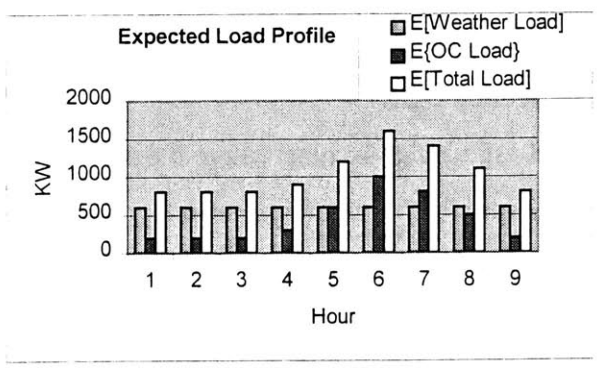


Figure 6.15 Example 6.2.2 Expected Load Profile

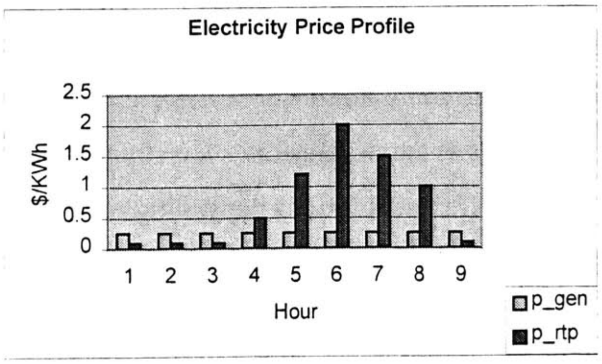


Figure 6.16 Example 6.2.2 Electricity Price Profile

Since there are 6561 possible paths, sampling is necessary (figure 6.17 and 6.18). Monte Carlo Simulation is used for sampling, and the algorithm is shown in Section 6.2.2.

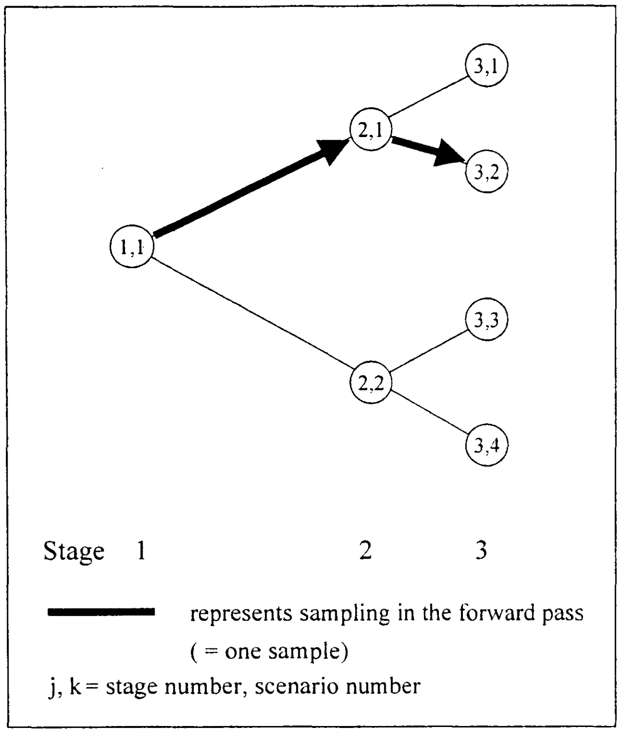

3,1
2,1
3,2
1,1
3,3
2,2
3,4
Stage 1 2 3
represents sampling in the forward pass (= one sample)
j, k = stage number, scenario number

Figure 6.17 Sampling of in the forward pass (= 1 sample)

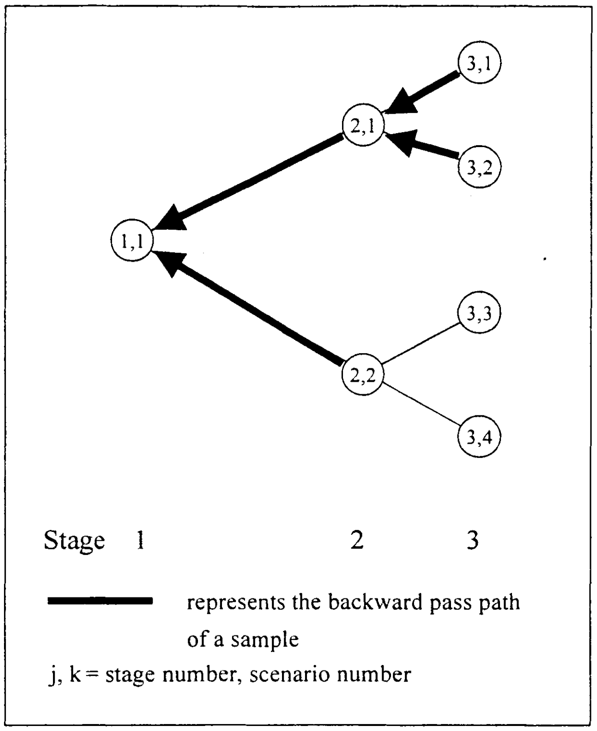
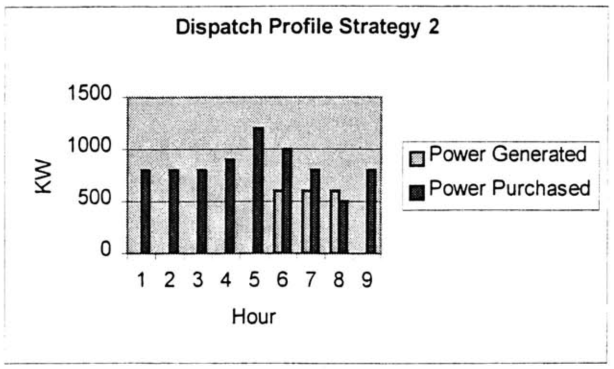

3,1
2,1
3,2
1,1
3,3
2,2
3,4
Stage 1 2 3
represents the backward pass path of a sample
j, k = stage number, scenario number

Figure 6.18 Backward pass of a sample

Generator Capacity	600kW
Operating Hour Limit	3hr
Number of Samples	50
Number of Iterations	10
Running Time	762.14seconds
Initial Weather Load	600kW

It is clear by looking at the expected load profile and the price profile that the optimal strategy, for most scenarios, would be to allocate the generator operating hours at hour 5, 6, and 7 (Strategy 1) (figure 6.19). This expected optimal result for most scenarios comes from finding the maximum of the product of RTP charge and expected total building load. However, for some paths, it is possible that the optimal strategy is to allocate the generator operating hours at hour 6, 7, and 8 (Strategy 2) (figure 6.20).



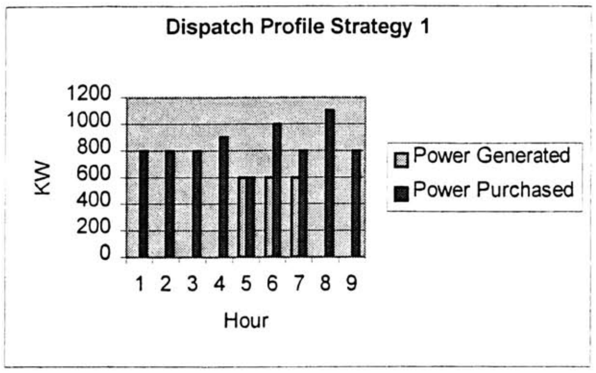


Figure 6.19 Example 6.2.2 Strategy 2

Figure 6.20 Example 6.2.2 Strategy 1

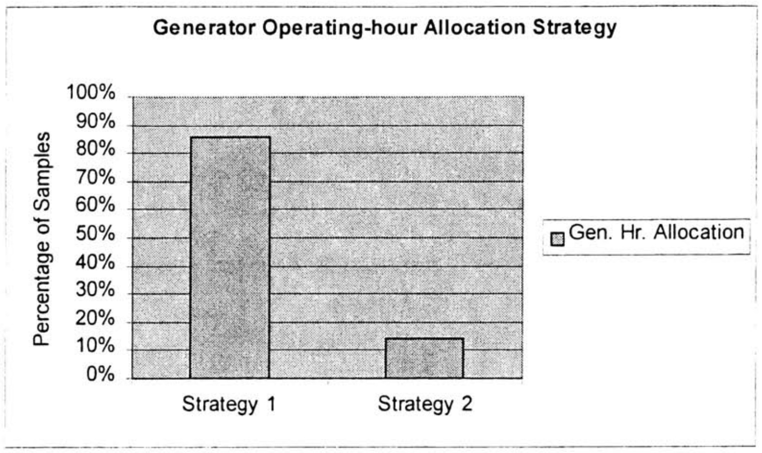


Figure 6.21 Example 6.2.2 Sampling Solution

The algorithm yields a result that verifies our expectation, 86% of the samples recommend strategy 1 and only 14% recommend strategy 2 (figure 6.21). Intuitively, this result can be interpreted as saying that applying strategy 1 will give the optimal result roughly 86% of the times. This percentage figure is not the exact error standard deviation measure (the measurement of the sampling standard deviation will be discussed in Section 6.2.4.2).

The main complication of this sampling approach is verifying and reading the results. Not all of the samples will yield correct results for two reasons. The first one, due to the random nature of sampling, is the possibility that at least one of the nodes in the sample has not been touched or has never been selected as a sample in the previous iterations.

The second reason is the possibility that one of the nodes in one of the samples has not been touched /sampled enough for it to receive enough cutting planes to yield the correct decision. As shown in figure 6.11 and 6.12, for the three stage deterministic problem (each sample can be thought as one deterministic problem), three to four iterations are needed to achieve the optimal result. This means, in the current example, that each sample has to be touched more than four times (since this example have a longer planning period).

This random incorrectness in the results will be referred later as the *sampling effect*. This sampling effect is unpredictable, meaning it is almost impossible to tell when and where the errors are going to occur or how big they will be. Hence, the best way to read the results is by using the statistical approach, as shown in figure 6.19 and as will be explained further in Section 6.2.4.2.

6.2.4 Analysis for the Stochastic Solution

6.2.4.1 Value of Information

As we can see from the solution in Example 6.2.1, the Stochastic Programming algorithm behaves just like the Dynamic Programming algorithm, in that it adopts a "*Wait and See*" policy (closed-loop control) instead of a "*Here and Now*" policy (open-loop control). An example of an open-loop control algorithm is writing all the subproblems into one big linear programming formulation and optimizing simultaneously, rather than decomposing the problem into stages and scenarios.

A closed-loop control yields a better optimal solution than an open-loop control. The difference between the two optimal costs is called the *value of information* (Bertsekas 1995; Birge and Louveraux 1997). The value of information in Example 6.2.1 is *$9*, as calculated from the result below. Moreover, figure 6.22 explains the definition of *scheme*, which is used in the calculation of the value of information.

Closed-Loop Solution				
	stage 3	stage 2	stage1	
Scheme1	use@3-1 2095	use@2-1 2125	add@1 2811.5	Opt. Cost **2277.5**
Scheme2	use@3-2 2478	use@2-2 2460		
Open-Loop Solution				
	stage 3	stage 2	stage 1	
	use@3 2286.5	use@2 2292.5	use@1 2811.5	Opt. Cost 2286.5

Table 6.9 Value of Information of Example 6.2.1

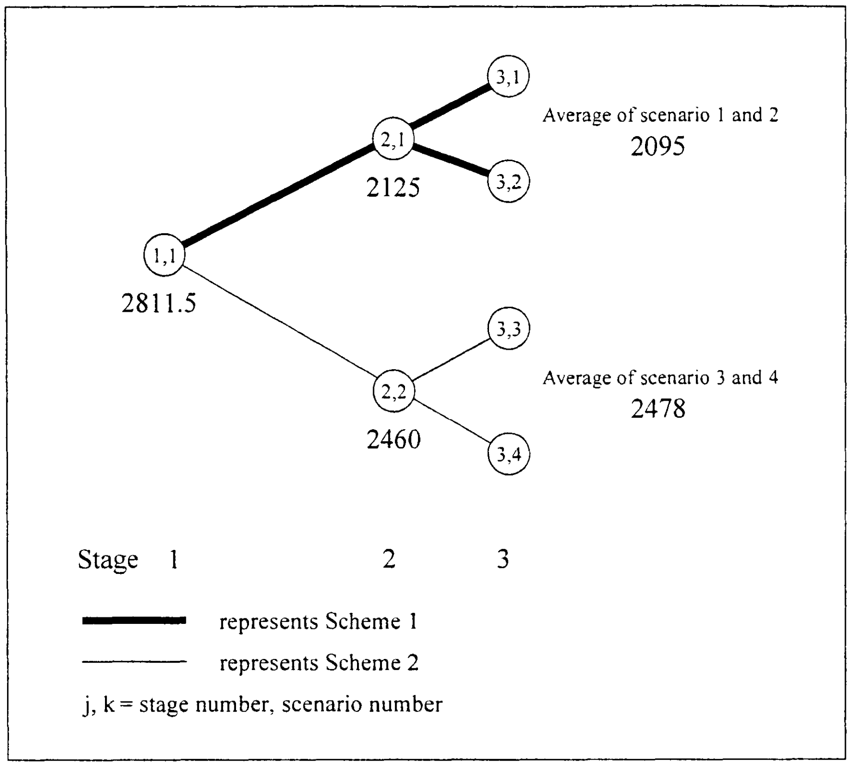

3,1
2,1
Average of scenario 1 and 2
2095
2125
3,2
1,1
2811.5
3,3
2,2
Average of scenario 3 and 4
2478
2460
3,4
Stage 1
2
3
represents Scheme 1
represents Scheme 2
j, k = stage number, scenario number

Figure 6.22 Definition of Scheme of Example 6.2.1

Unfortunately, for a long planning horizon (a lot of number of stages) where sampling is mandatory (in this case, Monte Carlo simulation is used), the possibility of incorrectness of some of the samples (as explained in Example 6.2.2) and the incompleteness of the samples forbid us from deploying different strategies for different possible paths. Thus, the sampling effect forbids us from taking the full advantage of the closed-loop control and thus prevents us from obtaining the value of information.

6.2.4.2 Correctness

As in the deterministic case, the correctness of the LP-version of the algorithm is presumed from the proof of Bender's Decomposition method and from tests done on the algorithm (Perreira and Pinto 1991, Birge and Louveaux 1997). However, for the MIP-version algorithm, there is no guarantee that it will always yield the optimal solution.

Moreover, the correctness of the results depends on the number of samples and the number of iterations, which are inputs for the algorithm. If the number of samples and the number of iterations are insufficient, the result can be inaccurate. This happened during one of the simulations, when Example 6.2.1 was run using three samples (*num_of_samples* = 3) and the five iterations (*iterations* = 5).

As also seen in the deterministic case, our problem structure for a planning horizon of three stages requires at most four iterations before reaching the correct solution. However, in the stochastic cases, the number of samples and the number of iterations has to beat the randomness, otherwise the majority of the nodes will never been touched/sampled enough times and the solution will be suboptimal. To find the number of samples necessary in order to achieve a certain level of accuracy or confidence, Equation 6.19 and 6.20 can be used.

The uncertainty around the upper bound due to the Monte Carlo simulation can be measured by the standard deviation of the estimator (Perreira and Pinto 1989):

$$\sigma_{upperbound} = \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}(E[\theta_{upper}] - \theta^i_{upper})} \qquad (6.19)$$

n = number of samples

$E[\theta_{upper}]$ = expected upper bound in an iteration (of n samples)

θ^i_{upper} = the calculated upper bound of one sample run

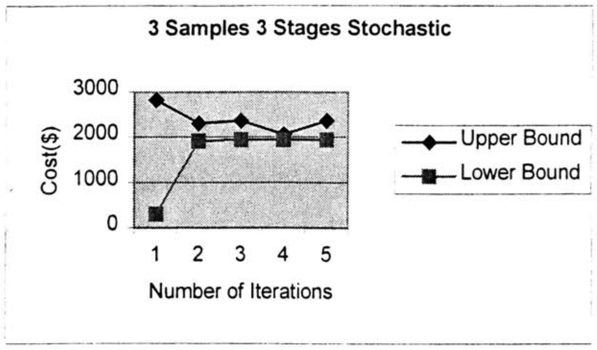
Hence, the 95% confidence interval for the "true" value of $E[\theta_{upper}]$ is given by:

$$[\,E[\theta_{upper}] - 2\sigma_{upperbound}\ ,\ E[\theta_{upper}] + 2\sigma_{upperbound}\,] \qquad (6.20)$$

As shown above, the more samples taken the smaller the standard deviation and the tighter the bound is. The accuracy of the upper bound has a positive linear relationship with number of samples.

6.2.4.3 Convergence

The convergence primarily depends on the number of samples, and by the number of iterations. The algorithm converges if it has found the optimal solution (solutions from one iteration to another are stable). The convergence with random sampling can be seen from the stability of the lower bound. In figures 6.23 and 6.24, the convergence is reached after 3 iterations. The graphs below are taken from the results of Examples 6.2.1.



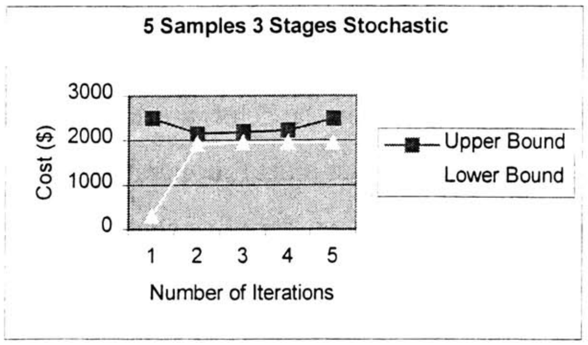


Figure 6.23 Convergence of Example 6.2.1, w/ 3 samples Figure 6.24 Convergence of Example 6.2.1, w/ 5 samples

The oscillation of the upper bound is apparent when the number of samples is three (figure 6.23) and the number of samples is five (figure 6.24). It is less apparent when the number of samples is ten (figure 6.25). The oscillation continues even after the optimal solution has been reached at iteration 3. One of the indications that the algorithm has found the optimal solution is the stability of the lower bound value. Increasing the number of iterations after the optimal solution has been found will not dampen or eliminate the oscillations due to the random sampling used by the algorithm.

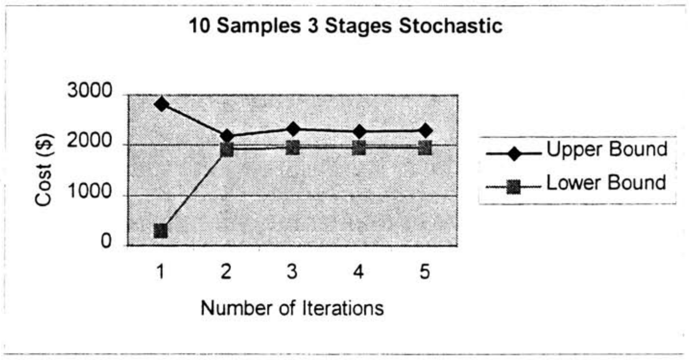


Figure 6.25 Convergence of Example 6.2.1, w/ 10 samples

6.2.4.4 Computational Efficiency

The primal simplex method is used in our simulation for the backward pass. Unlike the deterministic case, the linear programming optimizer is the dominant factor in the running time (*primopt*). This is due to the backward recursion, which requires evaluating all the children of the trial decision $\hat{X}_t^i$ (refer to the algorithm) to create a new hyperplane constraint for the parent node. Thus, the LP solver is run *num_of_children* times more than the MIP solver. Assuming

CPLEX's Primal Simplex method uses the Revised Simplex method (Bertsimas and Tsitsiklis 1997), the computational efficiency of the stochastic algorithm is:

	Stochastic Programming Algorithm	primopt
Best Case	$O(T*num\_of\_samples*num\_of\_children*primopt)$	$O(NUMROWS^2)$
Worst Case	$O(v*T*num\_of\_samples*num\_of\_children*primopt)$	$O(NUMROWS*NUMCOLS*v)$
Memory	$O(num\_of\_children^T*NUMROWS*NUMCOLS)$	$O(NUMROWS^2)$

Table 6.10 Stochastic Algorithm Computational Performance

v= number of vertices

$NUMCOLS$ = number of primal variables in the problem

$NUMROWS$ = number of constraints in the problem

However, this maximum is almost never achieved, except for specially structured problems. Most of the times, the Revised Simplex method's running time is (Bertsimas and Tsitsiklis 1997):

$$primopt = O(NUMROWS^2*v) \tag{6.21}$$

The best case (when the running time is the fastest) is when the number of iterations is one (*iteration* = 1). This can only happen if the optimal strategy is to allocate all the generator operating-hour at the first few stages and this cannot happen if random sampling is done. The worst case is when the algorithm has to traverse all of the vertices of the dual space. Empirically, the algorithm reaches the optimal solution in a finite time, after a relatively small number of iterations.

The relationships between the running time and the number of stages and between the running time and the number of iterations are similar to those of the deterministic cases. Figure 6.26 shows the positive linear relationship between the number of samples and the running time.

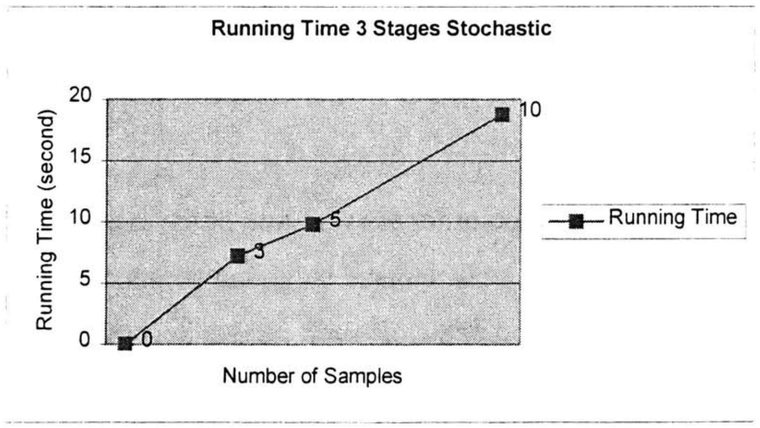


Figure 6.26 Relationship between running time and number of samples from Example 6.2.1

VII: Diverse Electricity Price Structures

7.1 Introduction

The challenge in designing a unified tool in response to the general electricity tariffs schedule as mentioned in Chapter 1 is the diverse energy charges. This thesis will focus on the Real Time Price (RTP) or Spot Price energy charge. Other tariff schedule such as Time of Use (TOU) energy charge can be solved by extending the algorithm without fundamental changes.

The RTP energy charge varies from hour to hour depending on the demand and of the supply of the electricity in the market. The TOU energy charge varies from one block (each block consists of several hours) to another (for example: 2 blocks in one day: peak hours and off peak hours). Demand Charge will be explained in the later sections and an unfinished work on Ratcheted Demand Charge will be covered in the Appendix A.

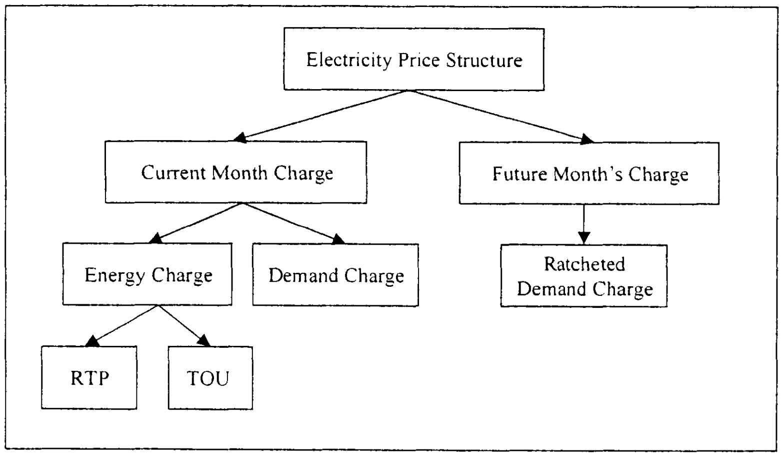


Figure 7.1 Diverse Price Structure

There are two main reasons from the utility standpoint for the variation of the energy charges. First, the charges are used to reduce the utility's peak demand by giving consumers financial incentives to shift their load from the peak period to the off-peak period. Second, the charges are used to cover the basic costs in generating and distributing electricity, such as fuel costs, general overheads, and maintenance costs.

7.2 Flat Demand Charge

On the top of the basic energy charge, there is also a demand charge, which is the charge imposed on the consumer's maximum power drawn from the grid over an hour within a determined length of a period (usually one month period). The demand charge is a penalty for the consumer's peak energy use, and its goal is for the utility company to generate sufficient revenue to allow it to invest in peak capacity.

There are two types of demand charge: a Flat Demand Charge and a Time-Varying Demand Charge. The Flat Demand Charge has some fixed price determined for each month, while the Time-Varying Demand Charge price varies from one period to another. Throughout the studies, the electricity price is considered as an input (i.e. given ahead by the utility company) to the optimization algorithm.

7.2.1 Problem Formulation

New constraints are added at each stage to find the maximum power purchased from the grid. The charge can be appended into the problem's main objective function only at the end of the period:

$$p\_demand \bullet x\_\max_T \tag{7.1}$$

x_max = expected maximum demand in the period

p_demand = the charge associated with the maximum demand; can be fixed or varying from one hour to another.

x_max itself is restricted by:

$$x\_\max \geq x_i \qquad i\text{=remaining hours in the period} \tag{7.2}$$

$$x\_\max \geq x_{current\_\max} \tag{7.3}$$

Hence, the linear programming statement of the main problem is modified to include the new objective function:

$$P_1^i \bullet X_1^i \qquad P_2^i \bullet X_2^i \qquad \text{-----------------} \qquad P_T^i \bullet X_T^i + p\_demand \bullet x\_\max_T \qquad (7.4)$$

and to include the new state transition equation of the demand charge:

$$\begin{array}{lllll} x\_\max_1^i \geq x_1^i & x\_\max_2^i \geq x_2^i & & x\_\max_T^i \geq x_T^i & \\ x\_\max_1^i \geq d_1^i & x\_\max_2^i \geq d_2^i & \text{-----------------} & x\_\max_T^i \geq d_T^i & ;\ \forall i = 1,\ldots,n \quad (7.5) \\ 0 = d_1^i & x\_\max_1^i = d_2^i & & x\_\max_{T-1}^i = d_T^i & \end{array}$$

Stage 1 *Stage 2* *Stage T*

i = index of possible paths in the scenario tree (there are n possible paths)

T = the last stage which is the end of the period for the demand charge

d_t^i = the transition variable at time t path j used to carry the value of the previous maximum

x_max_i = the current maximum electricity purchased at time i (KW)

----------------- = indicates typical formulation for the in-between stages

7.2.2 Sample Cases

Example 7.2.1

This example illustrates the weakness of the algorithm that uses Mixed Integer Programming (MIP) solver for the forward pass. This example has similar building load and electricity price profile (figure 7.3) to those of the Example 6.1.1 in Chapter 6, except there is a \$3.00 demand charge levied on the maximum load and the Load value in hour 12 is increased significantly (figure 7.2). This change should result in different dispatch profiles from the result when there is no demand charge imposed.

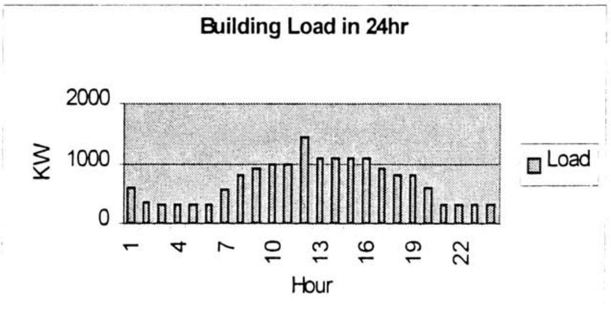


Figure 7.2 Example 7.2.1 Load Profile

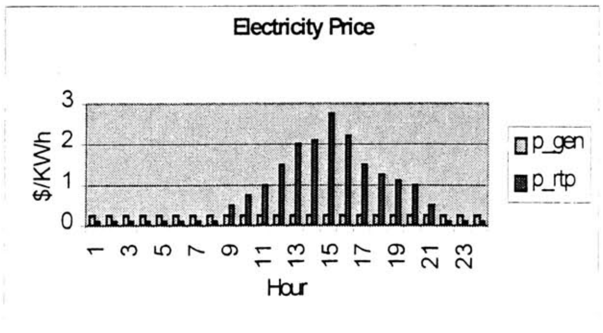


Figure 7.3 Example 7.2.1 Electricity Price profile

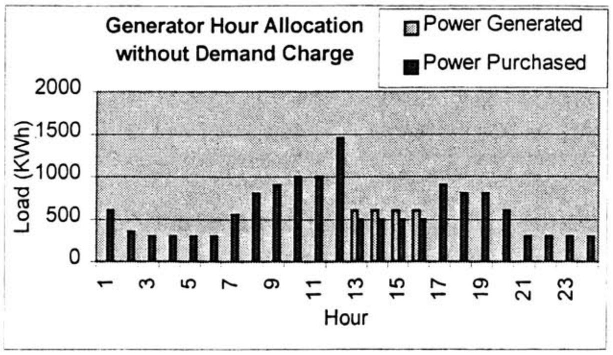


Figure 7.4 Example 7.2.1 Dispatch Profile without Demand Charge

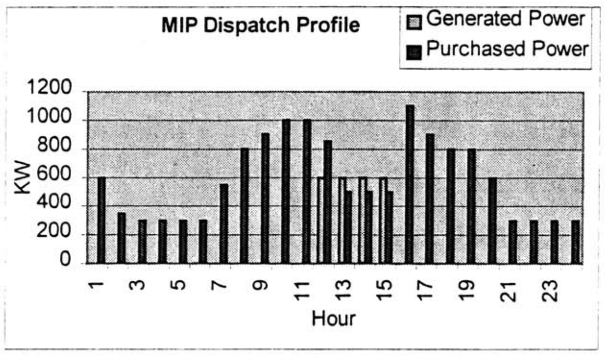


Figure 7.5 Example 7.2.1 Dispatch Profile with Demand Charge

The results when there is a demand charge imposed (figure 7.5) and when there is not (figure 7.4) are different. However, the result is not optimal. The optimal solution is to allocate the generator operating-hour in *hr 12, 13, 14, 15* (figure 7.6).

Generator Capacity	600 kW
Demand Charge	$3.00
Operating Hour Limit	4 Hr
Optimal cost	$17,520.00
MIP optimized Cost	$17,640.00
LP optimized Cost	$17,141.88
Difference	**$ 120.00**

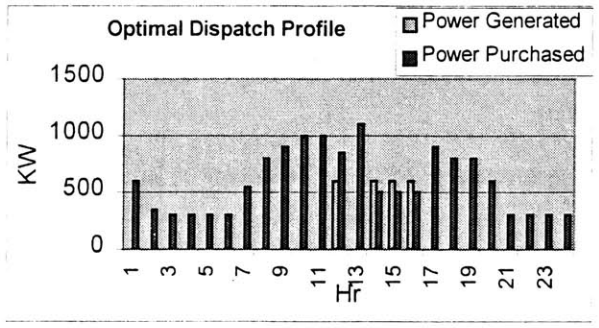


Figure 7.6 Example 7.2.1 Optimal Dispatch Profile

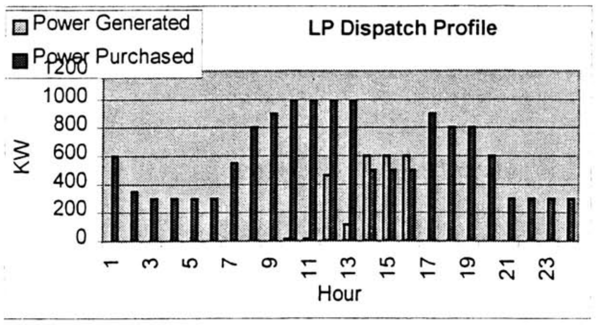


Figure 7.7 Example 7.2.1 LP Dispatch Profile

Generator Hour allocation with LP Solver							
Hour	10	11	12	13	14	15	16
Power Generated	12.5	12.5	462.5	112.5	600	600	600

Table 7.1 LP Solution of Example 7.2.1

Whenever LP and MIP solutions are different for a long planning horizon period, there is a big chance that an error occurs. As we can see from the LP solution above (table 7.1 and figure 7.7), the forward pass using the MIP solver eliminated some of the information from the dual space cuts and thus yields a suboptimal solution. The cutting planes at hours 10 and 11 are not sufficiently substantial to change the decision into allocating generator operating hours at those hours. However, the cut at hour 13 is substantial enough to skew the MIP solver result so that the algorithm allocates a generator operating hour at that hour, even though the LP solution allocates a small fraction at hour 13. This phenomena will be referred as the *LP cuts effect*.

The turning point of what LP solution value is sufficiently substantial to affect the MIP solver result needs further research and is beyond the scope of this thesis.

<u>Example 7.2.2</u>

This example tries to illustrate *the effect of ordering*. In this example, the Load and RTP price profiles (figure 7.8 and 7.9) are similar to the previous example, Example 7.2.1, with the RTP charge of hour 13 and 16 switched.

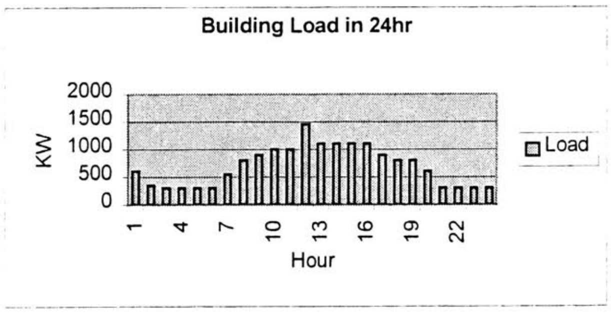


Figure 7.8 Example 7.2.2 Load Profile

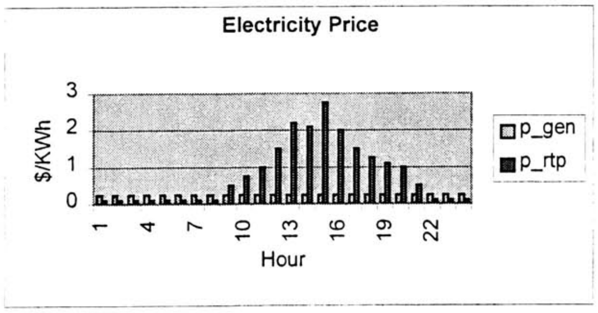


Figure 7.9 Example 7.2.2 Electricity Profile

The LP solver result is the same as that of the previous example with the solution of hour 13 and 16 switched (table 7.2 and figure 7.10).

Generator Hour allocation with LP Solver							
Hour	10	11	12	13	14	15	16
Power Generated	12.5	12.5	462.5	600	600	600	112.5

Table 7.2 LP Solution of Example 7.2.2

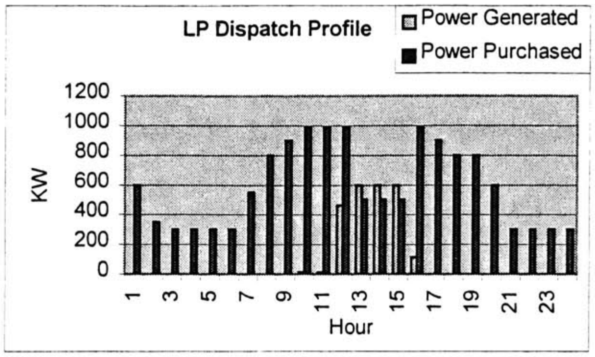


Figure 7.10 Example 7.2.2 LP Solution Profile

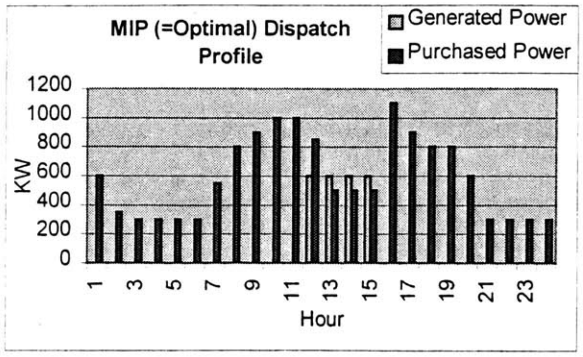


Figure 7.11 Example 7.2.2 MIP and Optimal Solution

Here, the results are now optimal (figure 7.11) because the algorithm obtained the solution during the forward pass by using the MIP solver, and thus it encountered the problem at Stage 16 later when there are no more generator operating hours available. In other words, the distractive stage (hour 13), which skews the solution, is encountered after all the generator operating hours have been allocated properly and thus does not affect the solution. Hence, for a problem that has a particular structure such that the distractive LP cuts happen later in the stage after all the generator operating hours have been properly allocated, the algorithm will always yield the correct result.

Example 7.2.3

In this example, the building load profile (figure 7.12) is the same as that of Example 7.1.1. we can see that the effect that we see in Example 7.1.1 can be lessened if we increase the RTP price at hour 16 from *2.2* to *3* (figure 7.13). In other words, if the optimal value difference is large enough the MIP solver in the forward pass will sense it. This effect will later be referred to as the *relative objective cost difference effect*.

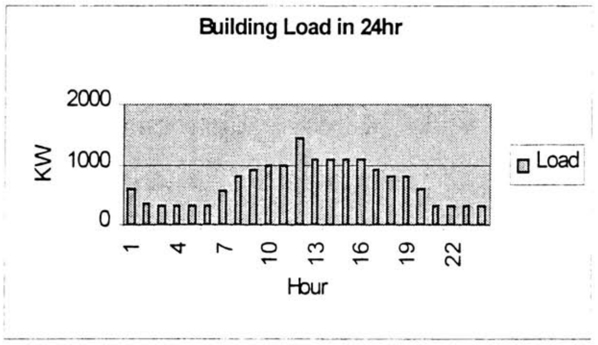


Figure 7.12 Example 7.2.2 Load Profile

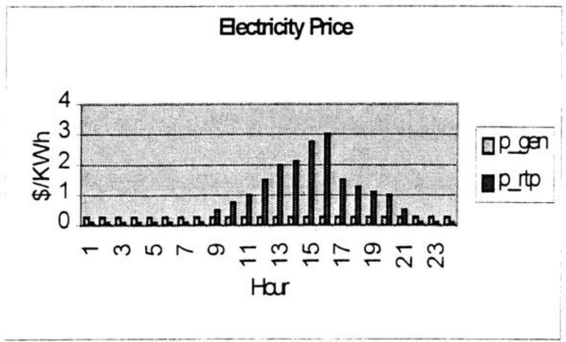
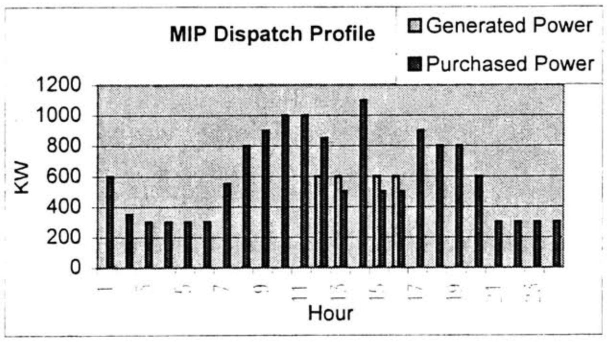


Figure 7.13 Example 7.2.2 Electricity Price Profile

The result using the LP solver is the same as in Example 7.2.1 (table 7.2). Moreover, the dispatch profile without the demand charge is identical to that of Example 7.2.1 (figure 7.4).

Generator Capacity	600 kW
Operating Hour Limit	4 Hr
Optimal Cost	$17,920.00
MIP optimized Cost	$17,980.00
LP optimized Cost	$17,541.88
Difference	**$ 60.00**



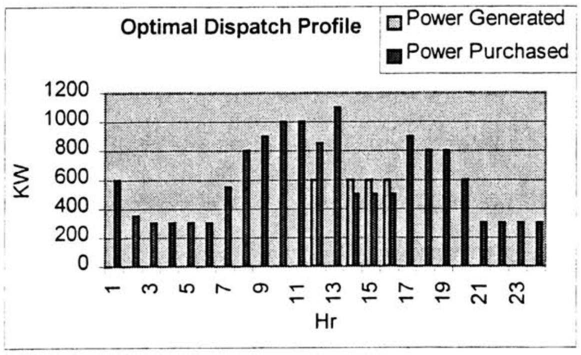


Figure 7.14 Example 7.2.2 MIP Dispatch Profile

Figure 7.15 Example 7.2.2 Optimal Dispatch Profile

The optimal solution is shown in figure 7.15. Now, the algorithm allocates the generator operating hour at hour 16 instead of hour 14 (figure 7.14). What happens here is that the marginal cost difference between hour 13 and 16 is more apparent than the marginal cost difference between hour 13 and 14. Moreover, the difference between the objective function at hour 13 (*p_rtp*=2.0) and that of hour 14 (*p_rtp*=2.1) is not substantial enough for the MIP solver to capture in the forward pass. If the RTP charge at hour 14 is increased so that it is substantially larger than that of hour 13 the algorithm will yield the correct result due to the relative objective cost difference effect.

Hence, the sensitivity of the algorithm depends on the marginal cost difference at each stage. The algorithm is not very sensitive to a subtle marginal cost difference among stages. The larger the marginal cost difference, the better the algorithm works. Moreover, the sensitivity of the algorithm is influenced by the resemblance between the LP solution and MIP solution.

Example 7.2.3

This example illustrates the advantage of our algorithm (closed-loop control) in the stochastic scenario over the regular deterministic optimization method (open-loop control) such as LP.

Stage (Stg)	Schenario	L (kW)	p_gen ($/kWh)	p_rtp ($/kWh)	gen_max OH	C_{gen}	demand_charge
1		300	1	2	1	500	3
2	1 (p_1=0.5)	400	1	2.5	1	500	3
	2 (p_2=0.5)	510	1	2.5	1	500	3
3	1 (p_1=0.25)	300	1	3	1	500	3
	2 (p_2=0.25)	450	1	3	1	500	3
	3 (p_3=0.25)	350	1	3	1	500	3
	4 (p_4=0.25)	550	1	3	1	500	3

Table 7.3 Case of Example 7.2.3

Closed-Loop Solution				
	Stage 3	stage 2	stage1	
Scheme1	use@3-1 3175	use@2-1 3250	add@1 5277.5	Opt. Cost **3492.5**
Scheme2	use@3-2 3855	use@2-2 3810		
Open-Loop Solution				
	stage 3	stage 2	stage 1	
	use@3 3515	use@2 3530	use@1 5277.5	Opt. Cost 3515

Table 7.4 Advantage of Closed-Loop Control of Example 7.2.3

The value of information in this example is *$22.5* (table 7.4). For this example, no samplings are done, thus, there is no sampling effect as in Examples 6.2.2. Moreover, even though the LP and MIP solutions are different, the LP cuts effect and the ordering effects do not skew the result. This is due to the short planning horizon period (3 stages) of this problem. Those effects will be apparent in Example 7.2.6.

Example 7.2.5

In this example, the algorithm will be tested for a longer planning horizon period. The demand charge will be \$3.00 and Monte Carlo Simulation is used for sampling. The building load is divided into the weather load and the expected occupancy load. The transition probabilities for the weather load due to the outside temperature swing are shown in the table below.

I/j	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800
100	0.5	0.5													
150	0.25	0.5	0.25												
200		0.25	0.5	0.25											
250			0.25	0.5	0.25										
300				0.25	0.5	0.25									
350					0.25	0.5	0.25								
400						0.25	0.5	0.25							
450							0.25	0.5	0.25						
500								0.25	0.5	0.25					
550									0.25	0.5	0.25				
600										0.25	0.5	0.25			
650											0.25	0.5	0.25		
700												0.25	0.5	0.25	
750													0.25	0.5	0.25
800														0.5	0.5

Table 7.5 Building Load PMF of Example 7.2.5

i = the current building load from weather (kW)

j = the future building load from weather (kW)

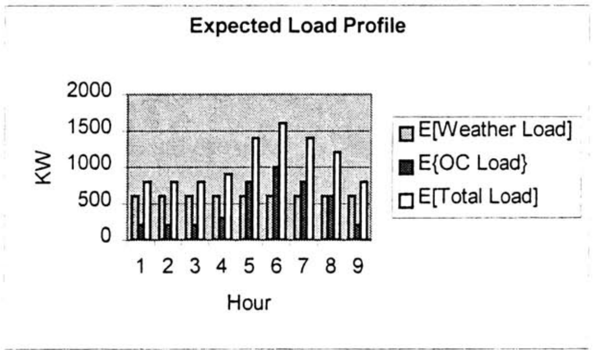


Figure 7.16 Example 7.2.5 Expected Load Profile

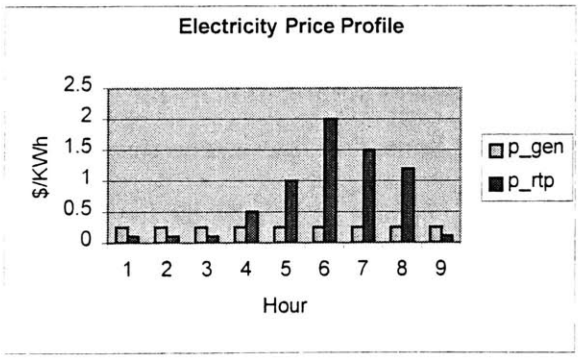


Figure 7.17 Example 7.2.5 Electricity Price Profile

Generator Capacity	600kW
Operating Hour Limit	3Hr
Demand Charge	\$ 3.00
Number of Iterations	10

Based on the demand charge, the expected load profile and the RTP charge profile, the allocation of generator operating hours at hours 5, 6, and 7 is expected to be the result for most samples (figure 7.18). The second possible strategy is to allocate the generator operating hours at hours 6, 7, and 8 (figure 7.19). The second strategy is better if the weather load at hour 5 is less than the weather load at hour 8 or if such that the RTP charge difference between hour 5 and 8 (RTP charge at hour 5 = \$1.00 and at hour 8 = \$1.20) costs more than the demand charge incurred. The result (figure 7.20) verifies our expectation.

Strategy	Gen. Hr. Allocation
1	5,6,7
2	6,7,8
3	5,6,8
4	6,8,9
5	4,5,6

Table 7.6 Solution of Example 7.2.5

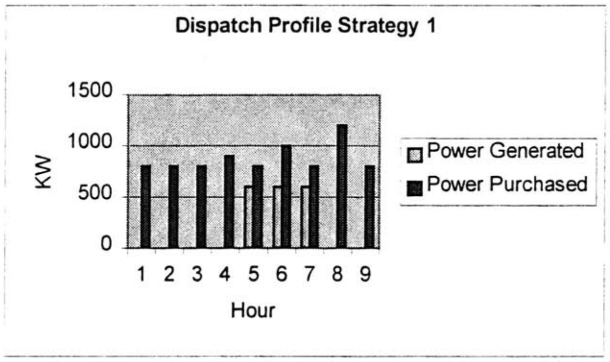


Figure 7.18 Example 7.2.5 Strategy 1 Dispatch Profile

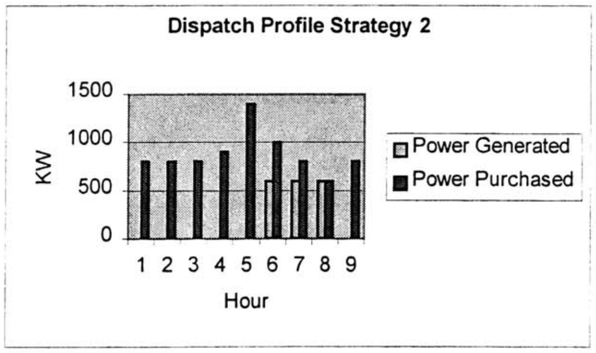


Figure 7.19 Example 7.2.5 Strategy 2 Dispatch Profile

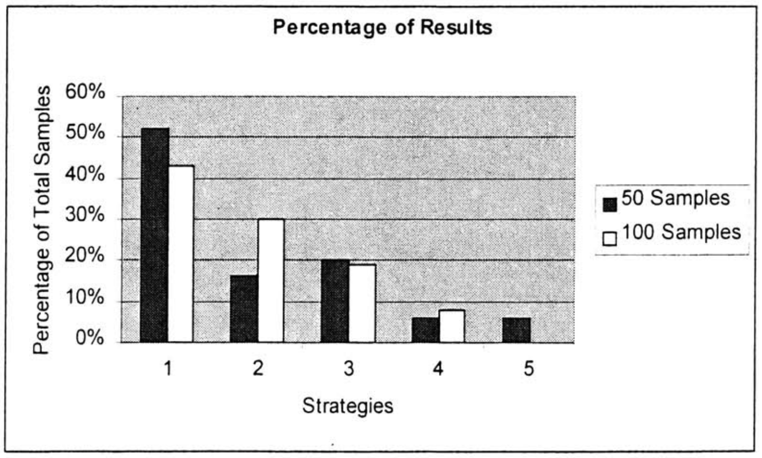


Figure 7.20 Example 7.2.5 MIP Results

Moreover, the result shows that increasing the number of samples increases the accuracy of the solution for two reasons. First, the simulation with 100 samples recommends Strategy 2 as the second best strategy as we expected, but that is not the case when the number of samples is 50. Second, the simulation with 100 samples eliminates Strategy 5, which will never become an attractive strategy under any circumstances. More importantly, the results illustrate the sampling effect that was mentioned in Example 6.2.2.

Example 7.2.6

This example shows how the ordering effect (Example 7.2.2) and the LP cuts effect (Example 7.2.1) can distort the results of the algorithm significantly.

The PMF is the same as that of Example 7.2.5. Further, the expected load profile and the RTP charge profile are similar to those of Example 7.2.5, except for hour 5 and 8. The occupancy load and the RTP charge for hour 5 and 8 are switched (figure 7.22); thus, we expect strategy 2 to predominate the results.

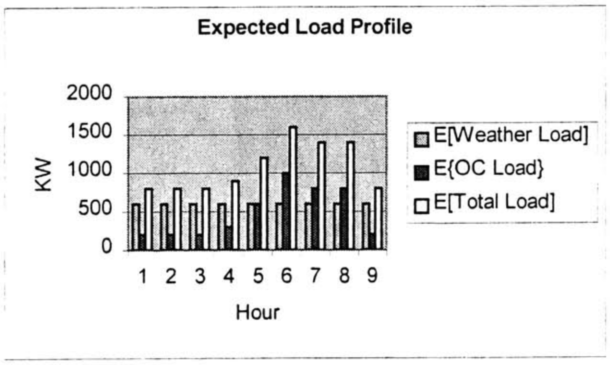


Figure 7.21 Example 7.2.6 Expected Load Profile

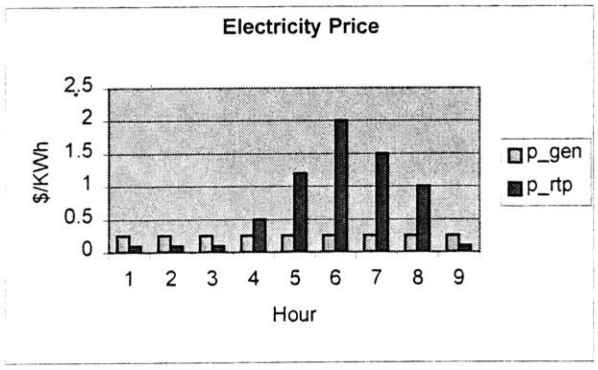


Figure 7.22 Example 7.2.6 Electricity Price Profile

Strategy	50 Samples	100 Samples
1	54%	39%
2	16%	23%
3	18%	20%
4	12%	12%
5	0%	2%
6	0%	2%
7	0%	2%

Table 7.7 Result of Example 7.2.6

Generator Capacity		600kW
Operating Hour Limit		3hr
Demand Charge	$	3.00
Number of Iterations		10

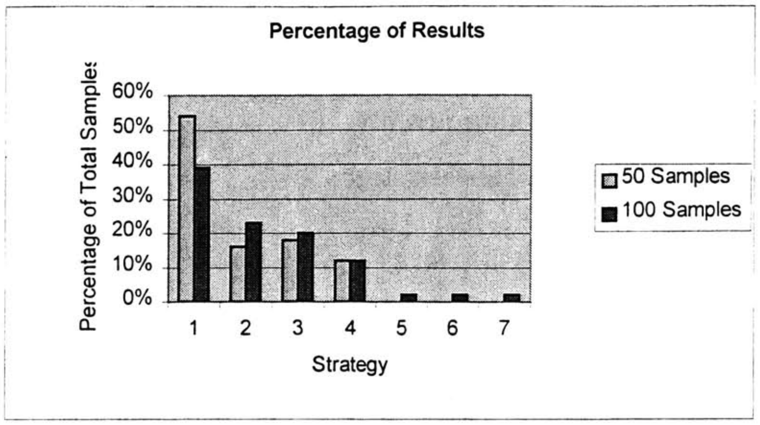


Figure 7.23 Example 7.2.6 MIP Results

The results (figure 7.23 and table 7.7) are not what was expected: Strategy 1 is still the predominant solution for majority of the samples followed by Strategy 2. This error is caused by the ordering effect and LP cut effect. Changing the order of the load and the RTP price (switching hour 5 and 8) has an effect on the correctness of the algorithm.

Moreover, the error is also caused by the sampling effect. When the number of samples is one hundred, the result is closer to our expectation than when the number of samples is fifty, even though the result is still incorrect. Strategy 1 is less dominating and Strategy 2 becomes the second most dominant solution (as compared to the third when *num_of_samples*=50). Increasing the number of samples will not eliminate the error caused by the ordering effect and LP cut effect.

7.3 Time-Varying Demand Charge

The previous section describes a pricing structure where the maximum demand charge is the same across all hours. In this section, another type of demand charge will be discussed, the Time-Varying Demand Charge, for which demand charge price varies from one period to another. The length of the period is set by the utility company.

7.3.1 Problem Formulation

The problem formulation is the same as the one for Flat Demand Charge, except that the demand charge price (*p_demand*) in Equation 7.4 changes according to the time when the maximum electricity is purchased from the utility company.

$P_1^i \bullet X_1^i$ $\quad\quad$ $P_2^i \bullet X_2^i$ $\quad\quad$ ---------------- $P_T^i \bullet X_T^i + p\_demand \bullet x\_\max_T$

T = the last stage which is the end of the period for the demand charge

x_max_i = the current maximum electricity purchased at time i (KW)

---------------- = indicates typical formulation for the in-between stages

7.3.2 Sample Cases

Example 7.3.1

This example is similar to example 7.2.3, except the demand charge at stage 2 is \$1.5/KWh instead of \$3/KWh (table 7.8) and this will affect the decision to allocate the generator operating hour at stage 3 for all scenarios. The algorithm returns the expected results (table 7.9) and thus, verifies our formulation for the planning horizon period of three (t = 3).

Stage (Stg)	Schenario	L (kW)	p_gen (\$/kWh)	p_rtp (\$/kWh)	gen_max OH	C_{gen}	demand_charge
1		300	1	2	1	500	3
2	1 (p_1=0.5)	400	1	2.5	1	500	1.5
	2 (p_2=0.5)	510	1	2.5	1	500	1.5
3	1 (p_1=0.25)	300	1	3	1	500	3
	2 (p_2=0.25)	450	1	3	1	500	3
	3 (p_3=0.25)	350	1	3	1	500	3
	4 (p_4=0.25)	550	1	3	1	500	3

Table 7.8 Case of Example 7.3.1

Closed-Loop Solution				
	stage 3	stage 2	stage1	
Scheme1	use@3-1	use@2-1	add@1	Opt. Cost
	2575	3250	3357.5	**2832.5**
Scheme2	use@3-2	use@2-2		
	3090	3810		

Table 7.9 Solution of Example 7.3.1

Example 7.3.2

This example illustrates another limitation of the algorithm with our current problem formulation. In this example the load is divided into two parts: weather loads and expected occupancy loads. The expected load and the electricity cost profiles are shown in figure 7.24 and 7.25 respectively. The weather load has the binomial PMF as shown in the table below (table 7.10). Each node branches into three possible scenarios in the next stage (*num_of_children* = 3).

i/j	300	400	500	600	700	800	900
300	0.5	0.5					
400	0.25	0.5	0.25				
500		0.25	0.5	0.25			
600			0.25	0.5	0.25		
700				0.25	0.5	0.25	
800					0.25	0.5	0.25
900						0.5	0.5

Table 7.10 Building Load PMF of Example 7.3.2

i = the current building load from weather (kW)

j = the future building load from weather (kW)

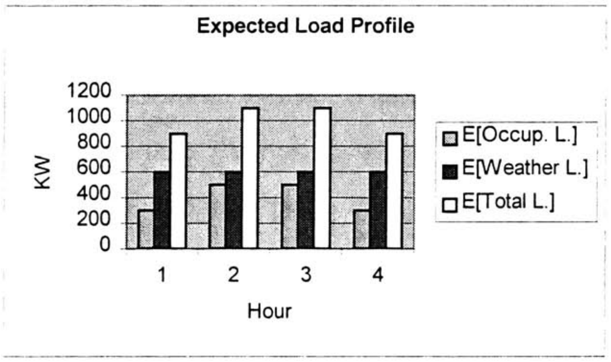


Figure 7.24 Example 7.3.2 Expected Load Profile

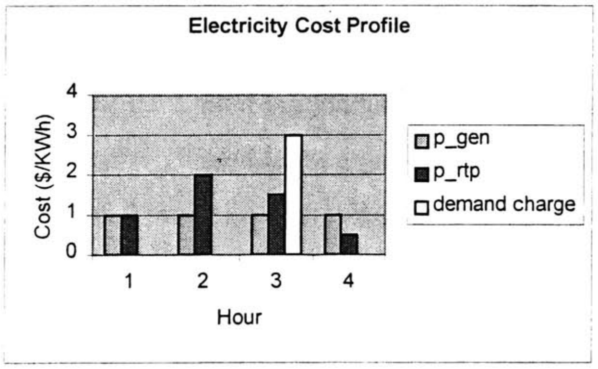


Figure 7.25 Example 7.3.2 Electricity Price Profile

Generator Capacity	600kW
Operating Hour Limit	1hr
Number of Iterations	6
Number of Samples	9
Number of Children	3

There is no random sampling for this example since the number of possible paths is still manageable (27 possible paths). The sampling is done exhaustively up to Stage 3 (= *T-1*) (table

7.11). This exhaustive sampling eliminates the sampling effect and ensures that all the nodes are evaluated sufficiently.

	Stage 1	Stage 2	Stage 3	Stage 4
Sample 1	0	0	0	1
Sample 2	0	0	1	4
Sample 3	0	0	2	7
Sample 4	0	1	3	10
Sample 5	0	1	4	13
Sample 6	0	1	5	16
Sample 7	0	2	6	19
Sample 8	0	2	7	22
Sample 9	0	2	8	25

Table 7.11 Samples of Example 7.3.2

The number in the table represents the scenario number at each stage, which increases exponentially relative to the increase in the stage number. Since in this case the initial stage is Stage 1, the total number of scenarios at stage t = $num\_of\_children^{t-1}$.

From calculating the expected value of all the samples (also apparent by looking at the Electricity Cost Profile and the Expected Load Profile), the expected solution is to allocate the generator operating hour at hour 2. However, the algorithm returned a strategy to allocate the generator operating hour at hour 1. Also interesting is that the algorithm actually evaluates the correct solution at iteration 4 (figure 7.26), which resulted in a dip (an upper bound is lower than the lower bound), but then, the next iteration overrode the optimal solution and returned to the suboptimal solution.

This error happens because the Nested Bender's Decomposition method that we derived in Chapter 4 assumes a constant C matrix. In the current formulation:

$$X_t = \begin{bmatrix} u_t \\ x_t \\ y_t \\ z_t \\ \theta_t \\ x\_\max_t \\ d_t \end{bmatrix} \qquad ;\forall t \tag{7.6}$$

$$C_t = \begin{bmatrix} p\_gen_t \\ p\_rtp_t \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \forall t \neq T; \qquad C_T = \begin{bmatrix} p\_gen_T \\ p\_rtp_T \\ 0 \\ 0 \\ 0 \\ p\_demand \\ 0 \end{bmatrix} \tag{7.7}$$

The definitions for the notations are from Chapter 3.

In the Flat Demand Charge case, C_T is fixed and thus complies with the assumption of Bender's Decomposition method. However, for the Time-Varying Demand Charge p_demand varies depending on when the maximum electricity was purchased from the grid, and thus C_T may vary from one iteration to the next.

The algorithm saves the cuts from the previous iteration, which stores the information of the previous objective function cost of previous C_T (the details of objective function cuts are explained in Section 6.1.4.1). This old objective functions cuts affect the new objective function cuts and the effects continues.

For example, in this case, since the $p\_demand = 3$ in the first three iterations, all the nodes save the objective function cuts for $p\_demand = 3$ and set the lower bound to a certain value. Further, since the algorithm works so that the LP cuts forms the lower bound, which will then set the decisions for the forward pass to find upper bound, the upper bound value will always be larger than the lower bound.

At iteration 4 when the algorithm tries to evaluate a new possible strategy, which is the actual optimal solution with $p\_demand = 0$ and thus has an upper bound lower than the lower bound, the algorithm overrides the solution in the next iteration by forming the new cuts that are affected by old p_demand (= 3) (figure 7.26). The lower bound could never decrease even though the C (cost matrix) components decrease, because it keeps accumulating the cuts from the previous iteration, which may restrict the solution to a higher optimal cost value.

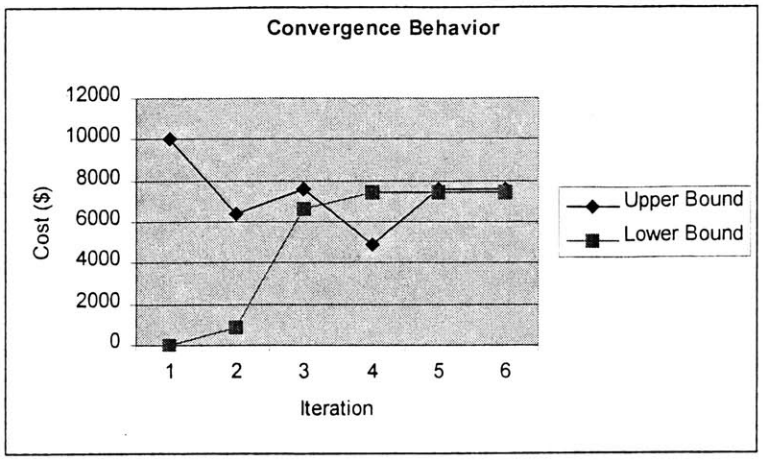


Figure 7.26 Example 7.3.3 Upper Bound and Lower Bound Behavior

One possible solution for this problem is to stop the iteration if the upper bound is lower than the lower bound. This is a heuristic method. So far it works for the trial cases. However, this solution approach cannot be used if random sampling is used. Only exhaustive sampling up to Stage *T-1* can be used (as was done in this example), which limits the usefulness of this approach.

7.4 Summary of Key Findings

Throughout this chapter, we have seen a lot of limitations for Nested Bender's Decomposition method when it is applied to our problem. There are two main causes for the inaccuracies. The first one is the randomness of the sampling method, which is also called the sampling effect. Example 6.2.2 illustrates this effect. The sampling effect causes some of the samples to yield incorrect solutions and even worse, this effect prevents us from taking the advantage of the closed loop nature of the algorithm and thus, the value of information (Section 6.2.4.1).

The second cause is the MIP nature of the problem, which creates another complication by running MIP solver in the forward pass. As we have mentioned earlier there is no guarantee that this method yield an optimal solution. There are several effects of running this MIP solver in the forward pass that lead to errors, such as the LP cuts effect (Example 7.2.2), the ordering effect (Example 7.2.1), and the relative objective cost difference effect (Example 7.2.3).

These effects are not apparent for a short planning horizon (small number of stages). All of the simulations for the three stages or less cases yield correct solutions.

In addition, with the current formulation, the algorithm cannot handle the Time-Varying Demand Charge due to its static cost vector assumption. However, the algorithm returns correct results when the planning horizon period is not more than three and the number of children is two (i.e. Example 7.3.1).

Other issues with the algorithm are the required memory and the running time. For each node in the tree of scenarios, some memory needs to be allocated to save the cuts from the previous iterations, the right-hand-side values affected by the state transition (Equation 7.8), the cost vector components, and the previous solution for the subproblem.

$$W_t X_t^i \leq h_t^i - T_{t-1} \hat{X}_{t-1}^j \tag{7.8}$$

W_t = recourse matrix (Section 5.2 and 4.3)

h_t^i = right-hand-side vector of scenario *i* stage *t* (Section 5.2 and 4.3)

T_{t-1} = technology matrix (Section 5.2 and 4.3)

$\hat{X}_{t-1}^j$ = trial value of scenario *j* (the parent of scenario *i*) stage *t-1* from the previous iteration (Section 5.2 and 4.3)

$h_t^i - T_{t-1} \hat{X}_{t-1}^j$ = right-hand-side value

Not including pointers for the dynamic array allocation, the memory required growth exponentially (*O(num_of_children^T)*). The running time for the 9 Stage problem with *num_of_children* = 3 (= 6561 possible paths) and 100 samples is about 25 minutes.

VIII: Electricity Costs Control Systems

8.1 Precedents

As mentioned earlier in Chapter 1, the thesis tries to construct an optimization algorithm for a unified tool/control system, which can minimize the electricity cost in building using a specified parameter (in this thesis, a generator is used as the parameter).

Previous efforts have been made to construct this tool. None of them have explored the possibility of using Stochastic Programming as the optimization engine and none of them used an auxiliary generator as the parameter.

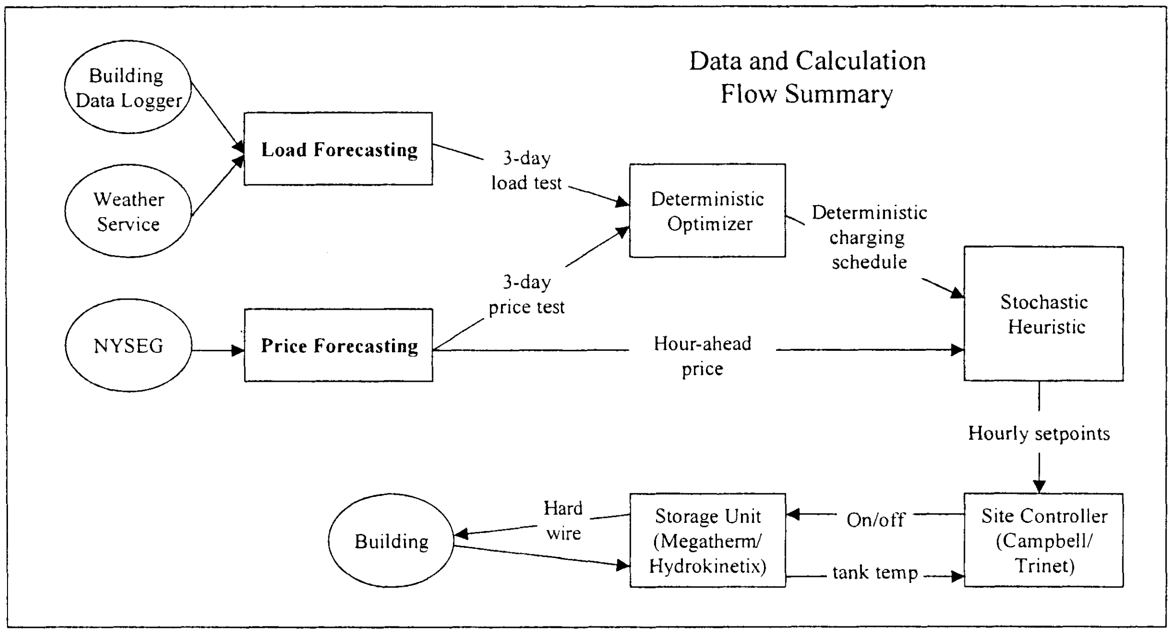


Figure 8.1 Control Flow of the RTP Based Control System (Daryanian *et al.* 1991)

The first work is an RTP based control system, which is done for a thermal storage system (Daryanian *et al.* 1991). This RTP based control system used an open-loop feed forward control. The control algorithm is composed of two stages (figure 8.1). The first stage uses as inputs the load and the day-ahead price forecasts with some extensions and adjustments and determined an optimal storage charging schedule under deterministic conditions. The second stage considers the inaccuracy in the future load and price forecasts, and modifies the charging schedule by a heuristic stochastic algorithm that uses the hour-ahead prices and the measured current tank temperature.

The advantage of this RTP based control system is that it is fast due to its heuristic method and its simplified linear optimization. However, due to the simplified linear optimization, the system only handles RTP charges and cannot handle complex electricity pricing structures such as Flat Demand Charge and Time-Varying Demand Charge. Moreover, there is no guarantee that the heuristic method will yield optimal solutions.

Another electricity cost control system is based on the closed-loop control optimization approach (Henze *et al.* 1997). During operation, the predictions of the weather, electrical price, and building loads are updated at the beginning of each operating time step (typically one hour) over the optimization period. The planning horizon for the proposed controller consists of a fixed length, moving window over the entire simulation period. At each time step only the action of the first hour is executed.

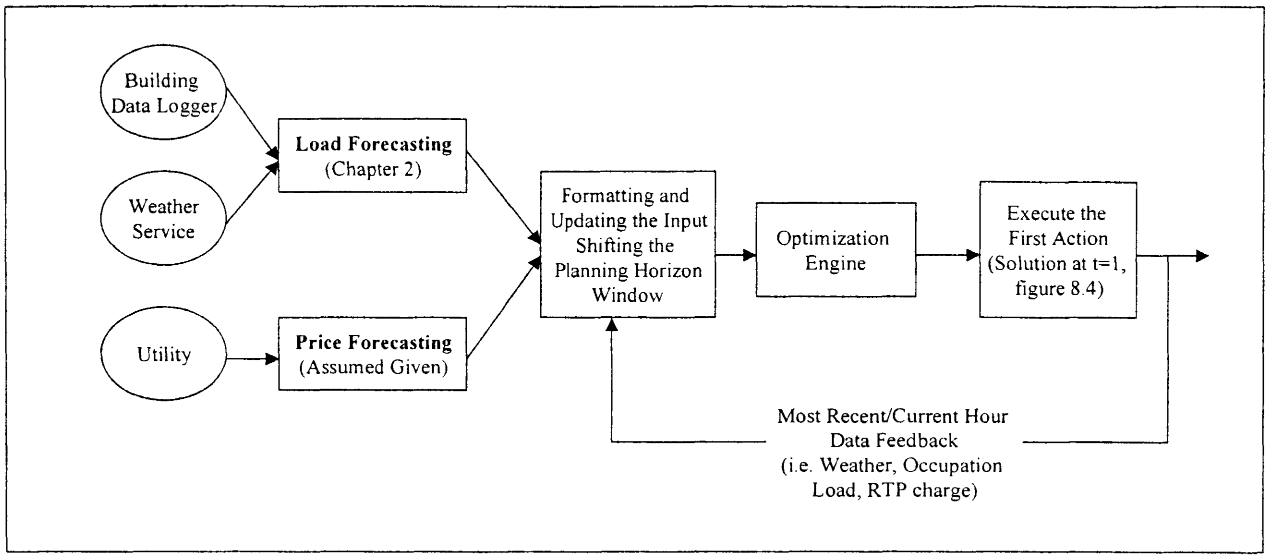
8.2 Planning Horizon

One important factor in constructing electricity cost control systems is determining the optimal planning horizon. As discussed before, the algorithm has a limited planning horizon due to its exponential memory consumption and its long running time. The optimal length of the planning horizon period depends on several parameters, such as the utility rate profile, the building load profile, and the properties and constraints of the generator (the size of the generator, the scope of the generator operating-hour constraints, etc.).

Studies on the optimal planning horizon are beyond the scope of the thesis and hence, are left for future works. Henze and Krarti (1999a) have done some research on the optimal planning horizon for thermal energy storage. The research findings may be extended to cover an auxiliary generator as the control parameter.

8.3 Optimal Controller

The system proposed in this thesis is a hybrid between the two precedents with some improvements. The control system is based on the closed-loop control approach (figure 8.2). At each time step (one hour, in our case), only the action of the first hour recommended by the optimizer is executed. The planning horizon is a fixed-length moving window, *T*. The current weather, occupation load, and RTP charge are fed back to the system, allowing the system to update its inputs.



Building Data Logger
Weather Service
Utility
Load Forecasting (Chapter 2)
Price Forecasting (Assumed Given)
Formatting and Updating the Input Shifting the Planning Horizon Window
Optimization Engine
Execute the First Action (Solution at t=1, figure 8.4)
Most Recent/Current Hour Data Feedback (i.e. Weather, Occupation Load, RTP charge)

Figure 8.2 The Proposed Control System

The input components are analogous to the ones proposed by Daryanian *et al.* (1991). At the start of each month, the historic load data are taken into account to find the outside temperature Probability Mass Function (PMF) during the particular month. The building loads can be derived from the outside temperature PMF by using the scenario tree (Section 2.4). Building loads PMF is unknown and is different from the outside temperature PMF. The PMF and the expected values of the occupancy load can be found from the historic load data as well. The planning horizon (can be one week or one month long or until the end of the month) considers the historical trend, long-range weather forecast, and an early RTP prediction by using the given algorithm from the utility company or by using the previously published rates.

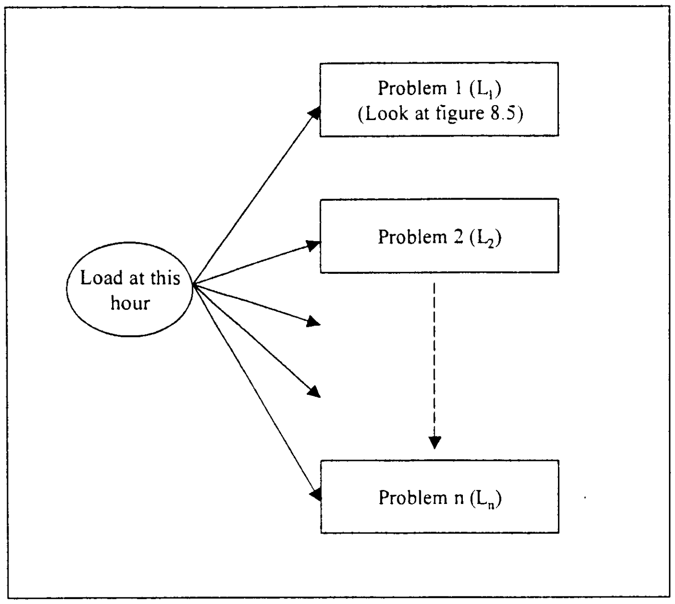


Figure 8.3 Diagram of the Optimization Engine

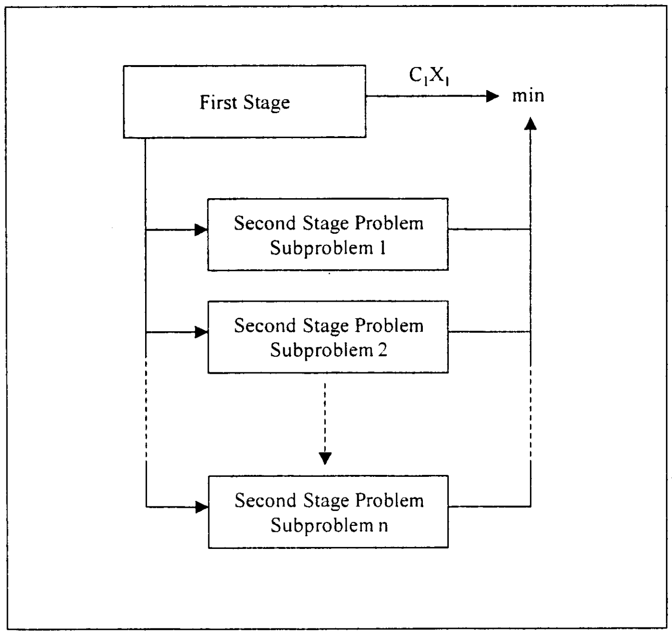

First Stage
C_1X_1
min
Second Stage Problem Subproblem 1
Second Stage Problem Subproblem 2
Second Stage Problem Subproblem n

Figure 8.4 Optimization Process within Each Problem in Figure 8.4

What is different from the precedents is the optimization engine. At each step different problems (figure 8.3) are formed based on the possible next hour's building load given the current load. Thus, each problem is an independent one and can be optimized in parallel, which could be done simultaneously using different computers. This method allows us to determine the optimal control for any possible scheme in the next hour.

Each problem consists of two stages and is formulated as a combination between Stochastic Programming and Linear Programming algorithm. The first stage represents the decision for the next hour (figure 8.4). The possible load changes are represented by different scenarios on the second stage whose PMF is obtained from the historical data. The second stage represents the linear programming of the problem from Stage 2 to Stage T based on the expected value of the random variable. Thus, each subproblem in the second stage is a deterministic optimization of the problem from hour 2 to hour T with different random factor value at Stage 2. These random factor values at Stage 2 are like the initial values, from which we calculate the expected values for the succeeding stages.

The two stage stochastic problem, then, is solved using the Bender's Decomposition method. Hence, the optimization problem has been reduced into a two stage stochastic case. There are several reasons for this reduction. First, Nested Bender's Decomposition method does not work well with a long planning horizon MIP problem. The error can be large and there is no way to tell ahead of time whether the algorithm will yield correct solutions for a particular problem or not. The only way to check the correctness is by verifying the result by using other methods. From the experiments, the algorithm always returns correct results for the number of stages less than or equal to three. Thus, reducing the problem into a two stage stochastic problem fits well within this range and can improve the accuracy.

Second, the algorithm is restricted to solving a limited number of stages (the largest tried is 10 stages with *num_of_children* = 3, almost 30,000 nodes, which is the limit in the PC with 64 MB RAM running Windows 95). This is due to the exponential growth of the memory requirement and the high overhead (pointers, sampling arrays, solution arrays, etc.) incurred. If the problem is solved in a deterministic manner using CPLEX, we can have the planning horizon for even more than a year or more than 8760 hours/stages. A year planning horizon with the current algorithm required $O(num\_of\_children^{8760})$ bytes of memory, which is impossible to be accomplished in a regular PC.

Third, solving the problem as a linear programming problem is much faster than using the Stochastic Programming method. The formulation is also much simpler than the one for Nested Bender's Decomposition method. Thus, the problem can be updated and simulated for the whole year period every hour to give us recommendation for the next hour.

Fourth, two-stage stochastic problem formulation is simpler than the multi-stage problem formulation. Moreover, the linear programming formulation of the subproblems requires is also simpler than the one for Nested Bender's Decomposition method and thus, reduces chances of errors and allows easy modification if we need to modify the formulation to solve a new type of electricity price structure.

Since the control yields different solutions for different paths within the first two stages, the proposed control strategy allows us to do the simulation every two hours, instead of an hourly basis. Moreover, it carries the value of information for the first two stages, which cannot be done with simple deterministic optimization (or with Stochastic Programming with random sampling).

It is true that solving the problem using expected value of the loads may not give us the optimal solution. However, based on the findings, Nested Bender's Decomposition method also does not guarantee achieving optimal solutions. Moreover, this new proposed optimization method is still better than the regular deterministic optimization because it treats the first two stages, which are the most important stages, as stochastic problems.

Problems may arise in constructing the module for updating the input. An efficient and reasonably accurate way to construct expected loads for the new subproblems from the feedback data (figure 8.2 and 8.3) is necessary. This area requires insightful studies, which are left for the future.

Lastly, loads are normally accurately predicted an hour ahead and are less well known further down the planning period. Hence, treating only the first two hours as a stochastic process might yield the same result as treating the whole problem as a simple deterministic problem. However, as discussed before, a slight change in the initial value of external temperature (at Stage 1) could result in a significant weather load difference further in the future depending on the PMF of the external temperature. The relation between the external temperature and weather load is described in Chapter 2.

IX: Conclusion

9.1 Limitations of Nested Bender's Decomposition Method

There are several limitations of the applied Nested Bender's Decomposition method. First, it has exponential growth of memory requirements along with the increase in the number of stages as explained before. This high memory requirement limits the length of the planning horizon period dramatically. It is almost impossible to solve a year planning horizon period (= $num\_of\_children^{8760}$ hours). The reasons why memory grows in an exponential manner just like the growth of the number of nodes (in the tree of possible paths) is because each cut from each iteration specific for each node need to be saved in arrays. Moreover, besides the solution from the previous iteration, the changes in the right-hand-side value (Section 7.4) due to the state transition for each node also need to be stored in arrays.

Second, this algorithm carries a lot of overhead. This overhead mainly comes interfacing with CPLEX, such as: setting up the environment for the CPLEX, formatting the problem matrix into acceptable CPLEX input, getting solutions from CPLEX, and closing/returning CPLEX pointers after optimizing. The other major overhead comes from initializing and freeing dynamic arrays, reading the problem from an input file, and creating new hyperplane constraints. This high overhead affects the running time and even worse, increases the memory requirement. Thus, for a simple problem with a small number of stages (analogous to a knapsack problem) Dynamic Programming method will be faster. This is true especially in the deterministic case as discussed in Chapter 7.

Third, the algorithm requires running the backward recursion in the dual space, which is impossible to be done in our original mixed integer problem. This causes some consequences. The first one is there is no easy way to know ahead of time the error margin for the problem. As we have discussed, the upper bound and lower will not converge since the upper bound is calculated using MIP Solver and the lower bound using LP Solver to get new dual constraints. All we know is the result from LP is always lower than the result from MIP. The second consequence is the LP cuts effect, the ordering effect, and the relative objective cost function effect, which may result in a non-optimal solution as explained in detail in Chapter 7.

In addition, the sampling effect may skew the reading and the interpretation of the recommended strategies. Unfortunately, random sampling is inevitable for a large problem as explained before. The other limitation is encounter by the current formulation in solving Time-Varying Demand Charge.

Fourth, the average heat transfer coefficient does not embody the actual heat transfer process. It does not work measure the wind effect, environmental condition (for example shades from the next building), shape of the building, and the orientation of the building. Avoiding this inaccuracy of building load representation due to external environment is almost impossible. To include other environment factors such as wind and solar gain, another software like DOE can be used. However, this means that besides predicting the occupation behaviors and external temperatures, we need to predict the wind direction and speed (for the planning period), to predict the solar intensity and cloud factor (for the whole planning period), and to know exact geometry of the building. Moreover, running the DOE program itself takes a lot of time. Hence, it is impractical to get all the values of the building for the whole planning period every hour or two to run the proposed control system with the moving planning horizon window.

9.2 Concluding Remarks

In this thesis, the building load (Chapter 2) is divided into two components the weather load and occupation load. The occupation load consists of people's load, equipment load and lighting load. The weather load is derived using 2R1C circuit model. This building load model has several limitations explained before. A single generator model, with its basic constraints, is adopted. The finer constraints to capture the effects of start-up, ramp-up, shut-down, and ramp-down are touched briefly in Section 3.2.

In the beginning, Stochastic Programming using Nested Bender's Decomposition was chosen for several reasons (Chapter 4). First, Stochastic Programming, unlike Linear Programming, handles stochastic cases, which is important because building loads (and also RTP charge, although in this thesis RTP charge is assumed to be deterministic) are stochastic. Second, Stochastic Programming can handle a more complex problem (richer constraints and problem statements in general) than Dynamic Programming and also carry the value of information.

Third, Stochastic Programming avoids state explosion, which is the result of state discretization in Dynamic Programming.

However, there are some problems with the algorithm. First, as pointed out in examples from the previous chapters, the algorithm is to some extent unreliable in handling our MIP problem, especially the one with a long planning horizon period (more than 3 stages). Some anomalies due to our MIP formulation can be described as an LP cuts effect, an ordering effect, and a relative objective cost difference effect (Chapter 7). These effects can be reduced by making the MIP formulation such that its solution closely resemble those of LP, but in most cases this resemblance is almost impossible. The other reason for the unreliability is the sampling, which is referred to as sampling effect (Chapter 6).

The unreliability caused by the MIP formulation and by the random sampling is unpredictable, meaning it is unknown ahead of time whether the algorithm will yield an optimal solution or not, or how big the error is going to be for a particular problem. So far, the algorithm works well for a short planning horizon (with number of stages less or equal to three).

Second, as discussed previously, memory requirements are very high for the algorithm, making it almost impossible to run the full year planning horizon period. Moreover, the high overhead drives down the algorithm performance even further (Section 9.1).

There are some possible ways to avoid the inaccuracy of the algorithm due to the MIP formulation. The first solution is to use the Dynamic Programming method for the deterministic simple case. Simple means it does not has any complicated price structure such as demand charge or ratcheted demand charge or any fine generator constraints. However, Dynamic Programming limits the richness of the constraints and its states grow exponentially along with the increase of the number of stages. The second way is to use LP instead of MIP in the forward pass and to construct some kind of method to clean up the answer. For example if for all $z < 0.5$ we will round it down to 0 and if $z \geq 0.5$ round it up to 1. However, this heuristic method may give a suboptimal result as shown if it was applied to Example 6.1.2.

The third way is to combine the stochastic optimization method with the deterministic one as explained in Section 8.3. The proposed method reduces the problem into a two-stage problem, which fits well within the acceptable range of Bender's Decomposition method (as explained before the algorithm works well for number of stages less than or equal to three).

Moreover, by only discretizing the first two stages the memory requirement and the running time will be much more reasonable than those of the Stochastic Programming.

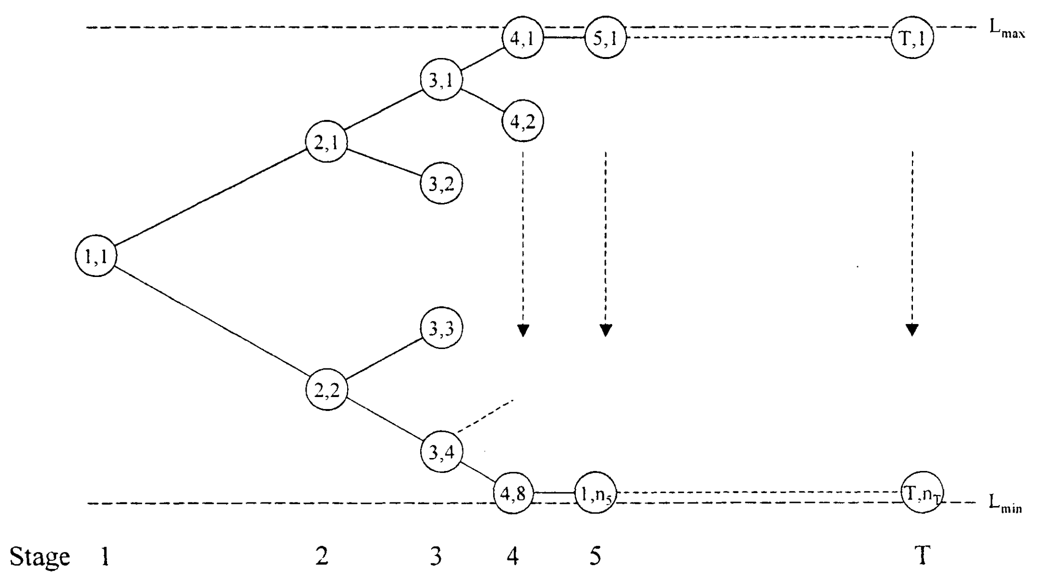
Lastly, Stochastic Programming works well for certain class of problems that has several properties. First, the problem should not have a very long planning horizon. For example, the scheduling of a hydrothermal generating system by Perreira and Pinto (1991) only has ten stages and two possible scenarios branching from each node (total possible paths is 512). This limit on planning horizon is due to the high memory requirement and the long computational time. Second, the problem that is normally solved with this Nested Bender's Decomposition method is an LP problem. An MIP problem is not suitable, because the algorithm requires recursion on the dual space. Third, the problem that is solved is a large-scale optimization problem with hundreds or thousands of constraints and variables. This stochastic large-scale problem cannot be solved with Dynamic Programming. Our problem is a small problem with several variables and several constraints. The proposed control system formulates the problem in this manner, where the second stage problems are composed of the rest of the hours in the year (several thousands variables and constraints).

9.3 Future Work

Several areas are open for further studies. The first one is the enhancement of the pricing structure. The ratcheted demand charge needs to be implemented next (some ideas on the formulation are presented in Appendix A). Moreover, the current formulation is applicable for the Time of Use (TOU) charges, however, it is yet to be implemented. For TOU, the formulation of the problem may need to be modified, especially to handle the complex variations of TOU such as stepped TOU with ascending and descending tariffs. However, there may be some ways that the code can be modified to handle simple TOU charges. For example: the *z_period* can be the TOU hour-block instead of a one-hour slot. This can help improve the performance of the program. In addition, there are other pricing models that can be incorporated into this algorithm such as: Customer Baseline Energy Charges, Retail Wheeling of Power from a Generator, etc.

The second area is the implementation of the proposed control system. Some simulations are necessary to compare the accuracy and computational efficiency with Stochastic Programming, Dynamic Programming, and Linear Programming. Moreover, since the proposed optimization engine is a hybrid between Stochastic Programming and Linear Programming, efficient procedures to formulate the problem into a two-stage stochastic problem (as proposed) need further studies.

The third area is the code optimization. The code can be optimized by avoiding duplications of subproblems. This can decrease the memory requirement and the actual running time of the algorithm, and thus make the algorithm more attractive. Another major optimization is to modify the code so that it handles the boundary cases of the scenario tree.



L_{min} = the minimum possible building load

L_{mz} = the maximum possible building load

n_i = number of possible scenarios in stage *i*

j, k = stage number, scenario number

Figure 9.1 Boundary Condition of the Scenario Tree with num_of_children=2 and 8 discretization components for *L*

The scenario tree stops expanding after *c* stages $(O(\log_{num\_of\_children}(\textit{number of Load discretization}) + 1)$ stages) (figure 9.1). Thus, after *c* stages, the increase on the number of nodes for each new iteration is reduced substantially. Handling the boundary condition can save a lot of memory space especially for a long planning horizon.

The fourth area is the effect of noise, meaning the impacts of forecasting uncertainty on the cost savings performance of a predictive optimal controller for auxiliary generator. There are four uncertainty models to be analyzed to predict future values of cooling loads (or weather variables): Unbiased Gaussian Noise, Correlated Gaussian Noise, Unbiased Uniform Noise, and Biased Uniform Noise (Henze *et al.* 1999b). The questions regarding the noise measurement are how to determine the constants (correlation constants, variance, weight scalar, bias scalar). Related to the noise effect is the effect due to the planning horizon. To improve the proposed control system, an optimal planning horizon needs to be determined. Some work has been done using thermal storage as the electricity costs saving parameter (Henze *et al.* 1999a).

The fifth area is the improvement of the sampling method. The code need to be modified to accommodate the PMF fir different building loads. The sampling procedure may need to be modified to address the PMF. For example, if the PMF is binomial then it should be more likely for the sampling procedure to choose the expected load compare to the extreme future load. This improvement may be able to reduce the sampling effect and thus, increases the accuracy of the algorithm.

The sixth area is the enrichment of the generator description. The current problem formulation assumes only one generator. We can extend the case with two generators with different capacities, for example. Then, we can go further by incorporating the amortized capital cost of the generator to give advice on the size and number of purchased generators. A study is necessary to incorporate the finer constraints: start-up, shut-down, ramp-up, and ramp-down of a generator.

$$z_k \geq \max\{z_{k-1} - z_{k-2}, z_{k-2} - z_{k-3}, \ldots, z_{k-n+1} - z_{k-n}\} \qquad (9.1)$$

subscript 'n' = minimum contiguous generator operating hour allocation

z_k = the decision to use the generator at hour k

Due to the transition behavior of the generator (Section 3.2), using the generator for one hour only is inefficient. This thesis provides the linear programming constraint (Equation 9.1), which can be used to prevent a stand-alone (scarcely fragmented) generator operating-hours allocation. However the optimal value of 'n' is still unknown and require further study.

APPENDIX A: Ratcheted Demand Charge

A1. Introduction to Ratcheted Demand Charge

Another variation of a One-month Period Demand Charge is the Ratcheted Demand Charge, where a present demand charge carries its effect into several periods later in the future (for example m months period). In other words, the charge is levied on the maximum of the demand levels set in the last m months.

At the future months within the effect, the demand charge will be the maximum of the previous demand charges that still have effects and that month present charge. This means that considerations for m future months must be taken when making a present decision since incremental usage in the current month will affect ratchets in the next m months.

A2. Linear Programming Problem Formulation

The last m months demand levels are known. Each future month has an associated expected demand charge. The ratchet level for the current month is bounded below by the maximum of the past m monthly demand charge and the present demand charge.

$$R_0 \geq D_{-1}, D_{-2}, \ldots, D_{-m}$$
$$R_1 \geq E[D_0], D_{-1}, \ldots, D_{-m+1}$$
$$\ldots$$
$$R_m \geq E[D_{m-1}], E[D_{m-2}], \ldots, E[D_0]$$
$$R_0 \geq x_0^{\max}$$
$$R_1 \geq x_0^{\max}$$
$$\ldots$$
$$R_m \geq x_0^{\max}$$

R_i = the ratchet at month i

D_i = the maximum power purchased from the utility company at month i

$x_0^{\max}$ = the current maximum at the current month

A3. Stochastic Programming Formulation

Those LP constraints above can be written into the Stochastic Programming statements of the main problem to include the new state transition equation of the ratcheted demand charge:

$P_1^i \bullet X_1^i$	$P_2^i \bullet X_2^i$	$P_s^i \bullet X_s^i + p\_ratchet \bullet R0_s^i$
$\dots$	$\dots$	$\dots$
$x\_\max_1^i \geq x_1^i$	$x\_\max_2^i \geq x_2^i$	$x\_\max_s^i \geq x_s^i$
$x\_\max_1^i \geq d_1^i$	$x\_\max_2^i \geq d_2^i$	$x\_\max_s^i \geq d_s^i$
$0 = d_1^i$	$x\_\max_1^i = d_2^i$	$x\_\max_{s-1}^i = d_{s-1}^i$
$D-1_1^i = x\_\max_0^i$	$D-1_2^i = D-1_1^i$	.... $D-1_s^i = D-1_{s-1}^i$
$D-2_1^i = D-1_0^i$	$D-2_2^i = D-2_1^i$	$D-2_s^i = D-2_{s-1}^i$
$\dots$	$\dots$	$\dots$
$D-m_1^i = D-m+1_0^i$	$D-m_2^i = D-m_1^i$	$D-m_s^i = D-m_{s-1}^i$
$R0_1^i \geq \{D-1_1^i,\dots,D-m_1^i\}$	$R0_2^i \geq \{D-1_2^i,\dots,D-m_2^i\}$	$R0_s^i \geq \{D-1_s^i,\dots,D-m_s^i\}$
$R0_1^i \geq x\_\max_1^i$	$R0_2^i \geq x\_\max_2^i$	$R0_s^i \geq x\_\max_s^i$
Stage 1	*Stage 2*	*Stage s*

$P_{s+1}^i \bullet X_{s+1}^i$	$P_2^i \bullet X_2^i$	$P_T^i \bullet X_T^i + p\_ratchet \bullet R0_T^i$
$\dots$	$\dots$	$\dots$
$x\_\max_{s+1}^i \geq x_{s+1}^i$	$x\_\max_{s+2}^i \geq x_{s+2}^i$	$x\_\max_T^i \geq x_T^i$
$x\_\max_{s+1}^i \geq d_{s+1}^i$	$x\_\max_{s+2}^i \geq d_{s+2}^i$	$x\_\max_T^i \geq d_T^i$
$0 = d_{s+1}^i$	$x\_\max_{s+1}^i = d_{s+2}^i$	$x\_\max_{T-1}^i = d_{T-1}^i$
$D-1_{s+1}^i = x\_\max_s^i$	$D-1_{s+2}^i = D-1_{s+1}^i$	.... $D-1_T^i = D-1_{T-1}^i$
$D-2_{s+1}^i = D-1_s^i$	$D-2_{s+2}^i = D-2_{s+1}^i$	$D-2_T^i = D-2_{T-1}^i$
$\dots$	$\dots$	$\dots$
$D-m_{s+1}^i = D-m+1_s^i$	$D-m_{s+2}^i = D-m_{s+1}^i$	$D-m_T^i = D-m_{T-1}^i$
$R0_{s+1}^i \geq \{D-1_{s+1}^i,\dots,D-m_{s+1}^i\}$	$R0_{s+2}^i \geq \{D-1_{s+2}^i,\dots,D-m_{s+2}^i\}$	$R0_T^i \geq \{D-1_T^i,\dots,D-m_T^i\}$
$R0_{s+1}^i \geq x\_\max_{s+1}^i$	$R0_{s+2}^i \geq x\_\max_{s+2}^i$	$R0_T^i \geq x\_\max_T^i$
Stage s+1	*Stage s+2*	*Stage 2s(=T)*

$\forall i = 1,\ldots,n$

T = the last stage which is the end of the period for the demand charge

s = the demand charge period (i.e. one month = 24*30 hrs)

m = the ratcheted demand charge period (it includes m last demand charges)

$R0_t^i$ = the current month ratcheted demand charge at time t, scenario i

$D - j_t^i$ = demand charge j month ago evaluated at time t, scenario i

In the formulation example above, the number of stages, T, is assumed to be twice the length of the demand charge period, s.

A4. Notes

The ratcheted demand charge model is an unfinished work. The formulation has not been verified, and thus, there is no guarantee on the correctness of the formulation.

References

ASHRAE, Inc. *1997 ASHRAE Handbook Fundamentals*. ASHRAE Inc. Atlanta GA, 1997.

Bertsekas, Dimitri. *Dynamic Programming and Optimal Control Vol. 1 & 2*. Athena Scientific. Belmont MA, 1995.

Bertsimas, Dimitris and Tsitsiklis, John. *Introduction to Linear Optimization*. Athena Scientific. Belmont MA, 1997.

Birge, John R. and Louveaux, Francois. *Introduction to Stochastic Programming*. Springer-Verlag. New York NY, 1997.

Birge, John R. Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *Operations Research* . Volume 33 Number 5. September-October 1985.

Braun, J.E. and Keeney, Kevin. A Simplified Method for Determining Optimal Cooling Control Strategies for Thermal Storage in Building Mass. *HVAC&R Research* . Volume 2 Number 1. January 1996.

Braun, J.E. Reducing Energy Costs and Peak Electrical Demand Through Optimal Control of Building Thermal Storage. *ASHRAE* . 1990.

Brignol, S. and Renaud, A. A New Model for Stochastic Optimization of Weekly Generation Schedules. *Fourth International Conference on Advances in Power System Control, Operation and Management Proceedings, IEE-APSCOM '97*. Volume 2 November 11-13, 1997, p.656-661.

Carpenter, P., Renaud, A., et al. Stochastic Optimization of Unit Commitment: a New Decomposition Method. *IEEE Transactions of Power System*. Volume 11 No.2, May 1996, p.1067-1073.

Cogdell, J.R. *Foundations of Electric Power*. Prentice Hall. New Jersey NJ, 1999.

Cote, Gilles and Laughton, M.A. Stochastic Production Costing in Generation Planning: A Large-Scale Mixed Integer Model. *Mathematical Programming Study*. North Holland Publishing Company. Volume 20 1982, p54-71.

Daryanian, Bahman. Scheduling of Electricity Consumption under Spot Prices. *PhD MIT Thesis*, 1989.

Daryanian, Bahman, Bohn, Roger E., and Tabors, Richard D. An Experiment in Real Time Pricing for Control of Electric Thermal Storage Systems. *Proceedings of IEEE/PES*. 1991.

Gabel, Steven D., Carmichael, Laurence, and Shavit, Gideon. Automated Control in Response to Real Time Pricing of Electricity . *ASHRAE Journal*. Vol 40 no 11 p.28. November 1998.

Grunenfelder WJ, Todtli J. The use of weather predictions and dynamic programming in the control of solar domestic hot water systems. [Conference Paper] *Proceedings of MELECON '85. Mediterranean Electrotechnical Conference (Cat. No.85CH2185-7). IEEE. 1985, pp.175-9 vol.4. New York, NY, USA.*

Henze, Gregor P., Krarti, Moncef, and H. Dodier. Development of a Predictive Optimal Controller for Thermal Energy Storage Systems. *HVAC&R Research 3*. 1997, p.128-148.

Henze, Gregor P., Krarti, Moncef, and Bell Dagfinn. Planning Horizon for a Predictive Optimal Controller for Thermal Energy Storage Systems. *ASHRAE Transactions*. 1999a.

Henze, Gregor P. and Krarti, Moncef. The Impact of Forecasting Uncertainty on the Performance of a Predictive Optimal Controller for Thermal Energy Storage Systems. *ASHRAE Transactions*. 1999b.

Heinemann D, Luther J. Control of renewable energy systems using Bayesian forecasting techniques and stochastic dynamic programming. [Conference Paper] *Clean and Safe Energy Forever. Proceedings of the 1989 Congress of the International Solar Energy Society.* Pergamon. 1990, p.407-11 vol.1. Oxford, UK.

Hiller, Randall Stephen. Stochastic Programming Approximation Methods with Application to Multistage Production Planning. *PhD Thesis, MIT*. August 1986.

Jacobs, J. et al. SOCRATES: a System for Scheduling Hydroelectric Generation under Uncertainty. *Annals of Operations Research*. Volume 59 1995, p.99-133.

Norford, L.K., Englander, S.E., and Wiseley, B.J. Demonstration Knowledge Base to Aid Building Operators in Responding to Real-Time Pricing Electricity Rates. *Research Project 833-RP, ASHRAE*, 1995.

Pereira, M.V.F and Pinto, L.M.V.G. Multi-stage Stochastic Optimization Applied to Energy Planning. *Mathematical Programming*. North Holland Publishing Company. Volume 52 1991, p.359-375.

Shanbag, Vinayak Vishnu. Optimal Control Systems in Response to Diverse Electricity Pricing Structures. *Master of Science Thesis, MIT.* February 1998.