An Algorithmic Study on Basic Planning Problems in Operations Research

by

Kakuzo Iwamura

Contents

Acknowledgments

First, I would like to thank Prof.Takehisa Fujisawa at Josai University for his encouragements to write out this dissertation. This was presented to " The Graduate School of Electro-Communications, the University of Electro-Communications " to get my late PhD degree in 2001. Without his encouragements, I would not be able to complete my work in this form. These days, every researcher knows it very well that *Integer Programming/Discrete Optimization without good structures* is very difficult to computationally solve, but in 1970s and up to the middle of 1980s they are not sure about it, at least in Japan. Up to the early years of 1980s, we saw lots of papers on solving Integer Programming problems and/or presenting algorithms to solve the general integer programming problems. So along with them, my younger days as a researcher was used up in devising good exact algorithms,i.e., dynamic programming algorithm, branch and bound type algorithm and so on. Most of them are not yet open, because they don't think that those results are important. Yet, I am satisfied enough that I can publish some of my work as one of Josai Mathematical Monographs through Prof. Masayuki Yamasaki.

Second, I would like to express my thanks to my colleagues abroad, Prof. Baoding Liu at Tsinghua University, Beijing and his young students. They are eager to study production programming and decision making, eager to know what was wrong with me, the reason I was left backwards in this fields and how the things happened and how the things will go. I hope decision making methods in this field will be practically improved in the future by some Asian young researchers. To my colleagues in Japan, i.e., Prof. Norio Okada, Prof. Yozo Deguchi at Josai University, Prof. Makoto Horiike at Teikyo University and researchers at the Bellman Continuum Japan Section, I would like to express my grateful thanks.

Last, but not least, I express my hearty thanks to my wife, Kazue, for her never ending support for my study. My daughter Eri and my son Akira also played their roles as my supporters. I say I am happy with my family.

February 28, 2003
Kakuzo Iwamura

Abstract

In this dissertation, we investigate two basic planning problems in Operations Research, non-probabilistic one and probabilistic one, through realistic planning problems. The first one is a Combinatorial Planning Model , which can be solved through algorithms of Deterministic Turing Machine. The second one is a natural model for Planning under Uncertainty. In Chapter 1, we give our motif and outline of this dissertation. In Chapter 2, we treat Combinatorial Planning Problems which at present don't give us any kind of polynomial-time algorithms. In Chapter 3, we treat Combinatorial Planning Problems which give us polynomial-time algorithms. In Chapter 4, we treat Planning Problems under Uncertainty. In Chapter 5, we apply a genetic algorithm to a problem which belongs to Combinatorial Planning Problems in Chapter 2. In Chapter 6, we state the profits and losses of the two methods, non-probabilistic and probabilistic, in model buildings and algorithms based on the author's thirty year experiences. Then we confirm that probabilistic treatments like genetic algorithm will be more important in the future.

Chapter 1

Introduction

1.1 Motivation

In this dissertation, we would like to investigate basic planning problems in Operations Research. The first one is a *Combinatorial Planning Model*, in which we see that there lies a great gap between a model which at present doesn't give us any kind of polynomial-time algorithms (**Chapter 2**) and the other model which allows us polynomial- time algorithms (**Chapter 3**) . This gap is a well-known P vs. NP problem. The second one is a natural model for Planning under Uncertainty. We call it *Uncertain Programming*, because we are going to make a plan even if we are not sure of the exact values of the input problem data itself. Uncertainty may come from the ambiguity of the input data, the lack of a number of data,or the fact that some data must obey legal regulations and so on(**Chapter 4**). In **Chapter 5** we treat the Combinatorial Planning Problem in the framework of Uncertain Programming, i.e. , Genetic Algorithm. Genetic Algorithm has been successfully applied to some kinds of Combinatorial Problems. Yet, here we have tried to combine Genetic Algorithm and Domain Specific Knowledge which we can get through Mathematical Programming. Finally in **Chapter 6**, we summarize our results through summing up both theoretical and computational considerations.

Let's see in a little detail what I have experienced since 1970. In 1970, I got a job in Mitsubishi Research Institute. There, I was ordered to make some programs in Operations Research. They were

1. Travelling Salesman problem(TSP),

2. Capacitated Facilities Location Programming (CFLP) problem

and other statistical problems. It is very easy to describe the Traveling Salesman problem. Given N cities in a country. Let the coordinates of the city i be

$(x_i, y_i)(1 \leq i \leq N)$ and define $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}(1 \leq i, j \leq N)$. A salesman visits all the N cities once and only once to sell his commodities. He makes an itinerary, or a *tour* and visits city i_1 first, city i_2 second, $\cdots$, city i_N last and then comes back to city i_1. A tour t will be denoted by $t = (i_1, i_2, \cdots, i_N, i_{N+1})$, where $i_{N+1} = i_1$. Then solve

minimize

$$\sum_{j=1}^{N} d_{i_j i_{j+1}} \tag{1.1}$$

subject to

$$(i_1, i_2, \cdots, i_N, i_{N+1}) : \quad tour. \tag{1.2}$$

This is a *Euclidean Traveling Salesman* problem with N cities. See Figure 1.1 just below. We can easily see that there are $(n-1)!/2$ tours for the*Euclidean Traveling Salesman* problem. $(n-1)!/2$ is much greater than 2^n which is not a polynomial function in n. It is well known that *Traveling Salesman* problem is NP-complete(See,Garey and Johnson[44],Lawler et al.[137],Reinelt[172],Yamamoto and Kubo[201], Trevisan[194]).

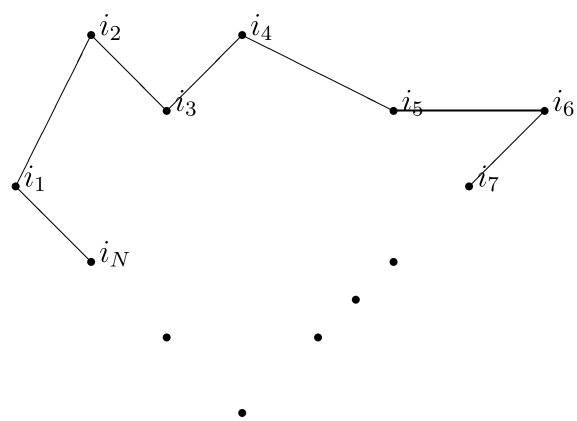

Figure 1.1: A Euclidean TSP with N cities

CFLP problem is expressed as

minimize

$$\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i \tag{1.3}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le c_i y_i \quad (1 \le i \le m) \tag{1.4}$$

$$\sum_{i=1}^{m} x_{ij} \ge d_j \quad (1 \le j \le n) \tag{1.5}$$

$$\sum_{i=1}^{m} y_i = l \tag{1.6}$$

$$x_{ij} \ge 0 \tag{1.7}$$

$$y_i = 0 \quad or \quad 1, \tag{1.8}$$

where input data t_{ij} stands for unit transportation cost from potential depot location i to demand destination j, f_i denotes fixed construction cost at potential depot location i when we determine to construct a new depot at location i, c_i is a maximum capacity of commodity at potential depot location i, d_j is a commodity demand at j, l is the total number of new depots that we have to construct. Furthermore, x_{ij}, y_i are decision variables such that

$$x_{ij} = \quad \text{amount of commodities to be transported from } i \text{ to } j,$$

$$y_i = \begin{cases} 1, & \text{if we construct a new depot at } i \\ 0, & \text{if we don't construct a new depot at } i. \end{cases}$$

We devised an excellent computer program to solve the real world CFLP problem with another efficient heuristic one. Computational results of our computer programs can be found partly in Mukawa,H., Sensui,J., Iwamura,K. and J.Kase[161]. As for the efficient heuristic algorithm for the CFLP Problem, the readers can consult Sorimachi,Y.[188]. They worked very well for a real world input data.

On the contrary, we were not able to develop an efficient computer program to solve the TSP for a real world data. Through IBM 360/370, the fastest computer in 1970 in the world, our algorithm using a minimum spanning tree bound needed more than ten minutes for a TSP problem instance(input data) sized 30 cities. At that time it meant that it cost more than 300 thousand yen to solve a 30 city TSP and so a 100 city TSP is far from being able to be solved. Furthermore its computing time varied drastically as the problem instances changed. This was a great surprise for me because I faithfully implemented an algorithm of Little,J.D.C. et al.[142] with the most professional programming techniques given by Mr. H. Mukawa. Our computer program showed both exponential computing time and its computing time dependency on each problem instance. We were shocked by its computational inefficiency and its unpredictable computing time. " Oh! What a shock. Little et al.'s algorithm got the prize because it solved a US state capital TSP problem. Yet, it showed a poor computing efficiency. *A great difference from the CFLP.* But why? " At that time, researchers considered that there must be an excellent algorithm to solve the TSP for all kinds of input data. Today we can see the true nature of the TSP in Applegate, Bixby, Chvatal and Cook[6] as follows:

Table 1.1: Computing Time of an Exact Algorithm in [6]

Name	Cities	Tree of subproblems	Running time
gr120	120	1 node	3.3 seconds
lin318	318	1 node	24.6 seconds
pr1002	1002	1 node	94.7 seconds
gr666	666	1 node	260.0 seconds
att532	532	3 nodes	294.3 seconds
pr2392	2,392	1 node	342.2 seconds
ts225	225	1 node	438.9 seconds
pcb3038	3,038	193 nodes	1.5 days
fnl4461	4,461	159 nodes	1.7 days
pla7397	7,397	129 nodes	49.5 days
usa13509	13,509	9,539 nodes	about 10 years

After finding a job at the Dept. of Math. , Josai University, I started a theoretical research activities in Integer Programming. I found a knapsack typed integer programming problem computationally greedily solvable through D.P.technique. Then, I changed my interest to Set-covering/Set- partitioning problems. At that time, still now I think, there was a rumor that general algorithms for the linear integer programming problems such as Gomory's fractional integer programming algorithm are inefficient for real world data. So, it was the time to try to invent a specific algorithm to solve a specific problem such as the Set- partitioning problem. Furthermore it has a wide application in transportation engineering. For example, air line crew scheduling, bus routing, truck dispatching and so on. I made a great effort to develop an efficient computer algorithm to solve the Set-Partitioning Problem. I paid every attention to make the computer program as fast as possible. It amounted over 600 pages documents to devise three computer programs to solve the Set-Partitioning Problem. The interested readers can have information through Suzuki, H. and K.Iwamura[190] , Iwamura,K. and Y. Maeda[101][102] ,Maeda, E. and K.Iwamura[153]. Still, our computational experiments showed the same aspect as I had experienced in developing computer programs to solve TSP. **Exponential computing time and a heavy data dependent computing time.** One data was solved by one algorithm efficiently while another data of the same size wasn't solved in one week through FACOM 230-38S. Of the three algorithms, I was not able to say that this one was always superior to that one.

Then I changed my interest to problems which gave us a polynomial time algorithm. I got three results in greedoid and greedy algorithms.

And furthermore I have changed my mind to Planning Problems under Uncertainty, because I knew it very well that there were some cases in which an OR Researcher had to make a decision, independent of the fact if he had enough and complete input data or not. Here I have mainly adopted Genetic Algorithm to carry out computing jobs.

Finally I have co-worked with Prof.Fushimi, Prof. Morohoshi and my student Mr. Shibahara to combine Genetic Algorithm and Domain Specific Knowledge of the problem itself. Here we combine stability in computing time of Genetic Algorithm with better individuals in the starting generation mathematical programming analysis can wisely produce.

1.2 An Overview

Here, we would like to see how the author's motive leads to results. We treat nine problems in all, two in **Chapter 2**, three in **Chapter 3**, three in **Chapter 4** and one in **Chapter 5**.

In **Chapter 2 section 1**, we discuss on some theorems of knapsack problem. Extending the solution procedure proposed by Dreyfus, S.E. and K.L. Prather[28], we devise a solution procedure of an ILP with all the constants nonnegative integer which is viewed as an ILP of knapsack type. We provide the validity proof of this procedure. In 1966, Gilmore, P.C. and R.E. Gomory[50] found the periodicity of the knapsack function for the first time. Hu, T.C.[66] stated this fact with a complete proof in the case $\rho_1 > \rho_2$. We also prove this fact in the case

$$\rho_1 = \cdots = \rho_k > \rho_{k+1} \geq \cdots \geq \rho_n.$$

This with the proof of Hu, T.C. offers an elementary but complete proof for the periodicity property of an arbitrary knapsack function. We also make an effort to find a small knapsack length b from which knapsack function has periodicity property.

In **Chapter 2 section 2** we revise Dual All Integer Algorithm when we apply it to solve Set Partitioning Problem. Historically, researchers transformed Set Partitioning Problem into Set Covering Problem, thus enlarging the column size of the problem from n to $n+m$ (See, Garfinkel and Nemhauser[46]). Enlarging the problem size directly leads to more computing time needed which is a disaster to decision makers. In our treatment, we save both computing time and in-core memory size. This kind of improvement is important, because this kind of improvement is also possible for almost all kinds of *NP-hard* Combinatorial Problems with linear constraints.

After brief introduction of Greedoid in **Chapter 3 section 1** , we treat a problem to get a lexicographically optimal base of a submodular system with respect to a positive weight vector(**Chapter 3 section 2**).

We show and prove the existence and uniqueness of the lexicographically optimal base. Then, we propose an algorithm to get the lexicographically optimal base. This algorithm completely differs from the one Fujishige[36] proposed for a polymatroid. We show that the lexicographically optimal base of a positive submodular system is the unique optimal solution of the corresponding $p(> 1)$-dratic separable mathematical programming problem. Finally, we see that the first problem of Morton, von Randow and Ringwald [160] can be captured and solved within our framework.

In **Chapter 3 section 3**, we first investigate the reason the greedy algorithm in **section 2** proceeds inversely. We see that the lexicographically minimum base of the dual supermodular polyhedron coincides with the lexicographically maximum base of the submodular polyhedron. Showing an algorithm(dual) to get the lexicographically minimum base answers the above mentioned problem. We also see that the lexicographically maximum base of a simplification of a submodular polyhedron can be changed to that of the original submodular polyhedron through proportional weighting. The same fact holds for an expansion of a given simple submodular polyhedron.

In **Chapter 3 section 4**, we show that the greedy algorithm over greedoid is a special case of a discrete decision process model. Therefore there is a possibility that we have some other equi-maximal cardinal set systems with its objective functions for which a greedy type algorithm works.

Although not included in this dissertation, we got another application of greedy algorithm over greedoid. That is " Drawing a tree on parallel lines" (See, [104][61]). In this application a greedoid reduces to a shelling structure. A matroidal approach to the tree drawing problem on parallel lines appeared in the thesis of Mr. Fukuhara who was a student of Prof. Kajitani at Tokyo Institute of Technology. Yet it produced no algorithms to solve this problem(See,[41](1990)). Treating the problem from a greedoidal point of view, we have devised both two polynomial time tree drawing heuristic algorithms and the exact polynomial time algorithm.

The need for *Uncertain Programming* is almost clear. In **Chapter 4 section 2** we give *A Genetic Programming for Chance Constrained Programming.* We show how we pose the problem. Then we show how we invent an algorithm to solve this problem. We give computational results for two examples from the literature. We also give computational results for *Stochastic Resource Allocation Problem* and *An Abstract Example*.

In **Chapter 4 section 3** we present *Chance Constrained Integer Programming Models for Capital Budgeting in Fuzzy Environments*, where uncertainty comes from fuzziness/possibility.

In **Chapter 4 section 4** we treat *Topological Optimization Models for Communication Network with Multiple Reliability Goals* under our *Uncertain Programming Philosophy*. We have found that our treatment was successful.

In **Chapter 5** we have tried to solve the well-known ,yet notorious Set Covering Problem by Genetic Algorithm. As already stated out , we took careful consideration for the first population of our Genetic Algorithm using LP like information. We carefully implemented our algorithm. We used bitwise representation to store coefficient matrix information. Computational results are given for three small, yet meaningful input data and two randomly generated medium sized input data.

In **Chapter 6** we summarize our results from both model-building and methodological point of view.

Chapter 2

Combinatorial Models

2.1 Knapsack Problem

Historically, Knapsack Problem was considered to be the easiest Combinatorial Problem. It was also considered one specific problem in Integer Programming. Among the all integer linear programming (abbreviated as ILP), integer linear programming with nonnegative integer constants (including coefficients of constraint matrix) , one constrained, is called *Knapsack Problem* [28][47][48][49][50][184]. Knapsack Problem with all the variables restricted to 0-1 is sometimes called *0-1 Knapsack Problem* [152][179]. In [82], the author reported a solution procedure of an ILP having a similar structure with 0-1 Knapsack Problem. Here in subsection 1, the author will show, extending the solution procedure proposed by Dreyfus, S.E. and K.L. Prather [28], a solution procedure of an ILP with nonnegative integer constants which is of Knapsack type. Theorem 1 and Theorem 2 provide the validity of this procedure which was not given in [28] even for the Knapsack case. Although the periodicity of the Knapsack function was for the first time found by Gilmore, P.C. and R.E. Gomory [50], it is hard to follow up their proof (see [56]). In [66] Hu, T.C. stated this fact with a complete proof in the case $\rho_1 > \rho_2$[1] . We also prove this fact in Theorem 4 in subsection 2, in the case of

$$\rho_1 = \cdots = \rho_k > \rho_{k+1} \geq \cdots \geq \rho_n.$$

This with the proof of Hu, T.C. offers an elementary, but complete proof for the periodicity property of any Knapsack Function. We also tried to find the small Knapsack length b from which Knapsack Function has periodicity property. This section comes from K.Iwamura[83].

Example We have a company named ENHANCE-PROJECT-EFFICIENCY(EPE). EPE has limits on its capital and labour power. Its capital is limited within

[1] The meaning of ρ_i is given in Assumption 5.

8 units, whereas its labour power is limited to 4 units at the most. Now at hand, it has 3 profitable projects named A,B,C. Project A uses 4 capital units, 1 labour unit and produces 6 returns. Project B uses 3 capital units, 2 labour units and produces 9 returns. Project C uses 3 capital units, 3 labour units and produces 7 returns. EPE is allowed to open plural A projects and/or projects A and C altogether and so on. Then find a total project plan which produces maximum total returns under its limits on capital and labour power. Letting EPE opens x_A - A projects, x_B-B projects, x_C-C projects, EPE has to solve the following ILP(Integer Linear Programming) of Knapsack type;

maximize

$$6x_A + 9x_B + 7x_C$$

subject to

$$\begin{gathered} 4x_A + 3x_B + 3x_C \leq 8 \\ 1x_A + 2x_B + 3x_C \leq 4 \\ x_A, x_B, x_C : \text{ non-negative integers.} \end{gathered}$$

$\square$

2.1.1 ILP with Nonnegative Integer Constants

Notations. Let $N = \{1, 2, 3, \cdots\}$ be the set of natural numbers, $N_0 = N \cup \{0\}$, $I = \{\cdots, -1, 0, 1, \cdots\}$ the of integers, $N_0^n = N_0 \times \cdots \times N_0$ n-fold direct product of N_0, I^m= m-fold direct product of I for $m, n \in N$. Notation $x \geq 0$ *integer* means *each component of x is nonnegative integer*. And iff means *if and only if*.

We concentrate on the following ILP with b regarded as a parameter, $b \in I^m$.

$F(b)$: $\max cx$ subject to $w_i x \leq b_i (1 \leq i \leq m)$, $x \geq 0$ integer, where $c = (c_1, \cdots, c_n)$, $x = (x_1, \cdots, x_n)^T$, $w_i = (w_{i1}, \cdots, w_{in})$, $b = (b_1, \cdots, b_m)^T$ and T denotes transpose operation.

As an ILP with nonnegative integer constants, we assume

Assumption 1. $c_j, w_{ij} \in N_0$ $(1 \leq i \leq m, 1 \leq j \leq n)$ and all these constants including $m, n \in N$ are fixed.

Therefore optimal objective function value of $F(b)$ is a function of b. So we define

Definition 1.

$$f(b) = \begin{cases} \max cx \text{ subject to } w_i x \leq b_i (1 \leq i \leq m), x \geq 0 \text{ integer}, & b \in N_0^m \subset I^m \\ -\infty, & b \in I^m \setminus N_0^m. \end{cases}$$

Definition 2. $x \in N_0^n$ is called a feasible solution for $F(b)$ (or simply, feasible) iff $w_i x \le b_i (1 \le i \le m)$ and is called an optimal feasible solution for $F(b)$ (or simply, optimal) iff x is feasible and attains $f(b)$ (i.e., $cx = f(b)$).

If $w_{ij_0} = 0 (1 \le i \le m)$ and $c_{j_0} = 0$ we can disregard x_{j_0} in the definition of $f(b)$. Moreover if $w_{ij_0} = 0 (1 \le i \le m)$ and $c_{j_0} > 0$ we can make $f(b)$ as large as possible. Therefore, hereafter we can assume without loss of generality

Assumption 2. For any $j (1 \le j \le n)$ there exists $i (1 \le i \le m)$ such that $w_{ij} > 0$.

And for any optimal x, if there exists j_0, $c_{j_0} = 0$ then

$$(x_1, \cdots, x_{j_0-1}, 0, x_{j_0+1}, \cdots, x_n)^T$$

is also optimal. So we can assume without loss of generality

Assumption 3. $c_j > 0$ for any $j (1 \le j \le n)$.

Next we define (See [3])

Definition 3. $[w_{ij}, \infty) = \{w : w_{ij} \le w$ and w is a real number$\}$, $w_{\cdot j} = (w_{1j}, \cdots, w_{mj})^T$, $\prod_{i=1}^{m} [w_{ij}, \infty) = m$-fold direct product of $[w_{ij}, \infty) (1 \le i \le m)$. And we call $b \in I^m$ breakpoint and write b: b.p. iff

$$f(b) > \max_{1 \le i \le m} f((b_1, \cdots, b_{i-1}, b_i - 1, b_{i+1}, \cdots, b_m)^T).$$

Definition 4. Write b: n.b.p. iff b is not a breakpoint.

From Assumptions 1,2,3, $0 \le f(b)$ for any $b \in N_0^m$. Moreover

Lemma 1. (1) $f(b) \ge \max_{1 \le i \le m} f((b_1, \cdots, b_{i-1}, b_i - 1, b_{i+1}, \cdots, b_m)^T)$ for any $b \in N_0^m$.

(2) $0 = (0, \cdots, 0)$: b.p.

(3) If $N_0^m \ni b$: b.p. then for any x: optimal , $w_i x = b_i (1 \le i \le m)$.

Lemma 2. For $b \in N_0^m$, $w_{\cdot j} \le b$ for some j iff $f(b) > 0$.

Remark. $w_{\cdot j} \le b$ for some j is equivalent to $b \in \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$. So for $b \in N_0^m$, $b \in \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$ (respectively $b \notin \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$) iff $f(b) > 0$ (respectively $f(b) = 0$).

Lemma 3. For $b \in N_0^m$, if $f(b) > 0$ then $f(b) = \max_{1 \le j \le n} \{f(b - w_{\cdot j}) + c_j\}$.

In order to obtain $f(b)$ for b in $\bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$, we may proceed step by step the origin owing to Lemma 3. But the following Lemma 4 enables us to go back only through the breakpoints.

Lemma 4. For any b: b.p. & $b \in \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$, we have

$$f(b) = \max_{1 \le j \le n, b - w_{\cdot j}: b.p.} \{f(b - w_{\cdot j}) + c_j\}.$$

Proof. $f(b) = \max_{1 \le j \le n} \{f(b - w_{\cdot j}) + c_j\}$ by Lemmas 2 and 3. Suppose that the maximum is attained at $b - w_{\cdot j}$: n.b.p. then there exists i such that (2) $b1 = (b_1, \cdots, b_{i-1}, b_i - 1, b_{i+1}, \cdots, b_m)^T$, $f(b) = f(b - w_{\cdot j}) + c_j = f(b1 - w_{\cdot j}) + c_j$.

On the other hand b is b.p. so $f(b) > f(b1) > 0$ (for, if $f(b1) = 0$ then one of $b_i - 1 - w_{ij}$, $b_k - w_{kj} (k \neq i)$ becomes negative, so that $0 < f(b) = -\infty + c_j < 0$. Contradiction.). Applying Lemma 3 for $b1$ we obtain

$$f(b) > f(b1) = \max_{1 \le l \le n} \{f(b1 - w_{\cdot l}) + c_l\} \ge f(b1 - w_{\cdot j}) + c_j$$

and by (2) $f(b1 - w_{\cdot j}) + c_j = f(b)$ contradiction. Q.E.D.

In order to obtain a solution procedure, we define and prove

Definition 5. For $b \in N_0^m$ let

$$R(b) = \{y : y \in N_0^m, y_i \le b_i (1 \le i \le m)\} \setminus \{b\}$$

and

$$B(b) = \{y : y \in R(b) \& y : b.p.\}$$

(Note that $b \notin R(b)$ and $b \notin B(b)$.)

Theorem 1. Assume that we have calculated $B(b_0)$ and $f(b)$ for all $b \in B(b_0)$.

(1) If b_0: b.p. then we can get the optimal value $f(b_0)$ and optimal solution which gives $f(b_0)$ by Lemma 4.

(2) If b_0: n.b.p. then finding $b_{\max}$ such that

$$f(b_{\max}) = \max_{b \in B(b_0)} f(b)$$

we see that $f(b_{\max}) = f(b_0)$. So for the optimal solution which gives $f(b_0)$ we can take that of $b_{\max}$.

Proof. First part is Lemma 4 itself. If b_0: n.b.p. then after finding $b_{\max}$ as stated in Theorem 1, we see that there exist no b.p. in

$$\{y : y \in N_0^m, (b_{\max})_i \le y_i \le (b_0)_i (1 \le i \le m)\} \setminus \{b_{\max}\}.$$

For, if there existed $\hat{b}$ then from the definition of b.p.

$$f(b_{\max}) < f(\hat{b}), \quad \hat{b} \in B(b_0)$$

which contradicts the definition of $b_{\max}$. Moreover as b_0: n.b.p. $f(b_{\max}) = f(b_0)$. Q.E.D.

Definition 6. Provided that $B(b_0)$ has been calculated and b_0: b.p., we call

$$PB(b_0) = \{b : b - w_{\cdot j} \in B(b_0) \cup \{b_0\} \text{ for some } j(1 \leq j \leq n)\} \setminus (B(b_0) \cup \{b_0\})$$

the set of potential breakpoints generated by b_0. And we call the element of $PB(b_0)$, potential breakpoint (p.b.p.) for b_0.

The meaning of p.b.p. will be clear. We can easily obtain

Lemma 5. There exist no b.p. in

$$(N_0^m \setminus \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)) \setminus \{0\}$$

(that is there exist no p.b.p. for the origin in this set.).

Theorem 2. Assume that $B(b)$ has been calculated for $b \in \bigcup_{j=1}^{n} \prod_{i=1}^{m} [w_{ij}, \infty)$. b: b.p. iff

$$f(b) = \max_{1 \leq j \leq n, b - w_{\cdot j} \in B(b)} \{f(b - w_{\cdot j}) + c_j\} > \max_{y \in B(b)} f(y).$$

Remark. Under the assumption of Theorem 2, we see that if

$$f(b) = \max_{1 \leq j \leq n, b - w_{\cdot j} \in B(b)} \{f(b - w_{\cdot j}) + c_j\} \leq \max_{y \in B(b)} f(y)$$

then b: n.b.p..

Proof. *If* part is trivial. *Only if* part; for $y \in B(b)$, $y_i \leq b_i (1 \leq i \leq m)$ so that $f(y) \leq f(b)$. As b : b.p. and $b \neq y$, $f(b) > \max_{y \in B(b)} f(y)$. Q.E.D.

Definition 7. As usual for $b, b' \in N_0^m$, we say that b is lexicographically equal or smaller than b' iff $b = b'$ or $b_1 < b'_1$ or there exists $k(1 \leq k \leq m-1)$ such that $b_1 = b'_1, \cdots, b_k = b'_k, b_{k+1} < b'_{k+1}$. And the set of current potential breakpoint $CPB(b_k)$ be such that $CPB(b_k) = PB(b_k) \setminus (\{\text{previous established b.p.}\} \cup \{\text{previously established n.b.p.}\})$.

Concluding all the preceding results, we obtain a solution procedure for an ILP with nonnegative integer constants.

An Algorithm to solve an ILP with Nonnegative Integer Constants

Step 1. If Ass.1-Ass.3 is not satisfied, then go to Step 8.

Step 2. Set $k := 1, b_k := 0$(zero vector), $B(b_k) := \emptyset$ and mark $b_1 : b.p.$

Step 3. Calculate the set of current potential breakpoint $CPB(b_k)$.

Step 4. If $CPB(b_k) \cap (R(b) \cup \{b\}) = \emptyset$, then go to Step 6.

Step 5. Calculate b_{k+1} by $b_{k+1} :=$ lexicographically minimum of $y \in CPB(b_k) \cap (R(b) \cup \{b\})$. If

$$\max_{b_{k+1}-w_{\cdot j} \in B(b_{k+1})} \{f(b_{k+1} - w_{\cdot j}) + c_j\} > \max_{y \in B(b_{k+1})} f(y),$$

then markb_{k+1}:b.p., set $k := k+1$ and go to Step 3. Otherwise set $CPB(b_k) := CPB(b_k) \setminus \{b_{k+1}\}$ and go to Step 4.

Step 6. Find b_{max} by

$$f(b_{max}) = \max_{b' \in B(b_k) \cup \{b_k\}} f(b').$$

Step 7. Take the optimal solution of b_{max} as that of b.STOP.

Step 8. Print out the adequate message. STOP.

It is easy to check that this procedure reduces to that of Dreyfus and Prather when $m = 1$.

Example. $m = 2$, $n = 3$, $c = (6, 9, 7)$, $w_1 = (4, 3, 3)$, $w_2 = (1, 2, 3)$, $b = (8, 4)^T$. In the following Table 2.1

$$M_1 = \max\{f(b_{k+1} - w_{\cdot j}) + c_j : 1 \leq j \leq n, b_{k+1} - w_{\cdot j} \in B(b_{k+1})\}$$

and

$$M_2 = \max\{f(y) : y \in B(b_{k+1})\}.$$

It is evident to determine $CPB(b_k)$ from already generated potential breakpoints and the elimination rule Solution Procedure indicates. The signal in the right upper corner of p.b.p. means as follows.

S1 (α) : eliminated from iteration $\alpha - 1$ to α because this is established to be a b.p..

S2 * : eliminated because this is not in $R(b) \cup \{b\}$,i.e., infeasible.

S3 $^{**}_{\alpha}$ eliminated from iteration $\alpha - 1$ to α because this is revealed to be a n.b.p..

At iteration 5 calculation is stopped because $CPB(b_5) \cap (R(b) \cup \{b\}) = \phi$ with the optimal solution $x_1 = 0, x_2 = 2, x_3 = 0$ and the optimal value $f(b) = 18$.

Table 2.1: An Example to Show How our Algorithm Works

k	Previously established breakpoint b_k	$B(b_k)$	p.b.p. generated at iteration k i.e., $b_k + w_{.j}$ $j=1$	$j=2$	$j=3$	b_{k+1}	M_1	M_2	$f(b_{k+1})$	Optimal feasible solution for b_{k+1} when b_{k+1} : *b.p.*
1	0 0	$\emptyset$	4(3) 1	3(2) 2	3_2^{**} 3	3 2	9	0	9	0 1 0
2	3 2	0 0	7(4) 3	6(4) 4	6^* 5	3 3	7	9		
						4 1	6	0	6	1 0 0
3	4 1	0 0	8(6) 2	7(4) 3	7_5^{**} 4	6 4	18	9	18	0 2 0
4	6 4	0,3,4 0,2,1	10^* 5	9^* 6	9^* 7	7 3	15	9	15	1 1 0
5	7 3	0,3,4 0,2,1	11^* 4	10^* 5	10^* 6	7 4	13	18		
						8 2	12	9	12	2 0 0

2.1.2 Periodicity Property of Knapsack Function

As $m = 1$ we write $w_{\cdot 1} \equiv w = (w_1, \cdots, w_n)$. Assumption 1 – Assumption 3 reduce to

Assumption 4. $c_j, w_j \in N (1 \leq j \leq n)$
and the problem

$$F(b) : \max\{cx : wx \leq b, x \geq 0 \text{ integer}\},$$

where we set

$$f(b) = \max\{cx : wx \leq b, x \geq 0 \text{ integer}\}$$

which is called Knapsack function[50].

Example Company EPE is asked from a tiny jeweler to let him know how to cut off a 33cm long gold bar. He can make a green jewel using 5cm with profit 5 thousand yen. A black jewel using 7cm with profit 7 thousand yen. A purple one using 13cm with profit 6 thousand yen. A blue one using 11cm with profit 5 thousand yen. And that's all he can make from the 33cm long gold bar. To answer for the jeweler, EPE just solves the following (one dimensional) Knapsack Problem;

maximize

$$5x_1 + 7x_2 + 6x_3 + 5x_4$$

subject to

$$5x_1 + 7x_2 + 13x_3 + 11x_4 \leq 33$$

$$x_1, x_2, x_3, x_4 \geq 0 (integers)$$

, where the jeweler is advised to produce

$$x_1 - \text{ green jewel,}$$

$$x_2 - \text{black jewel,}$$

$$x_3 - \text{ purple jewel,}$$

$$x_4 - \text{ blue jewel.}$$

□

Lemma 3 reduces to

Lemma 6. For $b \geq w_\alpha \equiv \min_{1 \leq j \leq n} w_j$, there exists optimal x with $x_{j_0} > 0$ iff

$$f(b) = f(b - w_{j_0}) + c_{j_0}.$$

Following the line of Hu, T.C. [66], we can assume without loss of generality

Assumption 5. $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$ where $\rho_j = c_j/w_j (1 \leq j \leq n)$.

Remark. Under this assumption either $\rho_1 > \rho_2 holds$ or there exists $k(2 \leq k \leq n)$ such that

$$\rho_1 = \cdots = \rho_k > \rho_{k+1} \geq \cdots \geq \rho_n.$$

Definition 8. For $s \in N_0$ let

$$f(b; LEs) = \max\{cx : wx \leq b, x \geq 0 \text{ integer}, x_1 \leq s\}.$$

With this definition we can prove in a same way as Hu did when $s = 0$ [66],

Lemma 7. In the case $\rho_1 > \rho_2$, for $s \in N_0$, $b \in N$, if

$$b \geq \rho_1 w_1/(\rho_1 - \rho_2) + w_1 s$$

then $f(b; LEs) < f(b)$ (that is, for any optimal $\overline{x}$ for $F(b)$, $\overline{x}_1 > s$). Until we arrive at the end of this section, let us assume and define

Assumption 6. $\rho_1 = \cdots \rho_k > \rho_{k+1} \geq \cdots \geq \rho_n (2 \leq k \leq n, n \geq 2)$.

Definition 9. Let a be the greatest common divisor of $w_1. \cdots, w_k$ and d_i be such that $w_i = a d_i (1 \leq i \leq k)$ (Note that $d_1, \cdots, d_k$ are mutually prime.). Then

$$f(b) = \max\left\{\rho_1 a (\sum_{j=1}^{k} d_j x_j) + \sum_{j>k}^{n} c_j x_j : a(\sum_{j=1}^{k} d_j x_j) + \sum_{j>k}^{n} w_j x_j \leq b, x \geq 0 \text{ integer}\right\}.$$

So we define

$$f_a(b) = \max\left\{\rho_1 a t + \sum_{j>k}^{n} c_j x_j : at + \sum_{j>k}^{n} w_j x_j \leq b, (t, x_{k+1}, \cdots, x_n) \geq 0 \text{ integer}\right\}.$$

Applying Lemmas 6 and 7 to $f_a(b)$

Lemma 8. (1) $f(b) \leq f_a(b)$.

(2) If $b \geq \rho_1 a/(\rho_1 - \rho_{k+1})$ then $f_a(b) = f_a(b-a) + \rho_1 a$.

(3) If $b \geq \rho_1 a/(\rho_1 - \rho_{k+1}) + as$ then for any optimal $\overline{x}$ for $f_a(b)$, $\overline{t} > s$.

To find small b from which $f(b) = f(b-a) + \rho_1 a$ we prepare

Lemma 9. For u, v such that $u < v$,

(1) if u is integer then there exists an integer in the interval $[u, v]$,

(2) if u is not integer then there exists integer in $[u, v]$ iff $[u] + 1 \leq v$ where $[u]$ is the greatest integer which is less than or equal to u.

Lemma 10. For mutually prime $e_1, e_2 \in N$ with $e_1x_1^0 + e_2x_2^0 = 1$, it holds for $t \in N$ that $(-tx_2^0/e_1) \in I$, or $[-tx_2^0/e_1]+1 \leq tx_1^0/e_2$ iff there exists $x_1, x_2 \in N_0$ such that $t = e_1x_1 + e_2x_2$.
Proof. *Only if* part; As

$$(tx_1^0/e_2) - (-tx_2^0/e_1) = t(e_1x_1^0 + e_2x_2^0)/(e_1e_2) = t/(e_1e_2) > 0,$$

by Lemma 9 there exists $p \in I$, $(-tx_2^0/e_1) \leq p \leq (tx_1^0/e_2)$. Letting $x_1 = tx_1^0 - e_2p$ and $x_2 = tx_2^0 + e_1p$, $e_1x_1 + e_2x_2 = t$ $(x_1, x_2 \in N_0)$. *If* part can be proved similarly. Q.E.D.

Remark. Condition $-tx_2^0/e_1 \in I$, or $[-tx_2^0/e_1] + 1 \leq tx_1^0/e_2$ is satisfied if $t \geq e_1e_2$.

Assuming $d_1 \leq \cdots \leq d_k$ (if not, reindex the x_j so as to satisfy this condition), we have three cases. (a) $d_1 = \cdots = d_k$, (b) $1 = d_1 < d_k$, (c) $1 < d_1 < d_k$. In case (a)

$$f(b) = f(b-a) + \rho_1 a d_1 = f(b-a) + c_1$$

for $b \geq c_1/(\rho_1 - \rho_{k+1})$ is easily derived. In case (b) $f_a(b) = f(b)$ for any $b \in N_0$ is also easily derived. And as $f_a(b) = f_a(b-a) + \rho_1 a$ for $b \geq \rho_1 a/(\rho_1 - \rho_{k+1})$ we have $f(b) = f(b-a) + \rho_1 a$ for $b \geq \rho_1 a/(\rho_1 - \rho_{k+1})$. To attack case (c) we prepare Condition (A) $1 < d_1 \leq \cdots \leq d_k$ and $d_1 < d_k$ and that there exist γ and δ such that d_γ, d_δ are mutually prime.

Remark. Condition (A) is always true when $k = 2$.

Lemma 11. Under Condition (A), let $d_\gamma x_\gamma^0 + d_\delta x_\delta^0 = 1$, $r = \min\{t : t \in N, -tx_\delta^0/d_\gamma \in I$ or $[-tx_\delta^0/d_\gamma] + 1 \leq tx_\gamma^0/d_\delta\}$ then for $t \in N$, $t \geq r$ there exist $x_1, \cdots, x_k \in N_0$ such that $at = w_1x_1 + \cdots + w_kx_k$.
Proof. Corresponding d_γ, d_δ to e_1, e_2 in Lemma 10, there exist x_γ, x_δ such that $t = d_\gamma x_\gamma + d_\delta x_\delta$. Noting that $ad_\gamma = w_\gamma$, $ad_\delta = w_\delta$ and setting $x_i = 0 (i \neq \gamma, \delta)$ we have $at = w_\gamma x_\gamma + w\delta x_\delta = w_1x_1 + \cdots + w_kx_k$. Q.E.D.

Remark. In fact, $r = d_\gamma d_\delta - (d_\gamma + d_\delta) + 1$ [189].

Theorem 3. Under Condition (A) let $b_A = \langle \rho_1 a/(\rho_1 - \rho_{k+1}) + ar \rangle$. If $b \geq b_A$ then $f(b) = f(b-a) + \rho_1 a$, where $\langle x \rangle$ is the least integer which is greater than or equal to x and r is defined in Lemma 11.
Proof. As $\langle x \rangle \geq x$, $b \geq \rho_1 a/(\rho_1 - \rho_{k+1}) + ar$. And by Lemma 7 for any optimal $\overline{x}$ for $f_a(b)$, $\overline{t} > r$ and by Lemma 11 there exist $x_1, \cdots, x_k \in N_0$ such that

$$at = w_1x_1 + \cdots + w_kx_k = a(d_1x_1 + \cdots + d_kx_k)$$

which lead to $f_a(b) = f(b)$. Noting the fact $\rho_1 a/(\rho_1 - \rho_{k+1}) \leq b$ and using Lemma 8 $f_a(b) = f_a(b-a) + \rho_1 a$. As $b - a \geq \rho_1 a/(\rho_1 - \rho_{k+1}) + a(r-1)$ similar argument as above yields $f_a(b-a) = f(b-a)$. Q.E.D.

For $b \geq b_A + a(d_j - 1) = \tilde{b}_j$,

$$f(b) = f(b-a) + \rho_1 a = \cdots = f(b - w_j) + \rho_1 w_j = f(b - w_j) + c_j$$

so that the optimal solution for $F(b)$ can be obtained by adding 1 to the jth component of the optimal solution for $F(b - w_j)(1 \leq j \leq k)$. This property is called *periodicity property*. Finally, we can prove in an elementary way that any Knapsack function has this property.

Lemma 12. For any $j(2 \leq j \leq k)$ there exists $u_j \in N$ such that for any $t \in N$, $t \geq u_j$ there exist $x_1, \cdots, x_j \in N_0$ with $(d_1, \cdots, d_j)t = d_1 x_1 + \cdots + d_j x_j$, where $(d_1, \cdots, d_j)$ is the greatest common divisor of $d_1, \cdots, d_j$.
Proof. (By induction). When $j = 2$, set $e_1 = d_1/(d_1, d_2)$, $e_2 = d_2/(d_1, d_2)$ and apply Lemma 10 with its Remark so that we can take $u_2 = d_1 d_2/(d_1, d_2)^2$. This proves Lemma for $j = 2$. Assuming Lemma is valid for j, let v_{j+1} be such that

$$((d_1, \cdots, d_{j+1})/(d_1, \cdots, d_j))v_{j+1} \geq u_j$$

and $((d_1, \cdots, d_{j+1})/(d_1, \cdots, d_j))v_{j+1}$: integer then for $t \geq u_{j+1}$,

$$u_{j+1} = v_{j+1} + (d_1, \cdots, d_j)d_{j+1}/((d_1, \cdots, d_j), d_{j+1})^2$$

(i.e. $t - v_{j+1} \geq (d_1, \cdots, d_j)d_{j+1}/((d_1, \cdots, d_j), d_{j+1})^2$), applying this Lemma for $(d_1, \cdots, d_j)$ and $d_{j+1}(k = 2)$ there exists $x_{1\cdots j}, x_{j+1} \in N_0$ such that

$$(d_1, \cdots, d_{j+1})(t - v_{j+1}) = (d_1, \cdots, d_j)x_{1\cdots j} + d_{j+1}x_{j+1}.$$

Noting $x_{1\cdots j} + ((d_1, \cdots, d_{j+1})/(d_1, \cdots, d_j))v_{j+1} \geq u_j$ and from the assumption of induction, there exist $x_1, \cdots, x_j \in N_0$ such that

$$(d_1, \cdots, d_j)(x_{1\cdots j} + ((d_1, \cdots, d_{j+1})/(d_1, \cdots, d_j))v_{j+1}) = d_1 x_1 + \cdots + d_j x_j.$$

So that $(d_1, \cdots, d_{j+1})t = d_1 x_1 + \cdots + d_{j+1} x_{j+1}$ for $t \geq u_{j+1}$. Q.E.D.

By $(d_1, \cdots, d_k) = 1$ and Lemma 12 we obtain

Theorem 4. If $b \geq \langle \rho_1 a/(\rho_1 - \rho_{k+1}) + a u_k \rangle$ then $f(b) = f(b-a) + \rho_1 a$, where u_k is given in Lemma 12.

Recalling the Remark beneath the Assumption 5 we have showed that the Knapsack function always has periodicity property.

Example. $n = 4$, $c = (5, 7, 6, 5)$, $w = (5, 7, 13, 11)$. As $5/5 = 7/7 > 6/13 > 5/11$, $k = 2$, $a = 1$, $d_1 = 5$, $d_2 = 7$, Condition (A) is satisfied with $\gamma = 1$, $\delta = 2$. And we have

$$x_1^0 = -4, x_2^0 = 3, r = 29, b_A = \langle 13/7 + 29 \rangle = 31, \tilde{b}_1 = 35, \tilde{b}_2 = 37.$$

According to the stopping criterion of references [46][66], it is assured that from $k = 31$, $b_k = 42$ there is no use calculating further. So, in this example our stopping criterion is superior to that of [46][66]. $\square$

2.2 Set Partitioning Problem

A careful consideration when one solves the Set-Partitioning Problem by dual all integer algorithm is presented. It saves both computing time and memory size.

Example A dispatching company DISPATCH EXPRESS(DE) has to dispatch some boxes of beverages to four sites S1,S2,S3,S4, where it places vending machine.See Figure 2.1. DE has to visit each site once a day, picking up some routes from candidate routes R1,R2,R3,R4,R5. Each route is automatically generated depending on traffic conditions so that today R1 lets DE visit S1,S3,S4 with cost 2. R2 lets DE dispatch its commodity to S2 only with cost 3. R3 lets DE visit S3,S4 with cost 6. R4 lets DE visit S1,S2 with cost 1. Finally R5 lets DE visit S2 and S3 with cost 5. Then, picking up the most costless routes is just solving the following Set Partitioning problem;

minimize

$$2x_1 + 3x_2 + 6x_3 + 1x_4 + 5x_5 \tag{2.1}$$

subject to

$$1x_1 + 1x_4 = 1 \tag{2.2}$$

$$1x_2 + 1x_4 + 1x_5 = 1 \tag{2.3}$$

$$1x_1 + 1x_3 + 1x_5 = 1 \tag{2.4}$$

$$1x_1 + 1x_3 = 1 \tag{2.5}$$

$$x_j = 0 \quad or \quad 1 (1 \le j \le 5). \tag{2.6}$$

Here

$$x_j = \begin{cases} 1 & \text{means that DE has to pick up route Rj} \\ 0 & \text{means that DE shouldn't pick up route Rj.} \end{cases}$$

$\square$

2.2.1 Introduction

A Set Partitioning Problem,

minimize

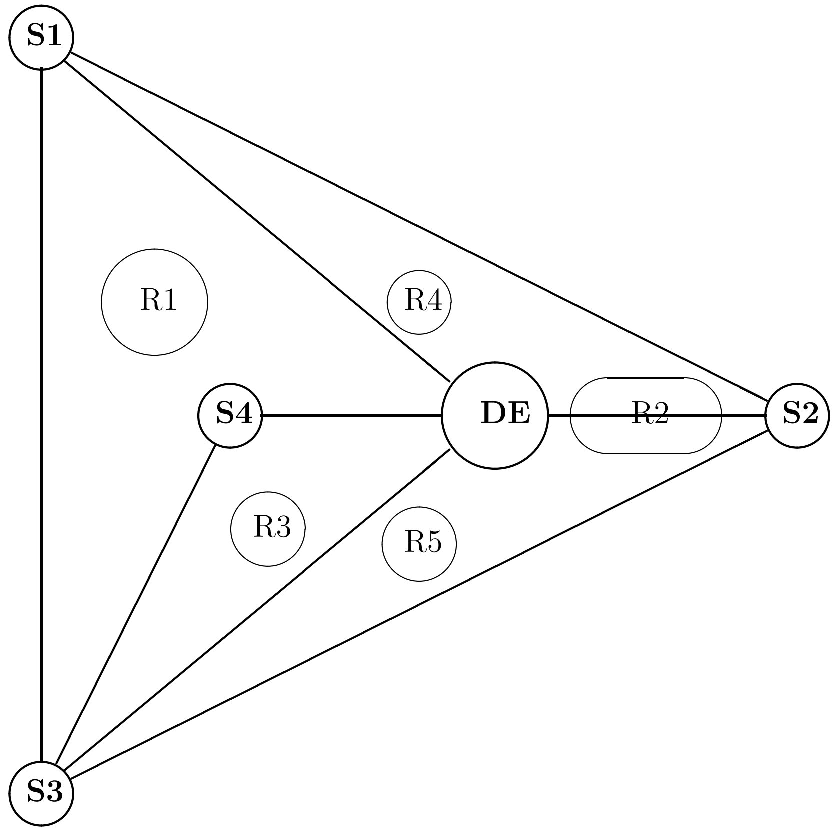

Figure 2.1: An Example of the Set Partitioning Problem

$$x_0 = \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = 1 (1 \leq i \leq m), x_j : binary (1 \leq j \leq n), \tag{2.7}$$

where c_j positive integer, $a_{ij} = 0$ or 1 can be solved by Dual All Integer Algorithm [46][66]. Salkin and Koncal [175][176][177] transformed this problem to the Set Covering Problem,

maximize

$$u_0 = \sum_{j=1}^{n} (c_j + Lh_j)(-x_j)$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 (1 \leq i \leq m), x_j : binary (1 \leq j \leq n), \tag{2.8}$$

where integer L is greater than $\sum_{j=1}^{n} c_j$, $h_j = \sum_{i=1}^{m} a_{ij}$ and then solved the original Set Partitioning Problem successfully.

Setting $x_{n+i} = \sum_{j=1}^{n} a_{ij}x_j - 1(1 \leq i \leq m)$, they applied Dual All Integer Algorithm to the dual feasible all integer tableau as follows [46][66];

$$\begin{array}{ccccccc} & & 1 & -x_1 & -x_2 & \cdots & -x_n \\ u_0 & & 0 & c_1 + Lh_1 & c_2 + Lh_2 & \cdots & c_n + Lh_n \\ x_{n+1} & & -1 & -a_{11} & -a_{12} & \cdots & -a_{1n} \\ x_{n+2} & & -1 & -a_{21} & -a_{22} & \cdots & -a_{2n} \\ \vdots & = & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n+m} & & -1 & -a_{m1} & -a_{m2} & \cdots & -a_{mn} \end{array} \tag{2.9}$$

Maximum tableau size could grow as large as $(m+n+2)(n+1)$, where we include a cut row.

Recently, Imai[74] discussed the importance to approximately solve the Set-Partitioning Problem greedily. But its performance is still $\Omega(\ln n)$. Therefore we think that the arguments below will be still worth stating.

2.2.2 Another Transformation

Let's consider another transformation which transforms (**??**) to

maximize

$$v_0 = -\sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij}x_j = 1(1 \leq i \leq m) \tag{2.10}$$

$$x_j \geq 0 \tag{2.11}$$

$$x_j : integer(1 \leq j \leq n) \tag{2.12}$$

where $v_0 = -x_0$.

Let M be any integer greater than the minimal value x_0 of (2.7), for example $M = \sum_{j=1}^{n} c_j + 1$,then we see that

$$v(2.12) > -M \tag{2.13}$$

as $v(2.7) = -v(2.12)$, where $v(P)$ denotes the optimal value of the 0-1 integer programming problem (P).

Consider one more problem such as

maximize

$$w_0 = -\sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} x_{n+i}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} = 1 (1 \leq i \leq m), x_u \geq 0 \text{integer} (1 \leq u \leq m+n). \tag{2.14}$$

We easily see that the following properties hold.

Property a (2.14) has a dual feasible integer solution $x_j = 0 (1 \leq j \leq n), x_{n+i} = -1 (1 \leq i \leq m)$ with the same dual feasible all integer tableau as (2.9), u_0, L replaced by w_0, M.

Property b (2.12) has a feasible integer solution if and only if (2.14) has a feasible integer solution whose objective function value w_0 is greater than $-M$.

Property c $v(2.14) \geq -\sum_{j=1}^{n} c_j - mM$

From these properties, we can obtain an optimal integer solution of (2.14) after finite iterations of Dual All Integer Algorithm. Moreover we have,

$$v(2.14) \begin{cases} > -M, & \text{iff every optimal solution of (2.14) is an optimal integer} \\ & \text{solution of (2.12) and v(2.14) = v(2.12),} \\ \leq -M, & \text{iff (2.14) is infeasible,} \end{cases}$$

so that we get the next *Procedure d.*

Procedure d; Every time any variable x_u $(n+1 \leq u \leq n+m)$ becomes nonbasic in the course of dual pivoting, we can drop x_u and its corresponding column from the tableau.

2.2.3 Illustrative Example

We quote Dual All Integer Algorithm from [46].

Step 0;(Preparation) Prepare simplex tableau,

$$x_{B_i} = y_{i0} + \sum_{j \in R} y_{ij}(-x_j), (0 \leq i \leq m) \tag{2.15}$$

where $x_B = x_0$ = objective function value, x_{B_i} $(1 \leq i \leq m)$ are basic variables x_j $(j \in R)$ are nonbasic variables. A vector $v \neq 0$ is called lexicographically positive if its first nonzero component is positive. We use notation $v >_L 0$ to denote v lexicographically positive. We use y_j to denote the j-th column of the simplex tableau (2.15). Simplex tableau (2.15) is called *dual feasible* if $y_j >_L 0$ for all $j \in R$, *all integer* if $y_{ij}(0 \leq i \leq m, 0 \leq j \leq n)$ are all integers. [u] denotes the largest integer less than or equal to u.

Step 1:(Initialization) Begin with a dual feasible all integer tableau (2.15).Go to step 2.

Step 2:(Test for optimality) If the solution is primal feasible, it is optimal to (2.15). STOP. If not, go to Step 3.

Step 3:(Cutting and pivoting) Choose a source row $(i \neq 0)$ in the tableau with $y_{i0} < 0$, say $i = r$. The topmost row with $y_{i0} < 0$, must be chosen at least periodically. Select the lexicographically smallest column with $y_{rj} < 0$, say $j = k$, as the pivot column. Compute $\overline{h}$ by

$$\overline{h} = \min_{j \in R_r} \frac{\overline{M}_j}{y_{rj}}$$

where $R_r = \{j \in R | y_{rj} < 0\}, \overline{M}_k = -1, \overline{M}_j = min\{u | y_j + uy_k >_L 0, u \quad \text{integer}\} \quad for \quad j \in R_r \backslash \{k\}$.

If $\overline{h} = 1$, execute one dual simplex iteration with the pivot element y_{rk}.

If $\overline{h} < 1$, adjoin the cut

$$s = [hy_{r0}] + \sum_{\{j \in R\}} [hy_{rj}](-x_j)$$

with $h = \overline{h}$, to the bottom of the tableau. Execute a dual simplex iteration with s as the departing variable and x_k as the entering variable. In any case, if x_k is a slack from a cut, delete the x_k row. Return to step 2.

To see the power of *Procedure d*, we take Example from [46](page 315). minimize

$$\begin{array}{ccccccccc}
3x_1 & +7x_2 & +5x_3 & +8x_4 & +10x_5 & +4x_6 & +6x_7 & +9x_8 & \\
x_1 & +x_2 & & & & & & & =1 \\
 & & x_3 & +x_4 & +x_5 & & & & =1 \\
 & & & & x_5 & +x_6 & +x_7 & & =1 \\
 & & & & & & x_7 & +x_8 & =1 \\
 & x_2 & & +x_4 & & +x_6 & & & =1
\end{array}$$

We start with the dual feasible all integer tableau (2.16) which is obtained through replacing u_0, L by $w_0, M = \sum_{j=1}^{8} c_j + 1 = 53$.

$$\begin{array}{cccccccccc}
 & 1 & -x_1 & -x_2 & -x_3 & -x_4 & -x_5 & -x_6 & -x_7 & -x_8 \\
w_0 & 265 & 56 & 113 & 58 & 114 & 116 & 110 & 112 & 62 \\
x_9 & -1 & -1^p & -1 & & & & & & \\
x_{10} & -1 & & & -1 & -1 & -1 & & & \\
x_{11} & -1 & & & & & -1 & -1 & -1 & \\
x_{12} & -1 & & & & & & & -1 & -1 \\
x_{13} & -1 & & -1 & & -1 & & -1 & &
\end{array} \tag{2.16}$$

r = 1, $R_r = \{1, 2\}$, k = 1, $\overline{M}_1 = -1$, $\overline{M}_2 = -2$, $y_{rk} = -1$ (having p on its upper right) gives $\overline{h} = 1$. Pivoting on y_{rk} makes x_1 basic, x_9 nonbasic so that we may drop x_9 column from the new tableau (2.17).

$$\begin{array}{ccccccccc}
 & 1 & -x_2 & -x_3 & -x_4 & -x_5 & x_6 & -x_7 & -x_8 \\
w_0 & 209 & 57 & 58 & 114 & 116 & 110 & 112 & 62 \\
x_1 & 1 & 1 & & & & & & \\
x_{10} & -1 & & -1^p & -1 & -1 & & & \\
x_{11} & -1 & & & & -1 & -1 & -1 & \\
x_{12} & -1 & & & & & & -1 & -1 \\
x_{13} & -1 & -1 & & -1 & & -1 & &
\end{array} \tag{2.17}$$

r=2, k=2, $\overline{M}_2 = -1$, $\overline{M}_3 = -1$, $\overline{M}_4 = -2$, y_{rk} = -1 (having p on its upper right) gives $\overline{h} = 1$. Pivoting on y_{rk} makes x_3 basic, x_{10} nonbasic so that we may drop x_{10} column from the next tableau (2.18)

$$
\begin{array}{c|ccccccc}
 & 1 & -x_2 & -x_4 & -x_5 & -x_6 & -x_7 & -x_8 \\
w_0 & 151 & 57 & 56 & 58 & 110 & 112 & 62 \\
x_1 & 1 & 1 & & & & & \\
x_3 & 1 & & 1 & 1 & & & \\
x_{11} & -1 & & & -1 & -1 & -1 & \\
x_{12} & -1 & & & & & -1 & -1 \\
x_{13} & -1 & -1 & -1 & & -1 & &
\end{array}
\tag{2.18}
$$

Doing in this way, i.e.,
x_5 basic, x_{11} nonbasic and so drop x_{11} column;
x_7 basic, x_{12} nonbasic and so drop x_{12} column;
x_6 basic, x_{13} nonbasic and so drop x_{13} column;
x_4 basic, x_5 nonbasic and so drop none,
we get final tableau (2.19) which is optimal,

$$
\begin{array}{c|cccc}
 & 1 & -x_2 & -x_5 & -x_8 \\
w_0 & -17 & 1 & 4 & 4 \\
x_1 & 1 & 1 & & \\
x_3 & 0 & -1 & 2 & -1 \\
x_4 & 1 & 1 & -1 & 1 \\
x_7 & 1 & & & 1 \\
x_6 & 0 & & 1 & -1
\end{array}
\tag{2.19}
$$

As v(2.19) = -17 > -53, we see that $x_1 = x_4 = x_7 = 1$, $x_j = 0$(otherwise),$x_0 = 17$ is an optimal solution. Final tableau size is half as large as the original. We also do away with needless calculations for the deleted columns.

Chapter 3

Greedoid and Greedy Algorithm

3.1 What is Greedoid?

A *Greedoid*(B.Korte and L.Lovász[118](1984)) is a set system $(E, \mathcal{F})$, where E is a finite set and $\mathcal{F}$ is a class of subsets of E satisfying

- (G1) $\emptyset \in \mathcal{F}$
- (G2) If $\emptyset \neq X \in \mathcal{F}$ then $X - \{a\} \in \mathcal{F}$ for some $a \in X$
- (G3) If $X, Y \in \mathcal{F}$ with $\mid X \mid > \mid Y \mid$, then there exists $a \in X - Y$ such that $Y \cup \{a\} \in \mathcal{F}$

Sets belonging to $\mathcal{F}$ are called *feasible sets*. A set system $(E, \mathcal{F})$ satisfying the above axioms (G1), (G3) and the following (M2)

- (M2) If $X \subset Y \in \mathcal{F}$ then $X \in \mathcal{F}$

are called *Matroid*([199]). Hence *Greedoid* is a direct relaxation of *Matroid* and so it has a lots of application in Combinatorial Optimization.

Let $(E, \leq)$ be a partially ordered set(*poset*,Birkhoff[11]). So, the set E with ordering $\leq$ satisfies

$$\text{for any} \quad x \in E, \quad x \leq x \tag{3.1}$$

$$\text{If} \quad x \leq y \quad \text{and} \quad y \leq x, \quad \text{then} \quad x = y \tag{3.2}$$

$$\text{If} \quad x \leq y \quad \text{and} \quad y \leq z, \quad \text{then} \quad x \leq z. \tag{3.3}$$

A *lower ideal* is a subset X of E such that

$$X \ni x \quad \text{and} \quad y \leq x \quad \text{implies} \quad y \in X.$$

Let $\mathcal{F}$ be a class of subsets of E which are lower ideals in the poset E. Then, we see that $(E, \mathcal{F})$ is a greedoid which are called *poset greedoid*(.Korte and Lovász[117][119][122], Korte, Lovász and Schrader[129]). A greedoid $(E, \mathcal{F})$ is said to be an *interval greedoid*, if $X \subseteq Y \subseteq Z, X \cup \{a\} \in \mathcal{F}$ and $Z \cup \{a\} \in \mathcal{F}$ imply $Y \cup \{a\} \in \mathcal{F}$ *(interval property)*. This condition is equivalent to ;

- (B) whenever $X, Y, Z \in \mathcal{F}$ such that $X, Y \subseteq Z$ then $X \cup Y \in \mathcal{F}$.

An interval greedoid is called a *shelling structure* if $E \in \mathcal{F}$.Thus the family of a shelling structure is closed under union.

ExampleTo clarify the difference between greedoid and matroid, we give here three examples.

Let $E_1 = \{a, b, c\}$ and $(E_1, \leq)$ be a poset given in the Figure 3.1.

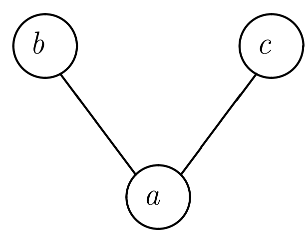

Figure 3.1: Poset $(E_1, \leq)$

Letting $\mathcal{F}_1$ be the set of lower ideals of $(E_1, \leq)$, we get

$$\mathcal{F}_1 = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}\}$$

which satisfies (G1),(G2),(G3) and so $\mathcal{F}_1$ is a greedoid(poset greedoid). We see that $\mathcal{F}_1$ is closed under set union and intersection operations with $\emptyset, E_1 \in \mathcal{F}_1$. Yet we have $\{b\} \subset \{a, b\} \in \mathcal{F}_1 \& \{b\} \notin \mathcal{F}_1$ and so $(E_1, \mathcal{F}_1)$ is not a matroid.

Let E_2 be a set of edges of a triangle in the Figure 3.2.

A subset X of E_2 is called *independent* if X does not contain a circuit in it. Let $\mathcal{F}_2$ be the set of independent sets. Then we get

$$\mathcal{F}_2 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}\}$$

which satisfies (G1),(M2),(G3) and so $(E_2, \mathcal{F}_2)$ is a matroid. Note that (G1),(M2),(G3) imply (G1),(G2),(G3) and therefore every matroid is a greedoid(*matroid greedoid*).

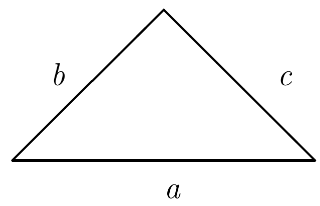

Figure 3.2: Triangle E_2

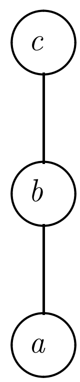

Figure 3.3: Chain Poset $(E_3, \leq)$

Let $(E_3, \leq)$ be a chain poset as in the Figure 3.3.

Let $\mathcal{F}_3$ be the set of lower ideals of $(E_3, \leq)$. Then we get a *chain poset greedoid* $(E_3, \mathcal{F}_3)$, where

$$\mathcal{F}_3 = \{\emptyset, \{a\}, \{a, b\}, \{a, b, c\}\}$$

and $(E_3, \mathcal{F}_3)$ is not a matroid. Again we see that $\mathcal{F}_3$ is closed under set union and intersection with $\emptyset, E_3 \in \mathcal{F}_3$.

3.2 Lexicographically Optimal Base of a Submodular System with Respect to a Weight Vector

Submodular system has been developed by Fujishige [37][38](1978-1987). He posed an algorithm to get lexicographically optimal base of a polymatroid with respect to a weight vector through geometric consideration[37] (1980). We have shown that the same results hold for a submodular system with $f(A) > 0$ ($\phi \neq A \in D$) and have presented a greedy procedure in an algebraic way (1987). In response to our work and to questions proposed by the author, Fujishige[38] (1987) has extended the same results for an arbitrary submodular system and has presented an algorithm to get it. His algorithm, which is not a direct extension of the algorithm for polymatroid, contains an oracle computation which has been pointed out by Morton, von Randow and Ringwald [160](1985). Here, we show a greedy procedure to get it though algebraic consideration, which is quite different from Fujishige's algorithm [36][38](1980,1987), because we get it algebraically.

Submodular system is essentially a poset greedoid with submodular function on it, which is implicitly stated in Fujishige and Tomizawa[40] (1983). Greedoids are created and have been investigated by Korte and Lovász[117][128] (1982-1986). Our result is a natural consequence through the study of greedoids and submodular systems. This chapter comes from K.Iwamura[92](1995).

3.2.1 Submodular System, Submodular Polyhedra and Their Basic Characteristics

We use the same symbol and terminology as that of Fijishige [37](1984). Let E be a finite set and denote by 2^E the set of all the subsets of E. Let a collection D of subsets of E be a *distributive lattice* with set union and intersection as the lattice operations, i.e., for any $X, Y \in D$ we have $X \cup Y, X \cap Y \in D$. A function f from D to the set R of reals is called a *submodular function* on D if for each pair of $X, Y \in D$

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

A pair (D, f) of a distributive lattice $D \subseteq 2^E$ and a submodular function $f : D \to R$ is called a *submodular system*. We assume that $\phi, E \in D$ and $f(\phi) = 0$. Note that the value $f(\phi)$ does not affect the other value $f(A)$ at $A \in D$ because $A \cup \phi = A$, $A \cap \phi = \phi$. Given a submodular system (D, f), define a polyhedron P_f by

$$P_f := \left\{ x \in R^E \mid x(X) \leq f(X) (\forall X \in D) \right\},$$

where R^E is the set of vectors $x = (x(e) : e \in E)$ with coordinates indexed by E and $x(e) \in R(e \in E)$ and

$$x(X) := \sum_{e \in X} x(e).$$

We call P_f the *submodular polyhedron* associated with the submodular system (D, f). Define

$$B_f := \{x \in P_f \mid x(E) = f(E)\},$$

which is called the *base polyhedron* associated with (D, f).

Lemma 3.1 Let $x \in P_f$ and $A, B \in D$. If $x(A) = f(A)$, $x(B) = f(B)$, then $x(A \cap B) = f(A \cap B)$ and $x(A \cup B) = f(A \cup B)$ hold.
Proof. Same as that of Fujishige [35](1978). Q.E.D.

Let χ_u be a characteristic function of u, i.e, $\chi_u(e) = 1$ for $e = u$ and $\chi_u(e) = 0$ for $e \in E \setminus \{u\}$. Define a *saturation function sat*(): $P_f \to 2^E$ by

$$sat(x) := \{u \in E \mid \forall d > 0, x + d\chi_u \notin P_f\} \quad (x \in P_f).$$

Then we have the following lemma, where $\wp(x) := \{A \in D | x(A) = f(A)\}$.

Lemma 3.2 Let $x \in P_f$. Then $sat(x)$ satisfies

$$sat(x) \in D, \quad x(sat(x)) = f(sat(x)).$$

Furthermore, $\wp(x)$ is a distributive lattice with a partial order relation defined by the set inclusion and $sat(x)$ is the maximum element of $\wp(x)$.
Proof. Same as that of Fujishige [36](1980). Q.E.D.

Note that $sat(x)$ is a function from P_f into D.

Lemma 3.3 Let $x \in P_f$. Then $x \in B_f$ iff $sat(x) = E$.
Proof. Use the definition of B_f and Lemma 3.2. Q.E.D.

For $x \in P_f$, $u \in sat(x)$, we can define *dependence function dep*() : $P_f \to D$ and also we can introduce capacity, exchange capacity and so on (Fujishige [37][38](1984,1987)), but we do not go into the details because we do not use them.

Let $n := |E|$. For any real sequences $a = (a_1, \cdots, a_n)$ and $b = (b_1, \cdots, b_n)$ of length n , a is called *lexicographically greater than or equal to* b if for some $j \in \{1, \cdots, n\}$

$$
\begin{aligned}
&a_i = b_i \quad (i = 1, 2, \cdots, j-1)\\
&a_j > b_j\\
&\text{or}\\
&a_i = b_i \quad (i = 1, 2, \cdots, n).
\end{aligned}
$$

A vector $w \in R^E$ such that $w(e) > 0 (e \in E)$ is called a *weight vector*. For a vector $x \in R^E$, denote by $T(x)$ the n-tuple (or sequence) of the numbers $x(e)(e \in E)$ arranged in order of increasing magnitude. Given a weight vector w, a base x of (D, f) is called a *lexicographically optimal base with respect to the weight vector* w if the n-tuple $T((x(e)/w(e))_{e \in E})$ is lexicographically maximum among all n-tuples $T((y(e)/w(e))_{e \in E})$ for all bases y of (D, f). The mathematical programming problem to get $x \in B_f$ such that

$$
\begin{aligned}
T((x(e)/w(e))_{e \in E}) \quad = \quad & \text{Lexicographically maximum } T((y(e)/w(e))_{e \in E})\\
& \text{subject to } y \in B_f
\end{aligned}
$$

is called *wlob* (weighted lexicographically optimal base) *problem* for submodular system.

3.2.2 Existence and Uniqueness of a Lexicographically Optimal Base with Respect to a Weight Vector

Let

$$c_1 := \min\left\{\frac{f(A)}{w(A)} \mid \phi \neq A \in D\right\}, \quad u_{c_1}(e) := c_1 w(e)(e \in E).$$

Then we see that $u_{c_1} \in P_f$ holds. By Lemma 3.2, we have

$$u_{c_1}(sat(u_{c_1})) = f(sat(u_{c_1})).$$

Let A_1 be a set such that

$$c_1 = \frac{f(A_1)}{w(A_1)}, \quad \phi \neq A_1 \in D.$$

Then $A_1 \subseteq sat(u_{c_1})$, because

$$\forall e \in A_1, \forall d > 0, \quad (u_{c_1} + d\chi_e)(A_1) = c_1 w(A_1) + d > f(A_1).$$

Thus we get $\phi \neq sat(u_{c_1}) \in D$. Therefore, we are in a position such that

$$u_{c_1}(e) = c_1 w(e)(e \in E), u_{c_1} \in P_f, \phi \neq sat(u_{c_1}) \in D, u_{c_1}(sat(u_{c_1})) = f(sat(u_{c_1})). \tag{3.4}$$

In case $sat(u_{c_1}) = E$, by Lemma 3.3, we see that

$$u_{c_1} \in B_f. \qquad \text{STOP}$$

In case $sat(u_{c_1}) \subset^{1} E$, let

$$\epsilon_1 := \min\left\{\frac{f(A) - u_{c_1}(A)}{w(A \setminus sat(u_{c_1}))} \mid A \setminus sat(u_{c_1}) \neq \phi, A \in D\right\}.$$

Then by Lemma 3.1, we get $\epsilon_1 > 0$. Let $c_2 := c_1 + \epsilon_1$, and let

$$u_{c_2}(e) := \begin{cases} c_1 w(e) = u_{c_1}(e), & e \in sat(u_{c_1}) \\ c_2 w(e) = u_{c_1}(e) + \epsilon_1 w(e), & e \in E \setminus sat(u_{c_1}). \end{cases}$$

By the definition of u_{c_2} and ϵ_1, and by the fact that $u_{c_1} \in P_f$, we get $u_{c_2} \in P_f$. Furthermore we get $\wp(u_{c_1}) \subseteq \wp(u_{c_2})$ and so $sat(u_{c_1}) \subseteq sat(u_{c_2})$. From the definition of ϵ_1, we have a set

$$A_1 \in D, A_1 \setminus sat(u_{c_1}) \neq \phi \text{ such that } \epsilon_1 = \frac{f(A_1) - u_{c_1}(A_1)}{w(A_1 \setminus sat(u_{c_1}))}.$$

Then

$$\begin{aligned} u_{c_2}(A_1) &= u_{c_2}(A_1 \cap sat(u_{c_1})) + u_{c_2}(A_1 \setminus sat(u_{c_1})) \\ &= c_1 w(A \cap sat(u_{c_1})) + (c_1 + \epsilon_1) w(A_1 \setminus sat(u_{c_1})) \quad \text{[by the definition of } u_{c_2}\text{]} \\ &= c_1 w(A_1) + \epsilon_1 w(A_1 \setminus sat(u_{c_1})) \\ &= u_{c_1}(A_1) + \epsilon_1 w(A_1 \setminus sat(u_{c_1})) \\ &= f(A_1) \end{aligned}$$

and so $A_1 \in \wp(u_{c_2})$.

By Lemma 3.1 and $sat(u_{c_1}) \in \wp(u_{c_2})$, we have

$$sat(u_{c_1}) \subset sat(u_{c_1}) \cup A \in \wp(u_{c_2}).$$

Thus $sat(u_{c_1}) \subset sat(u_{c_2})$. From Lemma 3.2 and $u_{c_2} \in P_f$, we have

$$u_{c_2}(sat(u_{c_2})) = f(sat(u_{c_2})). \tag{3.5}$$

Therefore, we are in a position such that

$$u_{c_2}(e) = \begin{cases} c_1 w(e)(e \in sat(u_{c_1})), \\ c_2 w(e)(e \in E \setminus sat(u_{c_1})), \\ \qquad u_{c_i} \in P_f (i = 1, 2), \phi \neq sat(u_{c_1}) \subset sat(u_{c_2}) \in D \end{cases} \tag{3.6}$$

$u_{c_i}(sat(u_{c_i})) = f(sat(u_{c_i}))(1 \leq i \leq 2)$ and $c_1 < c_2$.

[1] $X \subset Y$ means that X is a proper subset of Y.

Continuing this process, we get u_{c_P} such that $sat(u_{c_P}) = E$, i.e., $u_{c_P} \in B_f$. Set

$$c(e) := \left\{ \begin{array}{l} c_1(e \in sat(u_{c_1})) \\ c_2(e \in sat(u_{c_2}) \setminus sat(u_{c_1})) \\ \vdots \\ c_i(e \in sat(u_{c_i}) \setminus sat(u_{c_{i-1}})) \\ \vdots \\ c_P(e \in sat(u_{c_P}) \setminus sat(u_{c_{P-1}}) = E \setminus sat(u_{c_{P-1}})) \end{array} \right\} \tag{3.7}$$

Then we have

$$u_{c_P}(e) = \left\{ \begin{array}{l} c_1 w(e)(e \in sat(u_{c_1})) \\ c_2 w(e)(e \in sat(u_{c_2}) \setminus sat(u_{c_1})) \\ \vdots \\ c_i w(e)(e \in sat(u_{c_i}) \setminus sat(u_{c_{i-1}})) \\ \vdots \\ c_P w(e)(e \in sat(u_{c_P}) \setminus sat(u_{c_{P-1}})) \end{array} \right.$$

$u_{c_P} \notin B_f$, $\phi \neq sat(u_{c_1}) \subset \cdots \subset sat(u_{c_P}) = E$ which are all in D,

$$u_{c_i}(sat(u_{c_i})) = f(sat(u_{c_i})) \quad (1 \leq i \leq p)$$

and

$$c_1 < \cdots < c_P. \tag{3.8}$$

Note. For a *positive* submodular system (D, f), i.e., submodular system with $f(A) > 0(\phi \neq A \in D)$, we see that $c_1 > 0$.

Theorem 3.1. (Existence) Let $c(e)(e \in E)$ be those defined by (3.7). Then the vector x defined by

$$x = (c(e)w(e))_{e \in E} \tag{3.9}$$

is a lexicographically optimal base with respect to the weight vector w.

Proof. Let $z \in B_f$. We show that an equality

$$T((z(e)/w(e))_{e \in E}) \leq_l T((x(e)/w(e)_{e \in E}) \tag{3.10}$$

holds. First note that

$$z(A) \leq f(A) \qquad (\phi \neq A \in D) \tag{3.11}$$

holds. Let $q := (q_1, \cdots, q_n)$, $n = |E|$, be any permutation corresponding to x such that

$$\frac{x(q_1)}{w(q_1)} = \cdots = \frac{x(q_{j_1})}{w(q_{j_1})} = c_1 < \frac{x(q_{j_1+1})}{w(q_{j_1+1})} = \cdots = \frac{x(q_{j_2})}{w(q_{j_2})} = c_2 < \cdots <$$

$$< \frac{x(q_{j_{P-1}+1})}{w(q_{j_{P-1}+1})} = \cdots = \frac{x(q_{j_P})}{w(q_{j_P})} = c_P, \quad j_P = n, c_{j_0} = 0.$$

Let $S_i = \{q_{j_{i-1}+1}, q_{j_{i-1}+2}, \cdots, q_{j_i}\}(1 \leq i \leq p)$. Then we have $S_1 = sat(u_{c_1})$, $S_i = sat(u_{c_i}) \setminus sat(u_{c_{i-1}})(2 \leq i \leq p)$.

If $\dfrac{z(q_1)}{w(q_1)} < c_1$, then (3.10) holds.

If $\dfrac{z(q_1)}{w(q_1)} \geq c_1$, $\dfrac{z(q_2)}{w(q_2)} < c_1$, then (3.10) holds.

$\vdots$

If $\dfrac{z(q_1)}{w(q_1)} \geq c_1, \cdots, \dfrac{z(q_{j_1})}{w(q_{j_1})} \geq c_1$, then we see that

$$\frac{z(e)}{w(e)} = \frac{x(e)}{w(e)} = c_1 \quad (e \in S_1) \tag{3.12}$$

holds by $z(S_1) \geq c_1 w(S_1) = u_{c_1}(S_1) = f(S_1)$ and by (3.11).

If $\dfrac{z(e)}{w(e)} = c_1(e \in S_1); \dfrac{z(q_{j_1+1})}{w(q_{j_1+1})} < c_2$, then (3.10) holds.

If $\dfrac{z(e)}{w(e)} = c_1(e \in S_1), \dfrac{z(q_{j_1+1})}{w(q_{j_1+1})} \geq c_2, \dfrac{z(q_{j_1+2})}{w(q_{j_1+2})} < c_2$, then (3.10) holds.

$\vdots$

If $\dfrac{z(e)}{w(e)} = c_1(e \in S_1), \dfrac{z(q_{j_1+1})}{w(q_{j_1+1})} \geq c_2, \cdots, \dfrac{z(q_{j_2})}{w(q_{j_2})} \geq c_2$, then we see that

$$\frac{z(e)}{w(e)} = c_2 = \frac{x(e)}{w(e)} \quad (e \in S_2)$$

holds because $z(e) = c_1 w(e)$ $(e \in S_1)$ and

$$z(S_2 + S_1) \leq f(S_2 + S_1) = u_{c_2}(S_2 + S_1) = z(S_1) + c_2 w(S_2) \leq z(S_2 + S_1).$$

Continuing in this way, we see that (3.10) holds for any $z \in B_f$. Q.E.D.

Theorem 3.2. (Uniqueness, Fujishige[36] (1980)) Let $c(e)(e \in E)$ be those defined by (3.7). Then the vector x defined by (3.9) is the unique lexicographically optimal base of (D, f) with respect to a weight vector w.
Proof. Same as that of Fuhishige (1980). Use (3.8), especially

$$sat(u_{c_i}) \in D, \qquad u_{c_i}(sat(u_{c_i})) = f(sat(u_{c_i})).$$

Q.E.D.

Based on these algebraic arguments, we present an algorithm to get the lexicographically optimal base of submodular system (D, f) with respect to a weight vector w.

Algorithm to get the lexicographically optimal base

Step 1. Set $i := 1$ and compute $c_i := \min\left\{\frac{f(A)}{w(A)} | \phi \neq A \in D\right\}$ and set $u_{c_i}(e) := c_i w(e)(e \in E)$.

Step 2. If $sat(u_{c_i}) = E$, then STOP.

Step 3. Compute

$$\epsilon_i := \min\left\{\frac{f(A) - u_{c_i}(A)}{w(A \setminus sat(u_{c_i}))} \mid A \in D, A \setminus sat(u_{c_i}) \neq \phi\right\}$$

and set $c_{i+1} := c_i + \epsilon_i$ and set

$$u_{c_{i+1}}(e) := \begin{cases} u_{c_i}(e), & e \in sat(u_{c_i}), \\ u_{c_i}(e) + \epsilon_i w(e), & e \in E \setminus sat(u_{c_i}). \end{cases}$$

Set $i := i + 1$ and go to Step 2.

Theorem 3.3. (Fujishige [36](1980)) Let $\hat{x} \in B_f$ and let w be a weight vector. Define

$$\hat{c}(e) := \hat{x}(e)/w(e) \quad (e \in E)$$

and let the distinct numbers of $\hat{c}(e)$ $(e \in E)$ be given by

$$\hat{c}_1 < \hat{c}_2 < \cdots < \hat{c}_{\hat{p}}.$$

Furthermore, define $\hat{S}_i \subseteq E(1 \leq i \leq \hat{p})$ by

$$\hat{S}_i := \{e \in E \mid \hat{c}(e) \leq \hat{c}_i\} \qquad (1 \leq i \leq \hat{p}).$$

Then the following three conditions are equivalent:
(i) $\hat{x}$ is the lexicographically optimal base of P_f with respect to w;

(ii) $\hat{S}_i \in D$ and $\hat{x}(\hat{S}_i) = f(\hat{S}_i)(1 \leq i \leq \hat{p})$;
(iii) For any $e \in \hat{S}_i$, $\phi \neq dep(\hat{x}, e) \subseteq \hat{S}_i(1 \leq i \leq \hat{p})$.

Remark. If one of the three conditions holds, then we have $\hat{p} = p$.

Given a submodular system (D, f) and a weight vector w and $p > 1$, define a mathematical programming problem

$$P : \min f_w(x) = \frac{1}{p} \sum_{e \in E} \frac{x(e)^p}{w(e)^{p-1}} \text{ subject to } x \in B_f \text{ and } x \geq 0.$$

Fujishige[36] (1980) showed that for a polymatroid (D, f) with $p = 2$, its unique solution is the lexicographically optimal base w.r.t. w. Morton, Von Randow and Ringwald[160] (1985) extended it for $p > 1$, where (D, f) is a polymatroid. We can easily see that for a positive submodular system (D, f) with $p > 1$, the same result holds. As for an arbitrary submodular system, P might be infeasible. For example, for a submodular system (D, f) with $f(A) < 0 (A \in D)$. So, consider another problem

$$\hat{P} : \min f_w(x) = \frac{1}{p} \sum_{e \in E} \frac{x(e)^p}{w(e)^{p-1}} \text{ subject to } x \in B_f.$$

We have an example for which $\hat{P}$ has no optimal solution as follows: Let $E = \{1, 2, 3\}$, $D = \{\phi, \{3\}, \{1, 2, 3\}\}$, $f(\phi) = 0$, $f(\{3\}) = -2$, $f(\{1, 2, 3\}) = -3$. Then (D, f) is a submodular system with base polyhedron

$$B_f = \{(x_1, x_2, x_3) \mid x_1 + x_2 + x_3 = -3, x_3 \leq -2\}.$$

Let $w = (1, 1, 1)$. The lexicographically optimal base x^* becomes $x^* = (-\frac{1}{2}, -\frac{1}{2}, -2)$. Let $p = 3$ and let $x_1 = x_2 = -\frac{(t+3)}{2}$, $x_3 = t(\leq -2)$. Then $(x_1, x_2, x_3) \in B_f$ with

$$3f_w(x) = t^3 - \frac{1}{4}(t + 3)^3 \to -\infty \text{ as } t \to -\infty.$$

Problem $\hat{P}$ for this case has no minimum solution. For an even natural number p, if there exists a minimum solution $\hat{x}$ for $\hat{P}$, then we see that $\hat{x}$ is the lexicographically optimal base w.r.t. w.

Theorem 3.4. (Fujishige[36] (1980), Morton, von Random and Ringwald [160](1985)) Let x^* be the lexicographically optimal base of a positive submodular system (D, f) with respect to a weight vector w and let $p > 1$. Then x^* is the unique optimal solution of the problem P.

3.2.3 Illustrative Example

Example Let $E = \{1, 2, 3\}, \mathcal{D} = \{\emptyset, \{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}, f(\emptyset) = 0, f(\{1\}) = 3, f(\{1, 2\}) = 5, f(\{1, 3\}) = 4, f(E) = 6$. Then $\mathcal{D}$ is a distributive lattice and

f is a submodular function on $\mathcal{D}$. Therefore,$x = (x_1, x_2, x_3) \in P_f$ in and only if

$$x_1 \leq 3, x_1 + x_2 \leq 5, x_1 + x_3 \leq 4, x_1 + x_2 + x_3 \leq 6.$$

$x \in B_f$ if and only if $x \in P_f, x_1 + x_2 + x_3 = 6$. Applying our Primal Algorithm for a positive weight vector $w = (2, 1, 2)$, we get $(0, 0, 0) \longrightarrow (2, 1, 2) \longrightarrow (2, 2, 2)$:lexicographically optimal base of B_f.

We will further show that the first problem of Morton, von Random and Ringwald[160] (1985) can be solved within our framework. Their problem is as follows:

$$\min \sum_{j=1}^{n} \lambda_j x_j^p \text{ subject to } Ax \geq c, x \geq 0, \tag{3.13}$$

where $\lambda_j > 0 (1 \leq j \leq n)$, $p > 1$, $c_n \geq c_{n-1} \geq c_1 \geq 0$, and

$$A = (a_{ij})_{n \times n} \text{ with } a_{ij} = \begin{cases} 1, & i \geq j \\ 0, & i < j. \end{cases}$$

Let e_i be the i-th column vector of A,

$$\begin{aligned} E &:= \{e_i \mid 1 \leq i \leq n\}, \\ F_j &:= \{e_i \mid 1 \leq i \leq j\} \quad (1 \leq j \leq n), \\ F_0 &:= \phi, \\ D_j &:= E \setminus F_j = \{e_{j+1}, \cdots, e_n\} \quad (0 \leq j \leq n) \end{aligned}$$

Let $D = \{E = D_0, D_1, \cdots, D_{n-1}, D_n = \phi\}$. Let $\rho(D_j) := c_n - c_j (0 \leq j \leq n)$, where $c_0 = 0$. Then (E, D, ρ) is a submodular system with ϕ, $E \in D$, $\rho(\phi) = 0$. For $x, y \in R_+^n$, define $x \leq y$ if $x(e) \leq y(e) (e \in E)$, where R_+ is the set of nonnegative reals. $(R_+^n, \leq)$ is a poset with this partial order. Define

$$P := \{x \in R_+^n \mid Ax \geq c\},$$

O(3.13) := the set of optimal solutions to (3.13), minimal P := the set of minimal elements of P. Then we easily see that

$$O(3.13) \subseteq B_\rho \subseteq \text{ minimal } P \subseteq P.$$

Hence the problem (3.13) is equivalent to

$$\min \left\{ \frac{1}{p} \sum_{i=1}^{n} x(e_i)^p w(e_i)^{1-p} \mid x \in B_\rho \right\},$$

where $w(e_i) = \lambda_i^{-\frac{1}{(p-1)}}$. Let $d_j = \sum_{i=1}^{j} w(e_i)(1 \leq j \leq n)$ and $d_0 = 0$. Then $w(D_j) = d_n - d_j (0 \leq j \leq n)$. Now let us apply our algorithm to this problem:

$$\begin{aligned} c_1' &:= \min\left\{\frac{\rho(D_j)}{w(D_j)} \mid 0 \leq j \leq n-1\right\} \\ &= \min\left\{\frac{c_n - c_0}{d_n - d_0}, \frac{c_n - c_1}{d_n - d_1}, \frac{c_n - c_2}{d_n - d_2}, \cdots, \frac{c_n - c_{n-1}}{d_n - d_{n-1}}\right\}. \end{aligned}$$

Let $s'(0) = n$ and

$$c_1' = \frac{c_n - c_{s'}(1)}{d_n - d_{s'}(1)}$$

and $u_{c_1'}(e_i) = c_1' w(e_i)(1 \leq i \leq n)$. Then $u_{c_1'}(D_j) = c_1'(d_n - d_j)$,

$$sat(u_{c_1'}) = \cup\left\{A \mid A \in D, u_{c_1'}(A) = \rho(A)\right\} = D_{s'(1)}$$

for which $s'(1)$ is the least index j such that

$$c_1' = \frac{c_n - c_j}{d_n - d_j}, \quad 0 \leq s'(1) < s'(0).$$

If $s'(1) = 0$, then $sat(u_{c_1'}) = E$. STOP.
If $s'(1) \neq 0$, then $sat(u_{c_1'}) \neq E$ and so compute

$$\begin{aligned} \epsilon_1' &:= \min\left\{\frac{\rho(A) - u_{c_1'}(A)}{w(A \setminus sat(u_{c_1'}))} \mid A \in D, A \setminus sat(u_{c_1'}) \neq \phi\right\} \\ &= \min\left\{\frac{c_n - c_j - c_1'(d_n - d_j)}{d_{s'(1)} - d_j} \mid 0 \leq j \leq n-1, j < s'(1)\right\}, \end{aligned}$$

where

$$\frac{c_n - c_j - c_1'(d_n - d_j)}{d_{s'(1)} - d_j} = \frac{c_{s'(1)} - c_j}{d_{s'(1)} - d_j} - c_1'.$$

Let

$$\epsilon_1' := \frac{c_{s'(1)} - c_{s'(2)}}{d_{s'(1)} - d_{s'(2)}} - c_1'.$$

Then $(d_{s'(2)}, c_{s'(2)})$ is a point (d_j, c_j), $0 \leq j < s'(1)$ with the smallest slope coefficient $\frac{c_{s'(1)} - c_j}{d_{s'(1)} - d_j}$. Hence we see that

$$s'(0) = n = s(m), \quad s'(1) = s(m-1), \quad \cdots, \quad s'(m-1) = s(1), \quad s'(m) = s(0),$$

which is the same result as that of Morton, von Randow and Ringwald, although the decision proceeds inversely. The reader would have noticed that the (E, D) here, is a poset greedoid which comes from a chain as in the Figure 3.4.

The reason for the inverse decision process will be investigated in the next section.

3.3 Primal Dual Algorithms for the Lexicographically Optimal Base of a Submodular Polyhedron and Its Relation to a Poset Greedoid

In the preceeding subsection, we proved the existence and uniqueness of a lexicographically optimal base of a submodular system with respect to a weight vector (Iwamura, K.[92](1995)). There we presented a greedy procedure to get it, which is quite different from Fujishige's algorithm [36][38](1980, 1987) and explains the algorithm of the first problem of Morton, G. and von Randow, R. and Ringwald, K.[160](1985). There, we noticed that the greedy procedure proceeds inversely to the algorithm of Morton, G. and von Randow, R. and Ringwald, K. (1985) and asked ourselves why?

Here, we present another algorithm to get a lexicographically optimal base of a submodular system with respect to a weight vector. When the distributive lattice of a submodular system is simple, it is, in fact, a poset greedoid. It is well known that there exist two algorithms to find an optimal base of a matroid and/or a shelling structure (Korte, B. and Lovász, L. [120](1984)), Iwamura, K.([89](1985)) for a linear objective function. Hence our result can be considered as another example for which there exist two or more greedy algorithms. This chapter is based on K. Iwamura[93](1995).

3.3.1 Definition

Let E be a finite set and denote by 2^E the set of all the subsets of E. Let a collection $\mathcal{F}$ of subsets of E be a *distributive lattice* (Birkhoff[11]) with set union and intersection as the lattice operations, i.e., for any $X, Y \in \mathcal{F}$ we have $X \cup Y, X \cap Y \in \mathcal{F}$. A function f from $\mathcal{F}$ to the set R of reals is called a *submodular function* (Fijishige,S.[37]) on $\mathcal{F}$ if for each pair of $X, Y \in \mathcal{F}$

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

A triple $(E, \mathcal{F}, f)$ of a finite set E and a distributive lattice $\mathcal{F} \subseteq 2^E$ and a submodular function $f : \mathcal{F} \to R$ is called *submodular system.* We assume that $\phi, E \in \mathcal{F}$ and $f(\phi) = 0$. It is well known that for a distributive lattice $\mathcal{F} \subseteq 2^E$ with $\phi, E \in \mathcal{F}$ there uniquely exist a partition $\Pi = \{A_1, \cdots, A_k\}$ of E and a partial order $\leq$ on Π satisfying $\mathcal{F} \ni X$ iff there exists an ideal I of poset $(\Pi, \leq)$ such that $X = \cup\{A_i | A_i \in I\}$ (Birkhoff G.[11], Fujishige, S. and Tomizawa, N.[40]). Note that the correspondence $X \leftrightarrow I$ is a bijection. For a submodular system $(E, \mathcal{F}, f)$, by identifying each $X \in \mathcal{F}$ with $I \subseteq \Pi$, we obtain a distributive lattice $\mathcal{F}' \subseteq 2^{E'}$ with $E' = \Pi$ and a submodular function

$f' : \mathcal{F}' \to R$. That is to say,

$$\begin{aligned}\mathcal{F}' &:= \{I \subseteq \Pi \mid \cup\{A_i \in I\} \in \mathcal{F}\} \\ &= \{I \subseteq \Pi \mid I \text{ is an ideal of } (\Pi, \leq)\},\end{aligned}$$

$f'(I) := f(\cup\{A_i | A_i \in I\})$ for $I \in \mathcal{F}'$. We see that $(E', \mathcal{F}')$ is a poset greedoid (Korte, B. and Lovász, L.[117][118]) and hence $(E', \mathcal{F}', f')$ is still a (*simple*) submodular system. $(E', \mathcal{F}', f')$ is called a *simplification* of $(E, \mathcal{F}, f)$.

For a submodular system $(E, \mathcal{F}, f)$, define a *submodular polyhedron* $P(f)$ and a *submodular base polyhedron* $B(f)$ by

$$P(f) = \left\{x \in R^E \mid x(X) \leq f(X)(X \in \mathcal{F})\right\},$$

$$B(f) = \left\{x \in R^E \mid x(X) \leq f(X)(X \in \mathcal{F}) \text{ and } x(E) = f(E)\right\},$$

where coordinates indexed by E and $x(e) \in R(e \in E)$ and

$$x(X) := \sum_{e \in X} x(e).$$

Define

$$\overline{\mathcal{F}} := \{E - X \mid X \in \mathcal{F}\},$$

$$\overline{f}(E - X) := f(E) - f(X)(E - X \in \overline{\mathcal{F}}).$$

Then

$$\overline{\mathcal{F}} = \{X \subseteq E \mid X \text{ is an upper ideal of } (E, \leq)\}$$

with $\phi, E \in \overline{\mathcal{F}}$, $\overline{f}(\phi) = 0$ and $\overline{f}$ is *supermodular* on $\overline{\mathcal{F}}$, i.e., for each pair of $X, Y \in \overline{\mathcal{F}}$,

$$\overline{f}(X \cup Y) + \overline{f}(X \cap Y) \geq \overline{f}(X) + \overline{f}(Y).$$

$(E, \overline{\mathcal{F}}, \overline{f})$ is called *dual supermodular system* of $(E, \mathcal{F}, f)$. Define a *supermodular polyhedron* $P(\overline{f})$ and *supermodular base polyhedron* $B(\overline{f})$ by

$$P(\overline{f}) := \left\{x \in R^E \mid x(X) \geq \overline{f}(X)(X \in \overline{\mathcal{F}})\right\},$$

$$B(\overline{f}) := \left\{x \in R^E \mid x(X) \geq \overline{f}(X)(X \in \overline{\mathcal{F}}) \text{ and } x(E) = \overline{f}(E)\right\}$$

respectively (Fijishige, S.[37]). Then we have

$$\overline{f}(\phi) = f(\phi) = 0, \quad \overline{f}(E) = f(E), \quad B(\overline{f}) = B(f).$$

Any vector $x \in B(f) = B(\overline{f})$ is called a base of $B(f) = B(\overline{f})$. Let χ_u be a *characteristic function* of u, i.e., $\chi_u(e) = 1$ for $e = u$ and $\chi_u(e) = 0$ for $e \in E \setminus \{u\}$. Define a *dual saturation function* $\overline{sat}() : P(\overline{f}) \to 2^E$ by

$$\overline{sat}(x) = \left\{u \in E \mid \forall d > 0, x - d\chi_u \notin P(\overline{f})\right\}.$$

Then we have the following lemmas, where

$$\overline{\mathcal{A}}(x) := \left\{ A \in \overline{\mathcal{F}} \mid x(A) = \overline{f}(A) \right\}$$

(Iwamura, K.[92], Fujishige, S.[36]).

Lemma 3.4 Let $x \in P(\overline{f})$ and $A, B \in \overline{\mathcal{F}}$. If $x(A) = \overline{f}(A)$, $x(B) = \overline{f}(B)$, then $x(A \cap B) = \overline{f}(A \cap B)$ and $x(A \cup B) = \overline{f}(A \cup B)$ hold.

Lemma 3.5 Let $x \in P(\overline{f})$. Then $\overline{sat}(x)$ satisfies

$$\overline{sat}(x) \in \overline{\mathcal{F}}, \quad x(\overline{sat}(x)) = \overline{f}(\overline{sat}(x)).$$

Furthermore $\overline{\mathcal{A}}(x)$ is a distributive lattice with a partial order relation defined by the set inclusion and $\overline{sat}(x)$ is the maximum element of $\overline{\mathcal{A}}(x)$.

Lemma 3.6 Let $x \in P(\overline{f})$. Then $x \in B(\overline{f})$ iff $\overline{sat}(x) = E$.

Let $n := |E|$. For any real sequences $a = (a_1, \cdots, a_n)$ and $b = (b_1, \cdots, b_n)$ of length n , a is called *lexicographically greater than or equal to* b if for some $j \in \{1, \cdots, n\}$

$$\begin{aligned}
&a_i = b_i \quad (i = 1, 2, \cdots, j-1) \\
&a_j > b_j \\
&\text{or} \\
&a_i = b_i \quad (i = 1, 2, \cdots, n).
\end{aligned}$$

A vector $w \in R^E$ such that $w(e) > 0 (e \in E)$ is called a *weight vector*. For a vector $x \in R^E$, denote by $T(x)$ the n-tuple (or sequence) of the numbers $x(e)(e \in E)$ arranged in order of increasing magnitude. Given a weight vector w, a base x of $P(f)$ is called a *lexicographically maximum base with respect to the weight vector* w if the n-tuple $T((x(e)/w(e))_{e\in E})$ is lexicographically maximum among all n-tuples $T((y(e)/w(e))_{e\in E})$ for all bases y of $P(f)$. The mathematical programming problem to get $x \in B(f)$ such that

$$\begin{aligned}
T((x(e)/w(e))_{e\in E}) \quad &= \quad \text{Lexicographically maximum } T((y(e)/w(e))_{e\in E}) \\
&\quad\quad \text{subject to } y \in B(f)
\end{aligned}$$

is called *wl max b* (weighted lexicographically maximum base) *problem* for a submodular base polyhedron $B(f)$.

For a vector $x \in R^E$, denote by $\overline{T}(x)$ the n-tuple (or sequence) of the numbers $x(e)(e \in E)$ arranged in order of decreasing magnitude. Given a weight vector w, a base x of $B(\overline{f})$ is called a *lexicographically minimum base with* *respect to the weight vector* w if the n-tuple $\overline{T}((x(e)/w(e))_{e\in E})$ is lexicographically minimum among all n-tuples $\overline{T}((y(e)/w(e))_{e\in E})$ for all bases y of $P(\overline{f})$. The mathematical programming problem to get $x \in B(\overline{f})$ such that

$$\begin{aligned}\overline{T}((x(e)/w(e))_{e\in E}) &= \text{Lexicographically minimum } \overline{T}((y(e)/w(e))_{e\in E}) \\ &\quad \text{subject to } y \in B(\overline{f})\end{aligned}$$

is called *wl min b* (weighted lexicographically minimum base) *problem* for supermodular base polyhedron $B(\overline{f})$.

3.3.2 Primal Dual Algorithms for the Lexicographically Optimal Base of a Submodular Polyhedron and Its Relation to a Poset Greedoid

In the preceeding section (Iwamura[92]), we have developed an algorithm to get the (unique) lexicographically maximum base with respect to the weight vector w.

Algorithm to get the lexicographically maximum base(Primal)

Step 1. Set $i := 1$ and compute

$$c_i := \min\left\{\frac{f(A)}{w(A)} \mid \phi \neq A \in \mathcal{F}\right\}$$

and set $U_{c_i}(e) := c_i w(e)(e \in E)$.

Step 2. If $sat(U_{c_i}) = E$, then STOP.

Step 3. Compute

$$\epsilon_i := \min\left\{\frac{f(A) - U_{c_i}(A)}{w(A \setminus sat(U_{c_i}))} \mid A \in \mathcal{F}, A \setminus sat(U_{c_i}) \neq \phi\right\}$$

and set $c_{i+1} := c_i + \epsilon_i$ and set

$$U_{c_{i+1}}(e) := \begin{cases} U_{c_i}(e), & e \in sat(U_{c_i}), \\ U_{c_i}(e) + \epsilon_i w(e), & e \in E - sat(U_{c_i}). \end{cases}$$

Set $i := i + 1$ and go to Step 2.

With Lemmas 3.4-3.6, similar arguments as that of Iwamura, K.[92] show that the following algorithm produces the lexicographically minimum base with respect to the weight vector w.

Algorithm to get the lexicographically minimum base(Dual)

Step 1. Set $i := 1$ and compute

$$\overline{c_i} := \max\left\{\frac{\overline{f}(A)}{w(A)} \mid \phi \neq A \in \overline{\mathcal{F}}\right\}$$

and set $U_{\overline{c_i}}(e) := \overline{c_i} w(e)(e \in E)$.

Step 2. If $\overline{sat}(U_{\overline{c_i}}) = E$, then STOP.

Step 3. Compute

$$\overline{\epsilon_i} := \min\left\{\frac{U_{\overline{c_i}}(A) - \overline{f}(A)}{w(A \setminus \overline{sat}(U_{\overline{c_i}}))} \mid A \setminus \overline{sat}(U_{\overline{c_i}}) \neq \phi, A \in \overline{\mathcal{F}}\right\}$$

and set $\overline{c_{i+1}} := \overline{c_i} - \overline{\epsilon_i}$ and set

$$U_{\overline{c_{i+1}}}(e) := \begin{cases} U_{\overline{c_i}}(e), & e \in \overline{sat}(U_{\overline{c_i}}), \\ U_{\overline{c_i}}(e) - \overline{\epsilon_i} w(e), & e \in E \setminus \overline{sat}(U_{\overline{c_i}}). \end{cases}$$

Set $i := i + 1$ and go to Step 2.

Suppose that the above algorithm stops after d iterations, then we have

$$U_{\overline{c_d}}(e) = \begin{cases} \overline{c_1} w(e)(e \in \overline{sat}(U_{\overline{c_1}})) \\ \overline{c_2} w(e)(e \in \overline{sat}(U_{\overline{c_2}}) \setminus \overline{sat}(U_{\overline{c_1}})) \\ \vdots \\ \overline{c_i} w(e)(e \in \overline{sat}(U_{\overline{c_i}}) \setminus \overline{sat}(U_{\overline{c_{i-1}}})) \\ \vdots \\ \overline{c_d} w(e)(e \in \overline{sat}(U_{\overline{c_d}}) \setminus \overline{sat}(U_{\overline{c_{d-1}}}) = E \setminus \overline{sat}(U_{\overline{c_{d-1}}})), \end{cases}$$

$U_{\overline{c_d}} \in B(\overline{f}) = B(f)$, $\phi \subset$ [2] $\overline{sat}(U_{\overline{c_1}}) \subset \cdots \subset \overline{sat}(U_{\overline{c_d}}) = E$ which are all in $\overline{\mathcal{F}}$, $U_{\overline{c_d}}(\overline{sat}(U_{\overline{c_i}})) = \overline{f}(\overline{sat}(U_{\overline{c_i}}))$ $(1 \leq i \leq d)$ and $\overline{c_1} > \overline{c_2} > \cdots > \overline{c_d}$.

Theorem 3.5 (Primal-dual theorem). The above $U_{\overline{c_d}}$ is the lexicographically maximum base with respect to the weight vector w.
Proof. We use Theorem 3.3 (See, Iwamura, K.[92]). Define $\hat{c}(e) := U_{\overline{c_d}}(e)/w(e)(e \in E)$. Then we see that $\hat{p} = d$ with $\hat{c}_1 = \overline{c_d}$, $\hat{c}_2 = \overline{c_{d-1}}$, $\cdots$, $\hat{c}_d = \overline{c_1}$. Using $U_{\overline{c_d}} \in B(\overline{f}) = B(f)$, we get

$$U_{\overline{c_d}}(E) = \overline{f}(E) = f(E),$$

[2] $X \subset Y$ means that X is a proper subset of Y.

and

$$\begin{aligned} U_{\overline{c_d}}(E - \overline{sat}(U_{\overline{c_i}})) &= f(E) - \overline{f}(\overline{sat}(U_{\overline{c_i}})) \\ &= f(E) - \{f(E) - f(E - \overline{sat}(U_{\overline{c_i}}))\} \\ &= f(E - \overline{sat}(U_{\overline{c_i}})), \end{aligned}$$

where

$$\phi \subset E - \overline{sat}(U_{\overline{c_{d-1}}}) \subset E - \overline{sat}(U_{\overline{c_{d-2}}}) \subset \cdots \subset E - \overline{sat}(U_{\overline{c_1}}) \subset E,$$

all in $\mathcal{F}$. Furthermore

$$E - \overline{sat}(U_{\overline{c_i}}) = \{e \in E \mid \hat{c}(e) \leq \overline{c_{i+1}}\} \quad (0 \leq i \leq d-1).$$

Hence by Theorem 3.3 we get that $U_{\overline{c_d}}$ is the lexicographically maximum base with respect to weight vector w. Q.E.D.

A careful reader would have noticed that the proof for Theorem 3.1 of Iwamura, K.[92] remains valid for $z \in P(f)$. Hence the following mathematical programming problems,

Lexicographically maximum $T((y(e)/w(e))_{e \in E})$,
subject to $y \in P(f)$

Lexicographically minimum $\overline{T}((y(e)/w(e))_{e \in E})$,
subject to $y \in P(\overline{f})$

have the same solution as that of *wl min b* - and *wl max b* - problem. Hence, we call these problems *wlo* (weighted lexicographically optimal)-problems for a submodular system.

Let $(E, \mathcal{F}, f)$ be a submodular system and let $(E', \mathcal{F}', f')$ be its simplification. Let $w(e) > 0 (e \in E)$ be a weight vector and let

$$w'(A_i) := \sum_{e \in A_i} w(e) > 0 \quad (A_i = e'_i \in E'(1 \leq i \leq k)).$$

Theorem 3.6 Let $x'(e')(e' \in E')$ be the lexicographically maximum base of $(E', \mathcal{F}', f')$ with respect to the weight vector w' just above. Let $x(e) = (w(e)/w'(e'_i))x'(e'_i)$ for any $e \in e'_i$, $e'_i \in E'$. Then $x(e)(e \in E)$ is the lexicographically maximum base of $(E, \mathcal{F}, f)$ with respect to the weight vector w.

Proof. Submodular polyhedron corresponding to $(E', \mathcal{F}', f')$ and $(E, \mathcal{F}, f)$ become

$$P(f') = \left\{x' \in R^{E'} \mid x'(A) \leq f'(A)(A \in \mathcal{F}')\right\}$$

and

$$P(f) = \Big\{x \in R^E \mid x(X) \leq f(X)(X \in \mathcal{F})\Big\}$$

respectively.

Let $c'(e') = (x'(e')/w'(e'))(e' \in E')$ and let $c(e) = x(e)/w(e))(e \in E)$. Let the distinct numbers of $c'(e')(e' \in E')$ be given by $c'_1 < \cdots < c'_{p'}$ and define

$$S'_i = \{e' \in E' \mid c'(e') \leq c'_i\},$$
$$S_i = \{e \in E \mid c(e) \leq c'_i\}.$$

Then the distinct numbers of $c(e)(e \in E)$ are just the same as that of $c'(e')(e' \in E')$. By Theorem 3.3 , we see that

$$S'_i \in \mathcal{F}' \text{ and } x'(S'_i) = f'(S'_i)(1 \leq i \leq p').$$

$x \in B(f)$ because for any $X \in \mathcal{F}$, $X = \cup\{A_i | A_i \in I\}$, $I \in \mathcal{F}'$ and

$$x(X) = \sum_{A_i \in I} \sum_{e \in A_i} x(e) = \sum_{A_i \in I} x'(A_i) = x'(I) \leq f'(I) = f(X)$$

with $x(E) = f(E)$. By the definition of $\mathcal{F}'$ and $S'_i \in \mathcal{F}'$, we get

$$\mathcal{F} \ni \cup\{A_j | A_j \in S'_i\} = \{e \in E | c(e) \leq c'_i\} = S_i$$

with $x(S_i) = f(S_i)$ for $1 \leq i \leq p'$.

Again by Theorem 3.3 , we get the conclusion. Q.E.D.

Let $(E', \mathcal{F}')$ be an arbitrary poset greedoid on $E' = \{e'_1, \cdots, e'_m\}$. Let f' be a submodular function on $\mathcal{F}'$ with $f'(\phi) = 0$. Then $(E', \mathcal{F}', f')$ is a simple submodular system. For each $e'_i \in E'$, assign a subset E'_i of E such that

$$E'_i \cap E'_j = \phi(1 \leq i < j \leq m) \text{ and } \cup_{i=1}^m E'_i = E.$$

Let $|E| = \sum_{i=1}^m |E'_i| = n$, and let

$$\mathcal{F} := \{\cup_{i \in I} E'_i \mid \{e'_i | i \in I\} \in \mathcal{F}'\}.$$

Then clearly $(E, \mathcal{F})$ is a distributive lattice with set union and intersection as the lattice operations, and $\phi, E \in \mathcal{F}$. Define $f : \mathcal{F} \to R$ by $f(\cup_{i \in I} E'_i) = f'(\{e'_i | i \in I\})$ for any $\{e'_i | i \in I\} \in \mathcal{F}'$. Then f is a submodular function with $f(\phi) = 0$ and so $(E, \mathcal{F}, f)$ is a general submodular system, which we call the expansion of $(E', \mathcal{F}', f')$. In fact, the simplification of $(E, \mathcal{F}, f)$ is $(E', \mathcal{F}', f')$. Given a positive weight vector $w(e)(e \in E)$, define

$$w'(e'_i) := w(E'_i) = \sum_{e \in E'_i} w(e).$$

Then $w'(e'_i) > 0$ for any $e'_i \in E'$.

Corollary 3.7 (Expansion theorem). Let $x'(e')(e' \in E')$ be the lexicographically maximum base of $(E', \mathcal{F}', f')$ with respect to the weight vector w'. Let

$$x(e) = \frac{w(e)}{w'(e'_i)} x'(e'_i)$$

for any $e \in E'_i$ $e'_i \in E'$. Then $x(e)(e \in E)$ is the lexicographically maximal base of $(E, \mathcal{F}, f)$ with respect to the weight vector w.

Proof. Same as Theorem 3.6. Q.E.D.

3.3.3 Concluding Remark

Historically, submodular systems and greedoids have developed independently. But as we have seen, the algorithms to get the lexicographically optimal base of a submodular system can be derived by the algorithms for a simple submodular system, where its underlying distributive lattice is a poset greedoid. It might be interesting to investigate whether we can extend our results to a more general greedoid, say a shelling structure([119]).

3.4 Discrete Decision Process Model Involves Greedy Algorithm Over Greedoid

In the past thirty years, a huge amount of research activities to develop good algorithms for discrete optimization problems were carried out. Through these activities, it has widely acknowledged that a special algebraic structure named matroid allows us a very nice algorithm, a greedy algorithm (cf. D.J.A.Welsh[199]). Korte and Lovász[118] [120] showed that such a greedy algorithm also exists for a much wider algebraic structure - a greedoid - with a little bit restricted objective function. Here we see that this greedoid with its greedy algorithm lies within a discrete decision process(K.Iwamura[91]) .

3.4.1 Greedy Algorithm over Matroid

Let Q be the collection of independent sets of a matroid on Σ. Let R be the set of reals. Let $c : \Sigma \to R$ be a cost function and extend $c : 2^\Sigma \to R$ in the obvious way

$$c(F) = \sum_{f \in F} c(f) \qquad (F \subseteq \Sigma).$$

We well know[199]that the optimization problem

$$min\{C(F); F \in B\}, \qquad B \text{ is the base set of } (\Sigma, Q) \tag{3.14}$$

can be effectively solved by

The Greedy algorithm

Step0: Set $k = 1$.

Step1: Choose a member x_k such that $\{x_1, \ldots, x_k\} \in Q$ and $c(x_k) \leq c(x)$ for all $x \in \Sigma \setminus \{x_1, \ldots, x_{k-1}\}$ such that $\{x_1, \ldots, x_{k-1}, x\} \in Q$.

Step2: If no such x exists,stop. Otherwise set $k = k + 1$ and return to Step1.

Let $q_0 = \emptyset$.Define $\lambda : Q \times \Sigma \to Q \cup \{q_d\}$ by

$$\lambda(F, a) = \begin{cases} F \cup \{a\}, \text{ if } a \in \Sigma \setminus F \text{ and } F \cup \{a\} \in Q, \\ q_d, \text{ otherwise,} \end{cases}$$

where $q_d \notin Q$.

Set $Q_F = B$. Then $M = (Q, \Sigma, q_0, \lambda, Q_F)$ is a finite automaton, where
Q: state space,
Σ: alphabet,
$q_0 \in Q$: initial state,
λ: state transition function,
$Q_F \subseteq Q$: the set of final states,
q_d: dead state.

Let $h : R \times Q \times \Sigma \to R$ be defined by

$$h(\xi, F, a) = \begin{cases} \xi + c(a), \text{ if } a \in \Sigma \setminus F \text{ and } F \cup \{a\} \in Q, \\ +\infty, \text{ otherwise.} \end{cases}$$

Let $\xi_0 = 0$ (zero). Then $\Pi = (M, h, \xi_0)$ is a recursive loopless monotone sequential decision process (γ-lmsdp)([67]),in fact, it is a multi-stage sequential decision process.

Let $\Sigma^* = \{a_1 \ldots a_n : a_i \in \Sigma (0 \leq i \leq n), n \geq 0\}$ and let $\bar{h}(x) = h(\xi_0, q_0, x)$ and let $\bar{\lambda}(x) = \lambda(q_0, x)$ for any $x \in \Sigma^*$.Note that $\bar{h}(xa) = h(\xi_0, q_0, x) + c(a)$ holds. Define the set of optimal policy $O(\Pi)$ by

$$O(\Pi) = \{x \in F(M) :\sim (\exists y \in F(M))(\bar{h}(y) < \bar{h}(x))\},$$

where $F(M)$ is the set of accepting strings of M.

Let $G(q)$ be defined by

$$G(q) = \min\{\bar{h}(x) : \bar{\lambda}(x) = q\} \text{ for } q \in Q.$$

Let $Q_i = \{F \in Q : |F| = i\}$ for $i = 0, 1, \ldots, r$, where r is the rank of (Σ, Q). Consider the sequence

$$Q_0, Q_1, Q_2, \ldots, Q_r$$

and call each element of Q in the above list from left to right

$$q_0 = \emptyset, q_1, q_2, \ldots, q_{n-1}$$

with the dead state $q_d = q_n, |Q| = n$.

Ibaraki[67](1973) proposed an algorithm to get $O(\Pi)$ for γ-lmsdp Π.

<u>An Algorithm to get $O(\Pi)$</u>

$$\text{Let} X(q_j) = \{x \in \Sigma^* : \bar{\lambda}(x) = q_j\} \text{and let} X(q_i)a = \{xa : x \in X(q_i)\}.$$

Step1: Set $G(q_0) = \xi_0$,
$X(q_0) = \{\varepsilon\}$ (ε is the empty string), $j = 1$.

Step2: If $j = n$ then go to Step3. Otherwise set
$G(q_j) = min\{h(G(g_i), q_i, a) : \lambda(q_i, a) = q_j, i < j, a \in \Sigma\}$.
$X(q_j) = \cup\{X(q_i)a : \lambda(q_i, a) = q_j, i < j, a \in \Sigma\}$
and set $j = j + 1$ and return to Step2.

Step3: Set
$G^* = min\{G(q_j) : q_j \in Q_F\}$,
$O(\Pi) = \{x : x \in X(q_j) \text{ and } q_j \in Q_F \text{ and } \bar{h}(x) = G^*\}$
and stop. G^* is the value of the optimal
policies $O(\Pi)$. $\square$

Theorem 3.8 Any greedy solution of the optimization problem (1) belongs to $O(\Pi)$. In fact, the greedy algorithm is a special form of the above algorithm to get $O(\Pi)$. $\square$

3.4.2 Greedy Algorithm over Greedoid

It is easy to see that the same theorem holds for both greedy algorithm over greedoid and greedy algorithm over symmetric matroids.See,B.Korte and L.Lovász[118](1984) and A. Bouchet[16](1987).

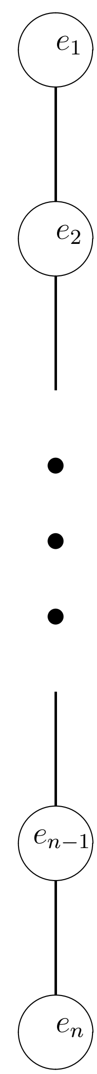

Figure 3.4: A Chain which Leads to a Poset Greedoid

Chapter 4

Uncertain Programming

4.1 The Need for Uncertain Programming

In Capital Budgeting Problem, let c_j be the profit obtained when one invest a_j amount of money to project j. But, is it really true that project j needs exactly a_j amount of money? If the cost structure changes , so does the amount of money needed to invest to project j, of course. In some situation we have to think that a_j is a random variable from normal distribution,or a random variable from log-normal distribution, and so on. Or, it might be a fuzzy and/or possibility number. The same phenomenom also occurs to c_j. Even under such uncertain situation, one has to make a plan and decide somehow. Therefore there exists need for uncertain programming,decision-making/planning under uncertainty.

Management decisions are sometimes made in uncertain environments. Stochastic programming offers a means of considering the objectives and constraints with stochastic parameters[25][32][109]. One method dealing with stochastic parameters in stochastic programming is the so-called *stochastic programming with recourse* which minimizes the original costs and expected recourse costs. The other one, *chance constrained programming* (CCP), was developed by Charnes and Cooper[18]. The basic technique of CCP is to convert the stochastic constraints to their respective deterministic equivalents according to the predetermined confidence level. It is also well-known that the concept of CCP has been extended to chance constrained goal programming and chance constrained multiobjective programming.

As a stochastic search method based on the mechanics of natural selection and natural genetics, genetic algorithm (evolution program or evolution strategies) has been applied to a wide variety of problems(Goldberg[54] and Michalewicz[156]), such as optimal control problems, transportation problems, traveling salesman problems, drawing graphs, scheduling and machine learning. And the three main avenues of research in simulated evolution, genetic algorithm, evolution program and evolution strategies, are summarized by

Fogel[34].So, throughout this book, *Uncertain Programming* means *Programming under Uncertainty.*

4.2 A Genetic Algorithm for Chance Constrained Programming

In this section, we will focus our attention on the technique of chance constrained programming(CCP), including chance constrained goal programming (CCGP) and chance constrained multiobjective programming(CCMOP). A genetic algorithm will be presented for solving CCP, CCGP and CCMOP. In order to deal with stochastic constraints, Monte Carlo simulation is employed to check the feasibility of a solution in the proposed genetic algorithm. Finally, we use some numerical examples to illustrate the effectiveness of genetic algorithm for chance constrained programming.

A typical formulation of chance-constrained programming (CCP) may be written as follows:

$$\begin{cases} \max \ E_\xi f(\mathbf{x}, \xi) \\ \text{subject to:} \\ \qquad \Pr\{\xi \mid g_i(\mathbf{x}, \xi) \le 0, i = 1, 2, \cdots, p\} \ge \alpha \end{cases} \tag{4.1}$$

where $\mathbf{x}$ is a decision vector, ξ is a stochastic vector, $f(\mathbf{x}, \xi)$ is the return function, E_ξ denotes the expected value operator on ξ, $g_i(\mathbf{x}, \xi)$ are stochastic constraint functions, $i = 1, 2, \cdots, p$, $\Pr\{\cdot\}$ denotes the probability of the event in $\{\cdot\}$ and α is a predetermined confidence level. So a point $\mathbf{x}$ is feasible if and only if the probability measure of the set $\{\xi | g_i(\mathbf{x}, \xi) \le 0, i = 1, 2, \cdots, p\}$ is at least α. In other words, the constraints will be violated at most $(1 - \alpha)$ of time.

The probabilistic constraints in CCP (4.1) are called joint chance constraints. Sometimes, the probabilistic constraints are separately considered as

$$\Pr\{\xi \mid g_i(\mathbf{x}, \xi) \le 0\} \ge \alpha_i, \quad i = 1, 2, \cdots, p \tag{4.2}$$

which are called separate chance constraints.

As an extension of chance constrained programming, chance constrained multiobjective programming may be written as follows:

$$\begin{cases} \max \ [E_\xi f_1(\mathbf{x}, \xi), E_\xi f_2(\mathbf{x}, \xi), \cdots, E_\xi f_m(\mathbf{x}, \xi)] \\ \text{subject to:} \\ \qquad \Pr\{\xi \mid g_i(\mathbf{x}, \xi) \le 0, i = 1, 2, \cdots, p\} \ge \alpha. \end{cases} \tag{4.3}$$

We can also formulate our uncertain decision system as a chance constrained goal programming:

$$
\begin{cases}
\min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ + v_{ij} d_i^-) \\
\text{subject to:} \\
\qquad E_\xi f_i(\mathbf{x}, \xi) + d_i^- - d_i^+ = b_i, \qquad i = 1, 2, \cdots, m \\
\qquad \Pr\{\xi \mid g_j(\mathbf{x}, \xi) \le 0\} \ge \alpha_j, \qquad j = 1, 2, \cdots, p \\
\qquad d_i^-, d_i^+ \ge 0, \qquad i = 1, 2, \cdots, m
\end{cases}
\tag{4.4}
$$

where

P_j = the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, for all j,

u_{ij} = weighting factor corresponding to positive deviation for goal i with priority j assigned,

v_{ij} = weighting factor corresponding to negative deviation for goal i with priority j assigned,

d_i^+ = positive deviation from the target of goal i,

d_i^- = negative deviation from the target of goal i,

$\mathbf{x}$ = n-dimensional decision vector,

f_i = a function in goal constraints,

g_j = a function in real constraints,

b_i = the target value according to goal i,

ξ = stochastic vector of parameters,

l = number of priorities,

m = number of goal constraints,

p = number of real constraints.

4.2.1 Monte Carlo Simulation

When the constraints are easy to be handled, we can convert the probability constraints to their deterministic equivalents. But if the constraints fail to be regular or hard to be calculated, it is more convenient to check the feasibility of a solution by a Monte Carlo method. Generally, let

$$
G(\mathbf{x}) = \Pr\{\xi \mid g_j(\mathbf{x}, \xi) \le 0, j = 1, 2, \cdots, p\} \tag{4.5}
$$

where $\xi = (\xi_1, \xi_2, \cdots, \xi_t)$ is a t-dimensional stochastic vector, and each ξ_i has a given distribution. For any given $\mathbf{x}$, we use the following Monte Carlo technique to estimate $G(\mathbf{x})$. We generate N independent random vectors $\xi^{(i)} = (\xi_1^{(i)}, \xi_2^{(i)}, \cdots, \xi_t^{(i)})$, $i = 1, 2, \cdots, N$, from their probability distributions, where the generating methods have been well-discussed by numerous literatures and summarized by Rubinstein[173]. Let N' be the number of occasions on which

$$g_j(\mathbf{x}, \xi^{(i)}) \le 0, \quad j = 1, 2, \cdots, p, \quad i = 1, 2, \cdots, N,$$

i.e., the number of random vectors satisfying the constraints. Then, by the basic definition of probability, $G(\mathbf{x})$ can be estimated by

$$G(\mathbf{x}) = \frac{N'}{N}. \tag{4.6}$$

This means that a chance constraint $\Pr\{\xi \mid g_j(\mathbf{x}, \xi) \le 0, j = 1, 2, \cdots, p\} \ge \alpha$ holds if and only if $N'/N \ge \alpha$. Certainly, this estimation is approximate and may change from a simulation to another. But it is available to real practice problem since the determination of the confidence level α itself is not precise.

4.2.2 A Genetic Algorithm

In this subsection we design a genetic algorithm to CCP. We will discuss the initialization process, evaluation function, selection, crossover and mutation operations in turn.

Initialization Process

At first, we will handle the constraints, i.e., eliminating the equalities presented in the set of real constraints if they exist. It is clear that z equality constraints imply that we can eliminate z variables in CCP by replacing them by the representations of the remaining variables. Because one usually can solve the system of equations such that z variables are represented by others. From now, we suppose that we have finished it.

We use a vector $V = (x_1, x_2, \cdots, x_n)$ as a chromosome to represent a solution to CCP and define an integer *pop_size* as the number of chromosomes. *pop_size* chromosomes will be randomly initialized by the following steps:

Step 1. Determine an interior point, denoted by V_0, in the deterministic constraint set.

Step 2. Select randomly a direction d in R^n and define a chromosome V as $V_0 + M \cdot d$ if it is feasible, otherwise, we set M as a random number in $[0, M]$ until $V_0 + M \cdot d$ is feasible, where M is a large positive number which ensures that all the genetic operators are probabilistically complete for the feasible solutions.

Step 3. Repeat Step 2 *pop_size* times and produce *pop_size* initial feasible solutions.

Evaluation Function

At first, we rearrange these chromosomes in order. The single-objective chance constrained programming is easy to be handled, the one with better objective value has higher rank.

Let us consider the case of chance constrained goal programming. The objective function of goal programming is $\sum_{k=1}^{l}\sum_{i=1}^{m} P_k(u_{ki}d_i^+ + v_{ki}d_i^-)$, where P_k is the preemptive priority factor which expresses the relative importance of various goals, $P_k \gg P_{k+1}$, for all k, but it is not suitable for the objective function to be as an evaluation function because that we have only the information $P_k \gg P_{k+1}$ on the priority factors. In fact, we have the following order relationship for the chromosomes: for any two chromosomes, if the higher-priority objectives are equal, then, in the current priority level, the one with minimal objective value is better. This relationship is an order on the feasible set and can rearrange these chromosomes in order. If two different chromosomes have the same objective value, then we rearrange them randomly.

For multiobjective chance constrained programming, we can also arrange these chromosomes if some information on weights is given.

We select the rank-based evaluation function because it ignores the information about the relative evaluations of different chromosomes. Now let a parameter $a \in (0,1)$ in the genetic system be given, then we can define the rank-based evaluation function as follows:

$$eval(V) = a(1-a)^{rank-1} \tag{4.7}$$

where *rank* is the ordinal number of V in the rearranged series. We mention that $rank = 1$ means the best individual, $rank = pop\_size$ the worst individual, and, of course,

$$\sum_{j=1}^{pop\_size} eval(V_j) \approx 1. \tag{4.8}$$

Selection Operation

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select a single chromosome for a new population in the following way:

Step 1. Calculate a cumulative probability a_i for each chromosome V_i, $(i = 1, 2, \cdots, pop\_size)$.

Step 2. Generate a random real number r in $[0,1]$.

Step 3. If $r \le a_1$, then select the first chromosome V_1; otherwise select the i-th chromosome V_i $(2 \le i \le pop\_size)$ such that $a_{i-1} < r \le a_i$.

Step 4. Repeat Steps 2 and 3 *pop_size* times and obtain *pop_size* copies of chromosomes.

In this process, the best chromosomes get more copies, the average stay even, and the worst die off.

Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. This probability gives us the expected number $P_c \cdot pop\_size$ of chromosomes which undergo the crossover operation.

Firstly we generate a random real number r in $[0, 1]$; secondly, we select the given chromosome for crossover if $r < P_c$. Repeat this operation *pop_size* times and produce $P_c \cdot pop\_size$ parents, averagely. For each pair of parents (vectors V_1 and V_2), the crossover operator on V_1 and V_2 will produce two children as

$$V_1' = \lambda_1 \cdot V_1 + \lambda_2 \cdot V_2, \qquad V_2' = \lambda_2 \cdot V_1 + \lambda_1 \cdot V_2$$

where $\lambda_1,\ \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 = 1$.

If the constraint set is convex, this arithmetical crossover operation ensures that both children are feasible if both parents are. If not, we only replace the parents by the feasible offsprings. We can produce offsprings several times by selecting different sets of λ_1 and λ_2 such that the parents can be replaced by their feasible offsprings as much as possible.

Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. This probability gives us the expected number $P_m \cdot pop\_size$ of chromosomes which undergo the mutation operation.

Generating a random real number r in $[0, 1]$, we select the given chromosome for mutation if $r < P_m$. Let a parent for mutation, denoted by a vector $V = (x_1, x_2, \cdots, x_n)$, be selected. Assume that $\{j_1, j_2, \cdots, j_z\}$ is a subset of $\{1, 2, \cdots, n\}$. We can use the procedure initialization to assign a new chromosome $V' = (x_1', x_2', \cdots, x_n')$ except for the fact that all $x_j' = x_j$ are predetermined for all $j \notin \{j_1, j_2, \cdots, j_z\}$. Repeat this operation *pop_size* times.

Following selection, crossover and mutation, the new population is ready for its next evaluation. The algorithm will terminate after a given number of cyclic repetitions of the above steps. The proposed genetic algorithm is shown as follows:

Procedure Genetic Algorithm

Input parameters;
Initialize the solutions (chromosomes);

REPEAT

Update the chromosomes by genetic operators;

Compute fitness of each chromosome by rank-based evalution function;

Select the chromosomes by spinning the roulette wheel;

UNTIL(*termination_condition*)

4.2.3 Numerical Examples

The computer code for the genetic algorithm to CCP has been written in C language. To illustrate the effectiveness of genetic algorithm, a set of numerical examples has been done, and the results are successful. Here we give some numerical examples which are all performed on NEC EWS4800/210II workstation with the following parameters: the population size is 30, the probability of crossover P_c as 0.2, the probability of mutation P_m as 0.4, the parameter a in the rank-based evaluation function as 0.1.

Production Process

This example is a modification of one in Kall and Wallace[109]. Let us consider a weekly production process of a refinery relying on countries for the supply of crude oil (raw_1 and raw_2, respectively), supplying on big company with gasoline ($prod_1$) for its distribution system of gas stations and another with fuel oil ($prod_2$) for its heating and/or power plants.

It is known that the productivities $\pi(raw_1, prod_1)$ and $\pi(raw_2, prod_2)$, i.e., the outputs of gas from raw_1 and output of fuel from raw_2 may change randomly, whereas the other productivities are deterministic. They are assumed to be

$$\pi(raw_1, prod_1) = 2 + \eta_1, \quad \pi(raw_1, prod_2) = 3,$$
$$\pi(raw_2, prod_1) = 6, \qquad \pi(raw_2, prod_2) = 3.4 - \eta_2$$

where η_1 has a uniform distribution $\mathcal{U}[-0.8, 0.8]$, η_2 has an exponential distribution $\mathcal{EXP}(0.4)$.

The weekly demands of the clients, h_1 for gas and h_2 for fuel are also varying randomly and represented by

$$h_1 = 180 + \xi_1, \qquad h_2 = 162 + \xi_2$$

where ξ_1 and ξ_2 have normal distributions $\mathcal{N}(0, 12)$ and $\mathcal{N}(0, 9)$, respectively.

The output of products per unit of the raw materials are $c_1 = 2$ for raw_1 and $c_2 = 3$ for raw_2, respectively. The total cost is thus $2x_1 + 3x_2$.

The production capacity, i.e., the maximal total amount of raw materials is assumed to be 100. Thus $x_1 + x_2 \leq 100$.

If the weekly production plan (x_1, x_2) has to be fixed in advance and can not be changed during the week, and clients expect their actual demand to be satisfied during the corresponding week. We hope that the total production cost is as low as possible. Different from the method, i.e., stochastic programming with recourse, employed in [109], we formulate a chance constrained programming for this production process problem as follows:

$$\begin{cases} \min f(\mathbf{x}) = 2x_1 + 3x_2 \\ \text{subject to:} \\ \qquad \Pr\{(2+\eta_1)x_1 + 6x_2 \geq 180 + \xi_1\} \geq \alpha_1 \\ \qquad \Pr\{3x_1 + (3.4-\eta_2)x_2 \geq 162 + \xi_2\} \geq \alpha_2 \\ \qquad x_1 + x_2 \leq 100 \\ \qquad x_1, x_2 \geq 0. \end{cases} \tag{4.9}$$

When the confidence levels of α_1 and α_2 are assumed to be 0.8 and 0.7, respectively, the optimal production plan is

$$(x_1, x_2) = (33.2, 22.1)$$

with cost 132.7 when we run our genetic algorithm with 500 generations.

Feed Mixture Problem

Van de Panne and Popp[195] presented a chance constrained programming for feed mixture problem which is to select four materials to mix in order to design a cattle feed mix subject to protein and fat constraints with the objective of minimizing cost. That CCP may be written as follows:

$$\begin{cases} \min f(\mathbf{x}) = 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4 \\ \text{subject to:} \\ \qquad x_1 + x_2 + x_3 + x_4 = 1 \\ \qquad 2.3x_1 + 5.6x_2 + 11.1x_3 + 1.3x_4 \geq 5 \\ \qquad \Pr\{\eta_1 x_1 + \eta_2 x_2 + \eta_3 x_3 + \eta_4 x_4 \geq 21\} \geq \alpha \\ \qquad x_1, x_2, x_3, x_4 \geq 0 \end{cases} \tag{4.10}$$

where η_1, η_2, η_3 and η_4 have normal distributions $\mathcal{N}(12.0, 0.2809^2)$, $\mathcal{N}(11.9, 0.1936^2)$, $\mathcal{N}(41.8, 20.25^2)$ and $\mathcal{N}(52.1, 0.6241^2)$, respectively.

When α is assigned to be 0.8, a run of our computer program with 1000 generations shows that the optimal solution is

$$(x_1, x_2, x_3, x_4) = (0.001, 0.739, 0.053, 0.199)$$

whose cost is 30.25 and the actual reliability is just 80%.

Stochastic Resource Allocation

Let us consider a stochastic resource allocation problem in which there are multiple locations of resources and multiple users. The task of stochastic resource allocation is to determine the outputs that result from various combinations of resources such that the certain goal are achieved. As an example, our object of study is assumed to be a water supply-allocation system in which there are 3 locations of water and 4 users. The scheme of water supply-allocation system are shown by Figure 4.1.

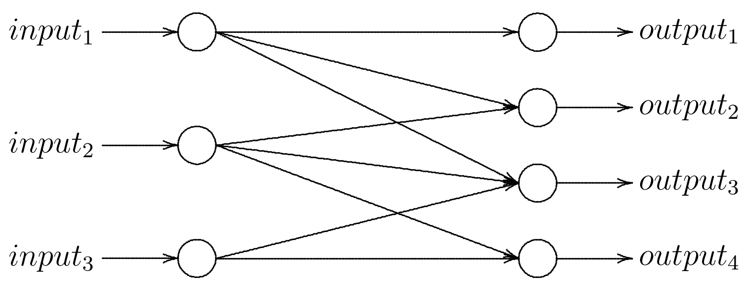

Figure 4.1: A Supply-Allocation System

In order to determine an optimal water allocation plan, we use 8 decision variables $x_1, x_2, \cdots, x_8$ to represent an action, where x_1, x_2, x_3 are quantities ordered from $input_1$ to outputs 1,2,3 respectively; x_4, x_5, x_6 from $input_2$ to outputs 2,3,4 respectively; x_7, x_8 from $input_3$ to outputs 3,4 respectively.

We mention that the inputs are available outside resources, they have their own properties. For example, the maximum quantities supplied by the three resources are marked by ξ_1, ξ_2, and ξ_3 which have two-parameter lognormal distributions $\mathcal{LOGN}(1.56, 0.56^2)$, $\mathcal{LOGN}(1.36, 0.45^2)$, and $\mathcal{LOGN}(0.95, 0.38^2)$, respectively. Thus at first we have the following constraint,

$$x_1 + x_2 + x_3 \le \xi_1, \qquad x_4 + x_5 + x_6 \le \xi_2, \qquad x_7 + x_8 \le \xi_3. \tag{4.11}$$

To handle the stochastic constraint (4.11), let $\alpha_1 = 0.9$, $\alpha_2 = 0.7$ and $\alpha_3 = 0.8$ be assigned as confidence levels of the three probabilistic constraints, respectively. The other clear constraint is $x_i \ge 0, \quad i = 1, 2, \cdots, 8$ which represent the quantities ordered from the resources are nonnegative.

The management goals are assumed to satisfy the demands of four users which are $3, 1, 2$ and 3 respectively. Then a chance constrained goal programming associated with this problem may be formulated as follows:

$$\begin{cases} \text{lexmin}\left\{d_1^-, d_2^-, d_3^-, d_4^-\right\} \\ \text{subject to:} \\ \qquad x_1 + d_1^- - d_1^+ = 3 \\ \qquad x_2 + x_4 + d_2^- - d_2^+ = 1 \\ \qquad x_3 + x_5 + x_7 + d_3^- - d_3^+ = 2 \\ \qquad x_6 + x_8 + d_4^- - d_4^+ = 3 \\ \qquad \Pr\{x_1 + x_2 + x_3 \le \xi_1\} \ge 0.9 \\ \qquad \Pr\{x_4 + x_5 + x_6 \le \xi_2\} \ge 0.7 \\ \qquad \Pr\{x_7 + x_8 \le \xi_3\} \ge 0.8 \\ \qquad x_i \ge 0, \quad i = 1, 2, \cdots, 8. \end{cases} \tag{4.12}$$

A run of computer program with 3000 generations shows that the optimal solution is

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (3.01, 0.00, 0.07, 1.31, 0.99, 0.15, 0.47, 1.07)$$

which can satisfy the first two goals, but the negative deviations of the third and fourth goals are 0.47 and 1.78, respectively.

An Abstract Example

Here we consider an abstract numerical example in which the objective function is multimodal and highly nonlinear.

$$\begin{cases} \max\left[x_1 \sin(x_1) + x_2 \sin(2x_2) + x_3 \sin(3x_3)\right] \\ \text{subject to:} \\ \qquad \Pr\left\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \le 10\right\} \ge 0.70 \\ \qquad \Pr\left\{\eta_1 x_1^2 + \eta_2 x_2^2 + \eta_3 x_3^2 \le 100\right\} \ge 0.80 \\ \qquad x_1, x_2, x_3 \ge 0 \end{cases} \tag{4.13}$$

where ξ_1, ξ_2, ξ_3 are random parameters with uniform distributions $\mathcal{U}[0.8, 1.2]$, $\mathcal{U}[1, 1.3]$, $\mathcal{U}[0.8, 1.0]$, respectively, η_1 has normal distribution $\mathcal{N}(1, 0.5)$, η_2 has exponential distribution $\mathcal{EXP}(1.2)$, η_3 has lognormal distribution $\mathcal{LONG}(0.8, 0.6)$.

We perform our computer program with 3000 generations and obtain the optimal solution

$$(x_1, x_2, x_3) = (7.8715, 0.9016, 0.6650)$$

with objective value 9.3537. The progress of evolution is shown in Figure 4.2.

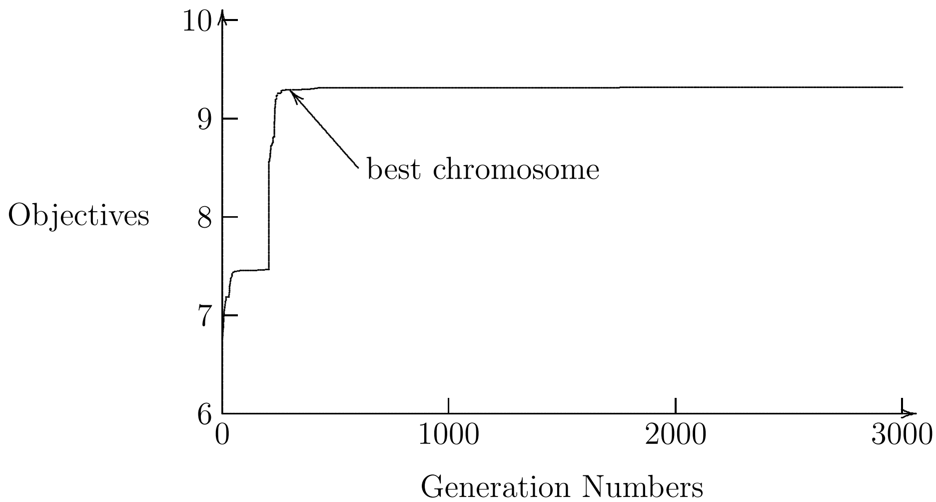

Figure 4.2: Progress of Evolution

4.2.4 Conclusion

The genetic algorithm provides an effective means to consider chance constrained programming, including chance constrained multiobjective programming and chance constrained goal programming. An advantage of genetic algorithm is to obtain the global optima fairly well, as well as the advantages of other genetic algorithm for different problems with multimodal objective functions. The other advantage is that we do not need to convert the stochastic constraints into their deterministic equivalents, where the translation is usually a hard task. This ensures that we can deal with more general chance constrained programming. In the proposed genetic algorithm, Monte Carlo simulation is also employed to check the feasibility of solutions. The effectiveness of genetic algorithm has been illustrated by a set of test problems.

Appendix

A random variable x has a uniform distribution $\mathcal{U}[a,b]$ if its probability density function is

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & otherwise. \end{cases}$$

A random variable x has an exponential distribution $\mathcal{EXP}(\beta)$ if its probability density function is

$$f(x) = \begin{cases} \frac{1}{\beta}e^{-x/\beta}, & 0 \le x < \infty, \beta > 0 \\ 0, & otherwise. \end{cases}$$

A random variable x has a normal distribution $\mathcal{N}(\mu, \sigma^2)$ if its probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \qquad -\infty < x < +\infty.$$

A random variable x has a two-parameter lognormal distribution $\mathcal{LOGN}(\mu, \sigma^2)$ if its probability density function is

$$f(x) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma x} \exp\left[-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right], & 0 < x < \infty \\ 0, & \textit{otherwise.} \end{cases}$$

4.3 Chance Constrained Integer Programming Models for Capital Budgeting in Fuzzy Environments

The original capital budgeting is concerned with maximizing the total net profit subject to budget constraint by selecting appropriate combination of projects. With the requirement of considering uncertainty of future demand and multiple conflicting goals, chance constrained integer goal programming was employed to model capital budgeting by Keown and Martin[111] in the working capital management and by Keown and Taylor[112] in the production area. In addition, De *et al.*[1] extended chance constrained goal programming to the zero-one case and applied it to capital budgeting problems.

When some parameters of decision systems are fuzzy numbers rather than stochastic variables, Liu and Iwamura[149] suggested the framework of chance constrained programming as well as chance constrained multiobjective programming and chance constrained goal programming in a fuzzy environment based on possibility theory. In order to deal with the chance constraints (represented by possibility), a technique of fuzzy simulation was presented. A fuzzy simulation based genetic algorithm was also designed for solving chance constrained programming models with fuzzy parameters.

This section will extend chance constrained programming with fuzzy parameters to integer case and apply it to capital budgeting problems in fuzzy environments. A fuzzy simulation based genetic algorithm is also designed for solving chance constrained integer programming models with fuzzy parameters. Finally, we presents some numerical examples to illustrate the models and algorithms.

4.3.1 Capital Budgeting

Consider a company which has the opportunity to initiate the machines in a plant. Suppose that there are n types of machines available. We use x_i to denote the numbers of type i machines selected, $i = 1, 2, \cdots, n$, respectively. Then all the variables x_i's are nonnegative integers. Let a_i be the level of funds that needs to be allocated to type i machine and a be the total capital available for distribution, then we have

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \le a, \tag{4.14}$$

i.e., can not exceed the amount available.

The other constraint is the maximum space availability limitation for the machines. Suppose that b_i are the spaces used by per type i machine, $i = 1, 2, \cdots, n$, respectively. If the total available space is b, then we have the following constraint,

$$b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \le b. \tag{4.15}$$

We suppose that different machines produce different products. Let η_i be the production capacity of the type i machine for product i, then the total products i are $\eta_i x_i$, $i = 1, 2, \cdots, n$, respectively. We also assume that the future demands for products i are ξ_i, $i = 1, 2, \cdots, n$. Since the production should satisfy the future demand, we have

$$\eta_i x_i \ge \xi_i, \quad i = 1, 2, \cdots, n. \tag{4.16}$$

If c_i are the net profits per type i machine, $i = 1, 2, \cdots, n$, then the total net profit is $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$. Our objective is to maximize the total net profit, i.e.,

$$\max c_1 x_1 + c_2 x_2 + \cdots + c_n x_n. \tag{4.17}$$

Thus we have a deterministic model for capital budgeting based on integer programming,

$$\begin{cases} \max c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{subject to:} \\ \qquad a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \le a \\ \qquad b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \le b \\ \qquad \eta_i x_i \ge \xi_i, \quad i = 1, 2, \cdots, n \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers.} \end{cases} \tag{4.18}$$

The capital budgeting problem (4.18) is clearly a general case of the knapsack problem which is concerned with trying to fill a knapsack to maximum total value,

$$\begin{cases} \max c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} \\ \qquad a_1x_1 + a_2x_2 + \cdots + a_nx_n \le a \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers} \end{cases} \tag{4.19}$$

where $(c_1, c_2, \cdots, c_n)$ and $(a_1, a_2, \cdots, a_n)$ are the respective values and weights of the n objects and a is the overall weight limitation.

Certainly, real capital budgeting problems are much more complex than the above-mentioned model. However, it is enough for illustrating the chance constrained integer programming with fuzzy parameters.

4.3.2 Chance Constrained Programming Models

Chance constrained programming was pioneered by Charnes and Cooper[18][19][20] as a means of handling uncertainty by specifying a confidence level at which it is desired that the uncertain constraint holds. In this subsection let us model the capital budgeting problems by chance constrained integer programming based on the works[111][112][1].

In practice, the production capacities η_i and future demands ξ_i are not necessarily deterministic. Here we suppose that they are stochastic variables. Let ψ_i and ϕ_i denote the probability density functions of η_i and ξ_i, $i = 1, 2, \cdots, n$, respectively. Then the constraints $\eta_i x_i \ge \xi_i$ are uncertain. Suppose that the manager gives α_i as the probabilities of meeting the demands of products i, $i = 1, 2, \cdots, n$, respectively. Then we have the following chance constraints,

$$\Pr\{\eta_i x_i \ge \xi_i\} \ge \alpha_i, \quad i = 1, 2, \cdots, n \tag{4.20}$$

where $\Pr\{\cdot\}$ denotes the probability of the event $\{\cdot\}$. Thus, a chance constrained integer programming is immediately formulated as follows,

$$\begin{cases} \max c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} \\ \qquad a_1x_1 + a_2x_2 + \cdots + a_nx_n \le a \\ \qquad b_1x_1 + b_2x_2 + \cdots + b_nx_n \le b \\ \qquad \Pr\{\eta_i x_i \ge \xi_i\} \ge \alpha_i, \quad i = 1, 2, \cdots, n \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers} \end{cases} \tag{4.21}$$

where the separate chance constraints $\Pr\{c_i x_i \ge d_i\} \ge \alpha_i$, $i = 1, 2, \cdots, n$ may be replaced by a joint form

$$\Pr\{\eta_i x_i \ge \xi_i, i = 1, 2, \cdots, n\} \ge \alpha$$

or some mixed forms.

Now we suppose that the following target levels and priority structure have been set by the manager: Priority 1: Budget goal, i.e.,

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + d_1^- - d_1^+ = a$$

where d_1^+ will be minimized. Priority 2: Space goal, i.e.,

$$b_1x_1 + b_2x_2 + \cdots + b_nx_n + d_2^- - d_2^+ = b$$

where d_2^+ will be minimized. Priority 3: Profit goal, i.e.,

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n + d_3^- - d_3^+ = c$$

where d_3^- will be minimized. We also suppose that the probability of satisfying the demand is at least α, i.e., $\Pr\{\eta_i x_i \geq \xi_i, i = 1, 2, \cdots, n\} \geq \alpha$, and all the variables x_i, $i = 1, 2, \cdots, n$ are nonnegative integers. Then we have a chance constrained goal programming as follows,

$$\begin{cases} \text{lexmin}\left\{d_1^+, d_2^+, d_3^-\right\} \\ \text{subject to:} \\ \qquad a_1x_1 + a_2x_2 + \cdots + a_nx_n + d_1^- - d_1^+ = a \\ \qquad b_1x_1 + b_2x_2 + \cdots + b_nx_n + d_2^- - d_2^+ = b \\ \qquad c_1x_1 + c_2x_2 + \cdots + c_nx_n + d_3^- - d_3^+ = c \\ \qquad \Pr\{\eta_i x_i \geq \xi_i, i = 1, 2, \cdots, n\} \geq \alpha \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers.} \end{cases} \tag{4.22}$$

Chance constrained programming models can be converted into deterministic equivalents when the random variables are normally distributed. However, it is very difficult to transform them to deterministic forms if the distributions of random variables belong to other classes. In order to solve general chance constrained programming models, Iwamura and Liu[98] proposed a stochastic simulation based genetic algorithm in which the stochastic simulation is used to check the chance constraints.

4.3.3 Modelling Capital Budgeting in Fuzzy Environment

We have discussed the chance constraints

$$\Pr\{\eta_i x_i \geq \xi_i\} \geq \alpha_i, \quad i = 1, 2, \cdots, n$$

where η_i and ξ_i are assumed random variables with known density functions and α_i are predetermined confidence levels. It is well-known that the density functions are generated by repetitions of experiments. However, in many cases, we have no such an experiment when we initiate the machines in a plant. Meanwhile, we have to regard η_i and ξ_i as fuzzy numbers and construct their membership functions by some expert knowledge. In this paper, we assume that the membership functions of η_i and ξ_i are all given. If we hope that the possibility of satisfying the demands ξ_i are at least α_i, $i = 1, 2, \cdots, n$, respectively, then we have chance constraints in a fuzzy environment as follows,

$$\text{Pos}\,\{\eta_i x_i \geq \xi_i\} \geq \alpha_i, \quad i = 1, 2, \cdots, n$$

where Pos represents the possibility. More generally, assume that we can substitute some products for others, for example, we have p classes of demands denoted by ξ_j, and the production capacities of the type i machines for the product classes j are η_{ij}, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, p$, respectively, then the chance constraints are written as

$$\text{Pos}\,\{\eta_{1j}x_1 + \eta_{2j}x_2 + \cdots + \eta_{nj}x_n \geq \xi_j\} \geq \alpha_i, \quad j = 1, 2, \cdots, p \tag{4.23}$$

or written as a joint form

$$\text{Pos}\,\{\eta_{1j}x_1 + \eta_{2j}x_2 + \cdots + \eta_{nj}x_n \geq \xi_j, j = 1, 2, \cdots, p\} \geq \alpha \tag{4.24}$$

where α is a predetermined confidence level. In some special cases, for example, all fuzzy numbers are trapezoidal, the chance constraints (4.23) can be converted to crisp equivalents. For detailed expositions, the readers may consult Liu and Iwamura[149].

The simplest chance constrained integer programming with fuzzy parameters for capital budgeting is

$$\begin{cases} \max c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} \\ \qquad a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq a \\ \qquad b_1x_1 + b_2x_2 + \cdots + b_nx_n \leq b \\ \qquad \text{Pos}\,\{\eta_{1j}x_1 + \eta_{2j}x_2 + \cdots + \eta_{nj}x_n \geq \xi_j, j = 1, 2, \cdots, p\} \geq \alpha \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers.} \end{cases} \tag{4.25}$$

In order to balance the multiple conflicting objectives, capital budgeting may be modelled by the following chance constrained goal programming with fuzzy parameters according to the target levels and priority structure set by the decision maker,

$$\begin{cases} \text{lexmin}\left\{d_1^+, d_2^+, d_3^-\right\} \\ \text{subject to:} \\ \qquad a_1x_1 + a_2x_2 + \cdots + a_nx_n + d_1^- - d_1^+ = a \\ \qquad b_1x_1 + b_2x_2 + \cdots + b_nx_n + d_2^- - d_2^+ = b \\ \qquad c_1x_1 + c_2x_2 + \cdots + c_nx_n + d_3^- - d_3^+ = c \\ \qquad \text{Pos}\left\{\eta_{1j}x_1 + \eta_{2j}x_2 + \cdots + \eta_{nj}x_n \geq \xi_j, j = 1, 2, \cdots, p\right\} \geq \alpha \\ \qquad x_i, i = 1, 2, \cdots, n, \quad \text{nonnegative integers.} \end{cases} \tag{4.26}$$

4.3.4 Fuzzy Simulation Based Genetic Algorithm

Genetic algorithms are a stochastic search method for optimization problems based on the mechanics of natural selection and natural genetics, i.e., the principle of evolution—survival of the fittest. Genetic algorithms have demonstrated considerable success in providing good solutions to many complex optimization problems and received more and more attentions during the past three decades. When the objective functions to be optimized in the optimization problems are multimodal or the search spaces are particularly irregular, algorithms need to be highly robust in order to avoid getting stuck at local optimal solution. The advantage of genetic algorithms is just to obtain the global optimal solution fairly. Genetic algorithms (including evolution program and evolution strategies) have been well discussed and summarized by numerous literatures, such as Goldberg[54], Michalewicz[156] and Fogel[34], and applied to a wide variety of problems, such as optimal control problems, transportation problems, traveling salesman problems, drawing graphs, scheduling, group technology, facility layout and location, as well as pattern recognition.

In this subsection, we design a fuzzy simulation based genetic algorithm for chance constrained integer programming models with fuzzy parameters in which the chance constraints are not assumed to have known crisp equivalent forms. We will discuss representation structure, handling constraints, initialization process, evaluation function, selection process, crossover operation and mutation operation in turn.

Representation Structure

There are two ways to represent a solution of an optimization problem, binary vector and floating vector. We can use a binary vector as a chromosome to represent real value of decision variable, where the length of the vector depends on the required precision. The necessity for binary codings has received considerable criticism.

An alternative approach to represent a solution is the floating point implementation in which each chromosome vector is coded as a vector of floating numbers, of the same length as the solution vector. Here we use a vector $V = (x_1, x_2, \cdots, x_n)$ as a chromosome to represent a solution to the optimization problem, where n is the dimension. Certainly, all variables x_i's will be confined to integer values. In fact, if we code the algorithm by C language, then we can ensure that the vector V is integer by defining it as an integer array.

Fuzzy Simulation

Fuzzy simulation was proposed by Liu and Iwamura[149] as a means of handling the possibility constraint $\text{Pos}\{\xi | g_i(\mathbf{x}, \xi) \le 0, i = 1, 2, \cdots, k\} \ge \alpha$ where ξ is a vector of fuzzy numbers. Although this chance constraint can be represented as an explicit form for some special cases, we need a numerical method for general cases. We will call the simulation to check the fuzzy constraints as *fuzzy simulation*.

Here the key problem is to check whether the chance constraint

$$\text{Pos}\{\eta_{1j}x_1 + \eta_{2j}x_2 + \cdots + \eta_{nj}x_n \ge \xi_j, j = 1, 2, \cdots, p\} \ge \alpha \tag{4.27}$$

holds or not. From the definition of operations over fuzzy numbers[203][29][30], we say the chance constraint (4.27) holds for a given decision vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ if and only if there is a crisp array $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ such that

$$\eta_{1j}^0 x_1 + \eta_{2j}^0 x_2 + \cdots + \eta_{nj}^0 x_n \ge \xi_j^0, \quad j = 1, 2, \cdots, p \tag{4.28}$$

with an inequality

$$\min_{1 \le j \le p} \left\{ \min \left\{ \mu_{\eta_{kj}}(\eta_{kj}^0), k = 1, 2, \cdots, n, \mu_{\xi_j}(\xi_j^0) \right\} \right\} \ge \alpha. \tag{4.29}$$

Thus, we can generate an array $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ uniformly from the α-cut set of fuzzy array $(\eta_{1j}, \eta_{2j}, \cdots, \eta_{nj}, \xi_j)_{j=1}^p$. If $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ satisfies the system of inequalities (4.28), then we can believe the chance constraint (4.27). If not, we will re-generate an array

$$(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$$

uniformly from the α-cut set of fuzzy array $(\eta_{1j}, \eta_{2j}, \cdots, \eta_{nj}, \xi_j)_{j=1}^p$ by that way and check the constraint. After a given number of cycles, if no feasible $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ is generated, then we say that the given chromosome $V = (x_1, x_2, \cdots, x_n)$ is infeasible. Now we summarize the above process as follows.

Step 1. Generate $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ uniformly from the α-cut set of the fuzzy array.

Step 2. If (4.28) holds, return FEASIBLE.
Step 3. Repeat Steps 1 and 2 N times.
Step 4. Return INFEASIBLE.

Remark: If the α-level set of the fuzzy vector is too complex to determine, we can sample a vector $(\eta_{1j}^0, \eta_{2j}^0, \cdots, \eta_{nj}^0, \xi_j^0)_{j=1}^p$ from a hypercube Ω containing the α-level set and then accept or reject it, depending on whether (4.28) holds or not.

Initialization Process

We define an integer *pop_size* as the number of chromosomes and initialize *pop_size* chromosomes randomly. Usually, it is difficult for complex optimization problems to produce feasible chromosome explicitly. So we employ one of the following two ways as the initialization process, depending on what kind of information the decision maker can give.

First case is that the decision maker can determine an interior point, denoted by V_0, in the constraint set. This is very possible for real decision problems. We also need to define a large positive number M which ensures that all the genetic operators are probabilitistically complete for the feasible solutions. This number M is used for not only initialization process but also mutation operation. The *pop_size* chromosomes will be produced as follows. We randomly select a direction d in $\Re^n$ and define a chromosome V as $V_0+M\cdot d$ if it is feasible, otherwise, we set M by a random number between 0 and M until $V_0 + M \cdot d$ is feasible. We mention that a feasible solution can be found in finite times by taking random number since V_0 is an interior point. Repeat this process *pop_size* times and produce *pop_size* initial feasible solutions $V_1, V_2, \cdots, V_{pop\_size}$.

If the decision maker fails to give such an interior point, but can predetermine a region which contains the feasible set. Usually, this region will be designed to have nice sharp, for example, an n-dimensional hypercube, because the computer can easily sample points from a hypercube. We generate a random point from the hypercube and check the feasibility of this point by the fuzzy simulation. If it is feasible, then it will be accepted as a chromosome. If not, then re-generate a point from the hypercube randomly until a feasible one is obtained. Repeat the above process *pop_size* times, we can make *pop_size* initial feasible chromosomes $V_1, V_2, \cdots, V_{pop\_size}$.

Evaluation Function

Evaluation function, denoted by $eval(V)$, is to assign a probability of reproduction to each chromosome V so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population, that is, the chromosomes with higher fitness will have more chance to produce offsprings by using *roulette wheel selection*.

Let $V_1, V_2, \cdots, V_{pop\_size}$ be the *pop_size* chromosomes at the current generation. One well-known method is based on allocation of reproductive trials according to rank rather than actual objective values. No matter what kind of mathematical programming (single-objective, multiobjective or goal programming), it is reasonable to assume that the decision maker can give an order relationship among the *pop_size* chromosomes $V_1, V_2, \cdots, V_{pop\_size}$ such that the *pop_size* chromosomes can be rearranged from good to bad, i.e., the better the chromosome is, the smaller ordinal number it has. Now let a parameter $a \in (0, 1)$ in the genetic system be given, then we can define the so-called *rank-based evaluation function* as follows,

$$eval(V_i) = a(1-a)^{i-1}, \qquad i = 1, 2, \cdots, pop\_size. \tag{4.30}$$

We mention that $i = 1$ means the best individual, $i = pop\_size$ the worst individual.

Selection Process

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select a single chromosome for a new population in the following way:

Step 1. Calculate the cumulative probability q_i for each chromosome V_i,

$$\begin{aligned} q_0 &= 0 \\ q_i &= \sum_{j=1}^{i} eval(V_j), \quad i = 1, 2, \cdots, pop\_size. \end{aligned} \tag{4.31}$$

Step 2. Generate a random real number r in $[0, q_{pop\_size}]$.

Step 3. Select the i-th chromosome V_i $(1 \le i \le pop\_size)$ such that $q_{i-1} < r \le q_i$.

Step 4. Repeat steps 2 and 3 *pop_size* times and obtain *pop_size* copies of chromosomes.

Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. This probability gives us the expected number $P_c \cdot pop\_size$ of chromosomes which undergo the crossover operation.

In order to determine the parents for crossover operation, let us do the following process repeatedly from $i = 1$ to *pop_size*: generating a random real number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent if $r < P_c$.

We denote the selected parents as $V_1', V_2', V_3', \cdots$ and divide them to the following pairs:

$$(V_1', V_2'), \quad (V_3', V_4'), \quad (V_5', V_6'), \quad \cdots$$

Let us illustrate the crossover operator on each pair by (V_1', V_2'). At first, generate a random number c from the open interval $(0, 1)$, then the crossover operator on V_1' and V_2' will produce two children X and Y as follows:

$$X = c \cdot V_1' + (1 - c) \cdot V_2' \quad \& \quad Y = (1 - c) \cdot V_1' + c \cdot V_2'. \tag{4.32}$$

If the feasible set is convex, this arithmetical crossover operation ensures that both children are feasible if both parents are. However, in many cases, the feasible set is not necessarily convex or hard to verify the convexity. So we must check the feasibility of each child by fuzzy simulation. If both children are feasible, then we replace the parents by them. If not, we keep the feasible one if exists, and then re-do the crossover operator by regenerating the random number c until two feasible children are obtained or a given number of cycles is finished. In this case, we only replace the parents by the feasible children.

Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. This probability gives us the expected number of $P_m \cdot pop\_size$ of chromosomes which undergo the mutation operations.

Similar to the process of selecting parents for crossover operation, we repeat the following steps from $i = 1$ to *pop_size*: generating a random real number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent for mutation if $r < P_m$.

For each selected parent, denoted by $V = (x_1, x_2, \cdots, x_n)$, we mutate it by the following way. We choose a mutation direction d in $\Re^n$ randomly, if $V + M \cdot d$ is not feasible for the constraints, then we set M as a random number between 0 and M until it is feasible, where M is a large positive number defined in the subsection of Initialization Process. If the above process can not find a feasible solution in a predetermined number of iterations, then sets $M = 0$. We replace the parent V by the new chromosome

$$V' = V + M \cdot d. \tag{4.33}$$

Procedure Genetic Algorithm

Following selection, crossover and mutation, the new population is ready for its next evaluation. The genetic algorithm will terminate after a given number of cyclic repetitions of the above steps. We can summarize the genetic algorithm for chance constrained programming with fuzzy parameters as follows.

Procedure Genetic Algorithm

Input parameters: pop_size, P_c, P_m;
Initialize the chromosomes by Initialization Process;
REPEAT
  Update chromosomes by crossover and mutation operators;
  Compute the evaluation function for all chromosomes;
  Select chromosomes by sampling mechanism;
UNTIL(*termination_condition*)

It is known that the best chromosome does not necessarily appear in the last generation. So we have to keep the best one from the beginning. If we find a better one in the new population, then replace the old one by it. This chromosome will be reported as the solution after finishing the evolutions.

4.3.5 Numerical Examples

The computer code for the genetic algorithm to chance constrained integer programming with fuzzy parameters has been written in C language. To illustrate the effectiveness of genetic algorithm, a set of numerical examples has been done, and the results are successful. Here we give some numerical examples which are all performed on a workstation with the following parameters: the population size is 30, the probability of crossover P_c is 0.2, the probability of mutation P_m is 0.4, the parameter a in the rank-based evaluation function is 0.05.

Suppose that we have five types of machines. According to the discussion in Section 4.3.3, when our objective is to maximize the total profit, the capital budgeting model is formulated as follows,

$$\begin{cases} \max\ 3x_1 + x_2 + 2x_3 + 3x_4 + x_5 \\ \text{subject to:} \\ \qquad 2x_1 + x_2 + 3x_3 + 6x_4 + 4x_5 \le 50 \\ \qquad 7x_1 + 6x_2 + 4x_3 + 8x_4 + x_5 \le 100 \\ \qquad \text{Pos}\left\{ \begin{array}{l} \eta_{11}x_1 + \eta_{21}x_2 + \eta_{31}x_3 \ge \xi_1 \\ \eta_{32}x_3 + \eta_{42}x_4 + \eta_{52}x_5 \ge \xi_2 \end{array} \right\} \ge 0.9 \\ \qquad x_1, x_2, x_3, x_4, x_5, \quad \text{nonnegative integers} \end{cases}$$

where η_{11} is a triangular fuzzy number (13,14,15), η_{21} is a fuzzy number with membership function

$$\mu_{\eta_{21}}(u) = \exp\left[-(u-8)^2\right],$$

η_{31} is a fuzzy number with membership function

$$\mu_{\eta_{31}}(u) = \begin{cases} \dfrac{1}{u-9}, & u \geq 10 \\ 0, & u < 10, \end{cases}$$

the demand of the first product ξ_1 is a fuzzy number with membership function

$$\mu_{\xi_1}(u) = \exp\left[-|u-50|\right],$$

η_{32} is a trapezoidal fuzzy number (8,9,10,11), η_{42} is a triangular fuzzy number (10,11,12), η_{52} is a fuzzy number with membership function

$$\mu_{\eta_{52}}(u) = \exp\left[-|u-10|\right],$$

and the demand of the second product ξ_2 is a triangular fuzzy number (30,40,50). A run of the fuzzy simulation based genetic algorithm with 300 generations shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*) = (10, 0, 7, 0, 1)$$

whose total profit is 45.

If the goal hierarchy is (i) budget goal, (ii) space goal, and (iii) profit goal, then we can model the capital budgeting problem by the following chance constrained integer goal programming with fuzzy parameters,

$$\begin{cases} \text{lexmin } \left\{d_1^+, d_2^+, d_3^-\right\} \\ \text{subject to:} \\ \qquad 2x_1 + x_2 + 3x_3 + 6x_4 + 4x_5 + d_1^- - d_1^+ = 50 \\ \qquad 7x_1 + 6x_2 + 4x_3 + 8x_4 + x_5 + d_2^- - d_2^+ = 100 \\ \qquad 3x_1 + x_2 + 2x_3 + 3x_4 + x_5 + d_3^- - d_3^+ = 50 \\ \qquad \text{Pos}\left\{ \begin{array}{l} \eta_{11}x_1 + \eta_{21}x_2 + \eta_{31}x_3 \geq \xi_1 \\ \eta_{32}x_3 + \eta_{42}x_4 + \eta_{52}x_5 \geq \xi_2 \end{array} \right\} \geq 0.9 \\ \qquad x_1, x_2, x_3, x_4, x_5, \quad \text{nonnegative integers} \end{cases}$$

where the parameters are defined as above. A run of the computer program with 400 generations shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*) = (10, 0, 7, 0, 1)$$

which can satisfy the first two goals, but the negative deviation of the third goal is 5.

4.3.6 Conclusion

In this section we extended chance constrained programming with fuzzy parameters to integer case and applied to capital budgeting problems in fuzzy environments. A fuzzy simulation based genetic algorithm was also designed for solving chance constrained integer programming models with fuzzy parameters. The time complexity of chance constrained integer programming with fuzzy parameters is the sum of the time spent for the fuzzy simulation and the time spent for the genetic algorithm, where the computation time for fuzzy simulation has to be spent since we have assumed that there is no direct method to substitute for it.

4.4 Topological Optimization Models for Communication Network with Multiple Reliability Goals

An important problem appearing in computer-communication network is to design an optimal topology for balancing system reliability and cost. When the reliability of nodes and communication links of a network is given, the system reliability is dependent on how nodes are connected by communication links. There are mainly two types of way, one is to minimize the total cost subject to a reliability constraint, while the other is to maximize the reliability subject to a cost constraint, for example, Aggarwal et al. [2][3], Chopra et al. [21].

Jan et al. [108] designed a branch-and-bound algorithm to minimize the total cost subject to a reliability constraint. It has been proved that communication network reliability problems are NP-hard. So some heuristic algorithms are designed to solve the problem of larger network. Chopra et al. [21] and Aggarwal et al. [2][3] employed greedy heuristic approaches for maximizing the overall and terminal reliability. Ravi et al. [171] designed a nonequilibrium simulated annealing algorithm. Painton and Campbell [168] presented a genetic algorithm for optimizing the system reliability. Kumar et al. [131][132] proposed a genetic algorithm for solving various network expansion problems, such as minimizing diameter, minimizing average distance, and maximizing computer-network reliability. Dengiz et al. [26] presented a genetic algorithm for optimization of all-terminal reliable networks.

In practice, a large network consists of a backbone network and several local access networks. This fact provides a motivation to develop topological optimization models with multiple reliability goals. In this paper, we will also design a stochastic simulation-based genetic algorithm for solving the proposed models and illustrate its effectiveness by some numerical examples.

4.4.1 Topological Models

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$ be a communication network in which $\mathcal{V}$ and $\mathcal{E}$ correspond to terminals and links, and $\mathcal{P}$ is the set of reliabilities for the links $\mathcal{E}$. If there are n vertices (terminals), then the links $\mathcal{E}$ may also be represented by the link topology of $\boldsymbol{x} = \{x_{ij} : 1 \le i \le n-1, i+1 \le j \le n\}$, where $x_{ij} \in \{0, 1\}$, and $x_{ij} = 1$ means the link (i, j) is selected, 0 otherwise.

If we assume that the terminals are perfectly reliable and links fail s-independently with known probabilities, then the success of communication between terminals in subset $\mathcal{K}$ of $\mathcal{V}$ is a random event. The probability of this event is called the $\mathcal{K}$-terminal reliability denoted by $R(\mathcal{K}, \boldsymbol{x})$ when the link topology is $\boldsymbol{x}$. A network $\mathcal{G}$ is called $\mathcal{K}$-connected if all the vertices in $\mathcal{K}$ are connected in $\mathcal{G}$. Thus the $\mathcal{K}$-terminal reliability $R(\mathcal{K}, \boldsymbol{x})$ is $\Pr\{\mathcal{G}$ is $\mathcal{K}$-connected with respect to $\boldsymbol{x}\}$. Notice that when $\mathcal{K} \equiv \mathcal{V}$, then $R(\mathcal{K}, \boldsymbol{x})$ is the overall reliability.

In addition, for each candidate link topology $\boldsymbol{x}$, the overall cost should be $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_{ij}$, where c_{ij} is the cost of link (i, j), $i = 1, 2, \cdots, n-1$, $j = i+1, i+2, \cdots, n$, respectively.

If we want to minimize the total cost subject to multiple reliability constraints, then we have

$$\begin{cases} \min \displaystyle\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_{ij} \\ \text{subject to:} \\ \qquad R(\mathcal{K}_k, \boldsymbol{x}) \ge R_k, \ \ k = 1, 2, \cdots, m \end{cases} \tag{4.34}$$

where $\mathcal{K}_k$ are target sets of $\mathcal{G}$, R_k are predetermined minimum reliabilities, $k = 1, 2, \cdots, m$, respectively. This is a type of chance-constrained programming.

If we want to maximize the $\mathcal{K}$-terminal reliability subject to a cost constraint, then we have the following dependent-chance programming model,

$$\begin{cases} \max R(\mathcal{K}, \boldsymbol{x}) \\ \text{subject to:} \\ \qquad \sum\limits_{i=1}^{n-1}\sum\limits_{j=i+1}^{n} c_{ij}x_{ij} \le c_0 \end{cases} \tag{4.35}$$

where c_0 is the maximum capital available.

Now we assume that $\mathcal{K}_1$, $\mathcal{K}_2$, $\cdots$, $\mathcal{K}_m$ are m target sets of $\mathcal{G}$, then we have a dependent-chance multiobjective programming model,

$$\begin{cases} \max\, [R(\mathcal{K}_1, \boldsymbol{x}), R(\mathcal{K}_2, \boldsymbol{x}), \cdots, R(\mathcal{K}_m, \boldsymbol{x})] \\ \text{subject to:} \\ \qquad \sum\limits_{i=1}^{n-1}\sum\limits_{j=i+1}^{n} c_{ij}x_{ij} \le c_0. \end{cases} \tag{4.36}$$

We can also formulate the topological optimization problem for communication network reliability as a dependent-chance goal programming according to the priority structure and target levels set by the decision maker,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ + v_{ij} d_i^-) \\ \text{subject to:} \\ \qquad R(\mathcal{K}_i, \boldsymbol{x}) + d_i^- - d_i^+ = R_i, \quad i = 1, 2, \cdots, m \\ \qquad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \le c_0 \\ \qquad d_i^-, d_i^+ \ge 0, \quad i = 1, 2, \cdots, m \end{cases} \tag{4.37}$$

where P_j = the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, for all j, u_{ij} = weighting factor corresponding to positive deviation for goal i with priority j assigned, v_{ij} = weighting factor corresponding to negative deviation for goal i with priority j assigned, d_i^+ = positive deviation from the target of goal i, d_i^- = negative deviation from the target of goal i, R_i = the target reliability level of the set $\mathcal{K}_i$, l = number of priorities, m = number of goal constraints.

4.4.2 $\mathcal{K}$-terminal Reliability

After a link topology $\boldsymbol{x}$ is given, we should estimate the $\mathcal{K}$-terminal reliability $R(\mathcal{K}, \boldsymbol{x})$ with respect to some prescribed target set $\mathcal{K}$. Estimating $\mathcal{K}$-terminal reliability has received considerable attention during the past two decades. It is almost impossible to design an algorithm to compute $R(\mathcal{K}, \boldsymbol{x})$ analytically. In order to handle larger network, we may employ the stochastic simulation (Monte Carlo simulation) which consists of repeating s-independently N times trials.

Step 1. Set counter $N' = 0$;

Step 2. Randomly generate an operational link set $\mathcal{E}'$ from the link topology $\boldsymbol{x}$ according to $\mathcal{P}$;

Step 3. If $(\mathcal{V}, \mathcal{E}')$ is $\mathcal{K}$-connected, then $N' + +$;

Step 4. Repeat the second and third steps N times;

Step 5. $R(\mathcal{K}, \boldsymbol{x}) = N'/N$.

4.4.3 Stochastic Simulation-based Genetic Algorithm

Genetic algorithms are a stochastic search method for optimization problems based on the mechanics of natural selection and natural genetics. Genetic algorithms have demonstrated considerable success in providing good solutions to many complex optimization problems and received more and more attentions during the past three decades. When the objective functions to be optimized in the optimization problems are multimodal or the search spaces are particularly irregular, algorithms need to be highly robust in order to avoid getting stuck at local optimal solution. The advantage of genetic algorithms is just to obtain the global optimal solution fairly. Genetic algorithms (including evolution programs and evolution strategies) have been well documented in the literature, such as in Holland [64], Goldberg [54] and Michalewicz [156], and applied to a wide variety of optimization problems. Especially, for chance-constrained programming, Iwamura and Liu [98] provided a stochastic simulation-based genetic algorithm for stochastic case; Liu and Iwamura [149][150] provided a fuzzy simulation-based genetic algorithm for fuzzy models. The dependent-chance programming models have also been solved by the simulation-based genetic algorithm for stochastic case [143][144] and for fuzzy case [145][146]. For detailed expositions, the reader may consult Liu [147].

In this subsection, we present a stochastic simulation-based genetic algorithm for solving the topological optimization models for communication network reliability.

Representation Structure

Now we use an $n(n-1)/2$-dimensional vector $V = (y_1, y_2, \cdots, y_{n(n-1)/2})$ as a chromosome to represent a candidate link topology $\boldsymbol{x}$, where y_i is taken as 0 or 1 for $1 \le i \le n(n-1)/2$. Then the relationship between a link topology and a chromosome is

$$x_{ij} = y_{(2n-i)(i-1)/2+j-i}, \quad 1 \le i \le n-1,\ i+1 \le j \le n. \tag{4.38}$$

Initialization Process

We set y_i as a random integer from $\{0, 1\}$, $i = 1, 2, \cdots, n(n-1)/2$, respectively. If the generated chromosome $V = (y_1, y_2, \cdots, y_{n(n-1)/2})$ is proven to be feasible, then it is accepted as a chromosome, otherwise we repeat the above process until a feasible chromosome is obtained. We may generate *pop_size* initial chromosomes $V_1, V_2, \cdots, V_{pop\_size}$ by repeating the above process *pop_size* times.

Evaluation Function

The evaluation function, denoted by $eval(V)$, assigns a probability of reproduction to each chromosome V so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population, that is, the chromosomes with higher fitness will have a greater chance of producing offspring through roulette wheel selection.

Let $V_1, V_2, \cdots, V_{pop\_size}$ be the *pop_size* chromosomes in the current generation. At first we calculate the objective values of the chromosomes. According to the objective values, we can rearrange these chromosomes $V_1, V_2, \cdots, V_{pop\_size}$ from good to bad (i.e., the better the chromosome, the smaller the ordinal number). For the single-objective case, a chromosome with larger objective value is better; for the multiobjective case, we may define a preference function to evaluate the chromosomes; for the goal programming case, we have the following order relationship for the chromosomes: for any two chromosomes, if the higher-priority objectives are equal, then, at the current priority level, the one with a minimal objective value is better, and if two different chromosomes have the same objective values at every level, then we are indifferent between them. Now let a parameter $a \in (0, 1)$ in the genetic system be given, then we can define the so-called *rank-based evaluation function* as follows,

$$eval(V_i) = a(1-a)^{i-1}, \qquad i = 1, 2, \cdots, pop\_size. \tag{4.39}$$

We mention that $i = 1$ means the best individual, $i = pop\_size$ the worst individual.

Selection Process

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select a single chromosome for a new population in the following way:

Step 1. Calculate the cumulative probability q_i for each chromosome V_i,

$$\begin{aligned} q_0 &= 0, \\ q_i &= \sum_{j=1}^{i} eval(V_j), \quad i = 1, 2, \cdots, pop\_size. \end{aligned} \tag{4.40}$$

Step 2. Generate a random real number r in $(0, q_{pop\_size}]$.

Step 3. Select the ith chromosome V_i ($1 \le i \le pop\_size$) such that $q_{i-1} < r \le q_i$.

Step 4. Repeat the second and third steps for *pop_size* times and obtain *pop_size* copies of chromosomes.

Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. In order to determine the parents for a crossover operation, let us repeat the following process from $i = 1$ to *pop_size*: Generate a random real number r from the interval $[0, 1]$, then the chromosome V_i is selected as a parent if $r < P_c$.

We denote the selected parents as $V_1', V_2', V_3', \cdots$ and split them into the following pairs:

$$(V_1', V_2'), \quad (V_3', V_4'), \quad (V_5', V_6'), \quad \cdots$$

Let us illustrate the crossover operation on each pair by (V_1', V_2'). We denote

$$V_1' = \left(y_1^{(1)}, y_2^{(1)}, \cdots, y_{n(n-1)/2}^{(1)}\right), \qquad V_2' = \left(y_1^{(2)}, y_2^{(2)}, \cdots, y_{n(n-1)/2}^{(2)}\right).$$

First, we randomly generate two crossover positions n_1 and n_2 between 1 and $n(n-1)/2$ such that $n_1 < n_2$, and exchange the genes of V_1' and V_2' between n_1 and n_2, thus produce two children by the crossover operation as follows,

$$\begin{aligned} V_1'' &= \left(y_1^{(1)}, \cdots, y_{n_1-1}^{(1)}, y_{n_1}^{(2)}, \cdots, y_{n_2}^{(2)}, y_{n_2+1}^{(1)}, \cdots, y_{n(n-1)/2}^{(1)}\right), \\ V_2'' &= \left(y_1^{(2)}, \cdots, y_{n_1-1}^{(2)}, y_{n_1}^{(1)}, \cdots, y_{n_2}^{(1)}, y_{n_2+1}^{(2)}, \cdots, y_{n(n-1)/2}^{(2)}\right). \end{aligned}$$

We note that the two children are not necessarily feasible, thus we must check the feasibility of each child and replace the parents with the feasible children.

Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. Similarly with the process of selecting parents for a crossover operation, we repeat the following steps from $i = 1$ to *pop_size*: Generate a random real number r from the interval $[0, 1]$, then the chromosome V_i is selected as a parent for mutation if $r < P_m$.

For each selected parent, denoted by $V = (y_1, y_2, \cdots, y_{n(n-1)/2})$, we mutate it in the following way. We randomly generate two mutation positions n_1 and n_2 between 1 and $n(n-1)/2$ such that $n_1 < n_2$, and regenerate the sequence $\{y_{n_1}, y_{n_1+1}, \cdots, y_{n_2}\}$ at random from $\{0, 1\}$ to form a new sequence $\{y'_{n_1}, y'_{n_1+1}, \cdots, y'_{n_2}\}$. We thus obtain a new chromosome

$$V' = (y_1, \cdots, y_{n_1-1}, y'_{n_1}, \cdots, y'_{n_2}, y_{n_2+1}, \cdots, y_{n(n-1)/2}).$$

Finally, we replace the parent V with the offspring V' if it is feasible. If it is not feasible, we repeat the above process until a feasible chromosome V' is obtained.

Genetic Algorithm Procedure

Following selection, crossover and mutation, the new population is ready for its next evaluation. The genetic algorithm will terminate after a given number of cyclic repetitions of the above steps. We now summarize the genetic algorithm for solving the topological optimization models for the communication network reliability as follows.

Input parameters: pop_size, P_c, P_m;
Initialize pop_size chromosomes with the Initialization Process;
REPEAT
Update chromosomes by crossover and mutation operators;

Compute the evaluation function for all chromosomes;
Select chromosomes by the sampling mechanism;
UNTIL(*termination_condition*)
Report the best chromosome as the optimal link topology.

4.4.4 Illustrative Examples

The computer code for the stochastic simulation-based genetic algorithm to topological optimization models has been written in C language. In order to illustrate the effectiveness of genetic algorithm, a lot of numerical experiments have been done and the result is successful. Here we give two numerical examples performed on a personal computer with the following parameters: the population size is 30, the probability of crossover P_c is 0.3, the probability of mutation P_m is 0.2, the parameter a in the rank-based evaluation function is 0.05. Each simulation in the evolution process will be performed 2000 cycles.

Example 1. Let us consider a 10-node, fully-connected network. Suppose that the cost matrix is

Nodes	1	2	3	4	5	6	7	8	9	10
1	-									
2	30	-								
3	43	26	-							
4	45	76	38	-						
5	50	45	17	35	-					
6	62	25	30	28	15	-				
7	25	46	30	16	25	38	-			
8	15	45	13	20	37	40	36	-		
9	51	15	45	10	34	10	46	42	-	
10	45	25	45	15	37	40	16	24	45	-

We suppose that the total capital available is 250. Thus we have a constraint,

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \le 250.$$

We also suppose that the reliabilities of all links are 0.9.

We may set the following target levels and priority structure:

Priority 1: For the subset of nodes $\mathcal{K}_1 = (1, 3, 6, 7)$, the reliability level $R(\mathcal{K}_1, \boldsymbol{x})$ should achieve 99%,

$$R(\mathcal{K}_1, \boldsymbol{x}) + d_1^- - d_1^+ = 99\%$$

where d_1^- will be minimized.

Priority 2: For the subset of nodes $\mathcal{K}_2 = (2, 4, 5, 9)$, the reliability level $R(\mathcal{K}_2, \boldsymbol{x})$ should achieve 95%,

$$R(\mathcal{K}_2, \boldsymbol{x}) + d_2^- - d_2^+ = 95\%$$

where d_2^- will be minimized.

Priority 3: For the subset of nodes $\mathcal{K}_3 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$, the reliability level $R(\mathcal{K}_3, \boldsymbol{x})$ (here the overall reliability) should achieve 90%,

$$R(\mathcal{K}_3, \boldsymbol{x}) + d_3^- - d_3^+ = 90\%$$

where d_3^- will be minimized.

A run of stochastic simulation-based genetic algorithm with 100 generations shows that the optimal link topology is

$$\boldsymbol{x}^* = \begin{pmatrix} - & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ & - & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ & & - & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ & & & - & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & - & 1 & 1 & 0 & 0 & 0 \\ & & & & & - & 0 & 0 & 1 & 0 \\ & & & & & & - & 0 & 0 & 1 \\ & & & & & & & - & 0 & 0 \\ & & & & & & & & - & 0 \\ & & & & & & & & & - \end{pmatrix}$$

whose reliability levels are

$$R(\mathcal{K}_1, \boldsymbol{x}^*) = 0.991, \quad R(\mathcal{K}_2, \boldsymbol{x}^*) = 0.956, \quad R(\mathcal{K}_3, \boldsymbol{x}^*) = 0.938,$$

and the total cost is 242.

If the total capital available is 210, then the optimal link topology obtained by the stochastic simulation-based genetic algorithm with 600 generations is

$$\boldsymbol{x}^* = \begin{pmatrix} - & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & - & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ & & - & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ & & & - & 0 & 0 & 1 & 1 & 1 & 1 \\ & & & & - & 1 & 0 & 0 & 0 & 0 \\ & & & & & - & 0 & 0 & 1 & 0 \\ & & & & & & - & 0 & 0 & 0 \\ & & & & & & & - & 0 & 0 \\ & & & & & & & & - & 0 \\ & & & & & & & & & - \end{pmatrix}$$

which can satisfy the first goal, but the deviations of the second and third goals are 0.08 and 0.15, respectively. In fact, the reliability levels are

$$R(\mathcal{K}_1, \boldsymbol{x}^*) = 0.99, \quad R(\mathcal{K}_2, \boldsymbol{x}^*) = 0.87, \quad R(\mathcal{K}_3, \boldsymbol{x}^*) = 0.75.$$

and the total cost is 207.

Example 2. Now we consider a 20-node, fully-connected network. Suppose that the cost matrix is

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	-																			
2	30	-																		
3	43	26	-																	
4	45	76	38	-																
5	50	45	17	35	-															
6	62	25	30	28	15	-														
7	25	46	30	16	25	38	-													
8	15	45	13	20	37	40	36	-												
9	51	15	45	10	34	10	46	42	-											
10	45	25	45	15	37	40	16	24	45	-										
11	10	35	35	35	16	30	17	35	33	31	-									
12	55	35	35	30	35	37	15	38	35	24	15	-								
13	10	40	10	40	15	34	35	10	47	45	35	35	-							
14	50	19	40	65	45	30	35	35	10	42	30	37	40	-						
15	45	16	40	10	45	37	10	35	35	45	30	40	40	25	-					
16	15	45	15	47	20	30	45	35	23	45	36	35	15	45	45	-				
17	30	40	25	48	20	25	36	15	25	49	10	25	25	37	35	25	-			
18	50	10	45	10	50	30	15	35	40	15	40	30	40	18	15	40	40	-		
19	30	40	25	43	20	25	35	25	25	46	10	25	27	35	35	25	55	40	-	
20	25	40	25	45	25	30	47	10	25	45	10	38	20	43	40	42	10	45	18	-

We suppose that the total capital available is 600, thus we have the following constraint,

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \leq 600.$$

We also suppose that the reliabilities of all links are 0.9.

We may set the following target levels and priority structure:

Priority 1: For the subset of nodes $\mathcal{K}_1 = (3, 6, 7, 13, 17, 19)$, the reliability level $R(\mathcal{K}_1, \boldsymbol{x})$ should achieve 99%,

$$R(\mathcal{K}_1, \boldsymbol{x}) + d_1^- - d_1^+ = 99\%$$

where d_1^- will be minimized.

Priority 2: For the subset of nodes $\mathcal{K}_2 = (1, 2, 4, 5, 9, 11, 12, 14, 15, 18, 20)$, the reliability level $R(\mathcal{K}_2, \boldsymbol{x})$ should achieve 96%,

$$R(\mathcal{K}_2, \boldsymbol{x}) + d_2^- - d_2^+ = 96\%$$

where d_2^- will be minimized.

Priority 3: For the set of all nodes $\mathcal{K}_3 \equiv \mathcal{V}$, the reliability level $R(\mathcal{K}_3, \boldsymbol{x})$ (here the overall reliability) should achieve 95%,

$$R(\mathcal{K}_3, \boldsymbol{x}) + d_3^- - d_3^+ = 95\%$$

where d_3^- will be minimized.

A run of stochastic simulation-based genetic algorithm with 300 generations shows that the optimal link topology is

$$
\boldsymbol{x}^* = \left(\begin{array}{cccccccccccccccccccc}
- & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 & & - & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
 & & & - & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
 & & & & - & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & & & & & - & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & & & & & & & & - & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
 & & & & & & & & & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & & & & & & & & & & - & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
 & & & & & & & & & & & - & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & - & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & - & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & & - & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & & & - & 0 & 0 & 1 \\
 & & & & & & & & & & & & & & & & & - & 0 & 0 \\
 & & & & & & & & & & & & & & & & & & - & 0 \\
 & & & & & & & & & & & & & & & & & & & -
\end{array}\right)
$$

which can satisfy the three goals. Furthermore, the reliability levels are

$$R(\mathcal{K}_1, \boldsymbol{x}^*) = 0.993, \quad R(\mathcal{K}_2, \boldsymbol{x}^*) = 0.971, \quad R(\mathcal{K}_3, \boldsymbol{x}^*) = 0.953,$$

and the total cost is 594.

Chapter 5

Set Covering Problem and Genetic Algorithm

About 30 to 20 years ago, there were some eager yet hard research activities in solving Set Covering problems and/or Set Partitioning problems. We can see it well in C.E.Lemke, H.M.Salkin and K.Spielberg[138], H.M.Salkin and R.Koncal[175, 176, 177], K.Iwamura[84, 85, 86, 88], R.S.Garfinkel and G.L.Nemhauser[45][46], H.Konno and H.Suzuki[116],H.M.Salkin[174] and so on. They used either cutting plane algorithm and /or branch and bound algorithm and then found that these algorithms showed exponential and heavy data dependent computing time, see K.Iwamura,Y.Deguchi and N.Okada[95], K.Iwamura and N.Okada[105], H.Konno and H.Suzuki[116], G.L.Nemhauser and L.A.Wolsey[165]. This fact coincides with the theoretical results of NP-completeness in M.R.Garey and D.S.Johnson[44]. In recent years, we saw some advancements in solving NP-complete problems in K.Tagawa, D.Okada, Y.Kanzaki, K.Inoue and H.Haneda[191], and J.Xie and W.Xing[200]. They showed and hinted that for NP-complete problems, genetic algorithm might find a fairly good solutions. So, we thought that genetic algorithm with*Domain Specific Knowledge* might enhance its computing efficiency , finding a good solution at an early stage of computing process, keeping its computing time stable/ controllable by setting the maximum generation number at some reasonable value. Yes, about 20 years ago in Japan, there was a saying such that when an integer programming problem could not be solved in a suitable amount of time, then it wouldn't be solved even if we used ten times greater amount of computing time. Still, Set Covering and/or Set Partitioning problems have a wide range of applications, there should be some efforts to find a more smart algorithm to solve some *medium sized* Set Covering and/or Set Partitioning problems. So here, we carried out *designing a genetic algorithm to solve medium sized Set Covering problems using Domain Specific Knowledge* with computational experiences, where *Domain Specific Knowledge* means any kinds of knowledge which we can get from input data information we have to solve.

As for the small sized ones, any algorithms including *enumeration type algorithms* would be sufficient. For the big sized problems having more than 200 thousands columns, heuristic algorithms or artificial intelligence type algorithms or some other technologies might be asked for, although exact problem size would be affected by what kind of computers we could use. In the next section, we will explain how we have designed our code to solve the medium sized Set Covering problems.

5.1 Definitions and Domain Specific Knowledge

Let m, n be natural numbers, c_j be positive integers, i.e., *costs*, $(1 \leq j \leq n)$ and Let a_{ij} be 0 or 1 for $1 \leq i \leq m, 1 \leq j \leq n$.

The following Integer Programming Problem,
minimize

$$cx = \sum_{j=1}^{n} c_j x_j \tag{5.1}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad (1 \leq i \leq m) \tag{5.2}$$

$$x_j \in \{0, 1\} \tag{5.3}$$

is called *Set Covering problem*.

For this Set Covering problem , we first see that

Theorem5. 1 *Any basic feasible solution x_j of the Linear Programming Problem derived from the Set Covering problem satisfies*

$$0 \leq x_j \leq 1 \qquad for \quad 1 \leq j \leq n \tag{5.4}$$

Proof See R.S.Garfinkel and G.L.Nemhauser[46], H.M.Salkin[174], H.Konno & H.Suzuki[116].

We call the linear programming problem (5.1),(5.2),(5.4) an *LP(linear programming) relaxed Set Covering problem* and/or *LP problem derived from the Set Covering problem*. Based on this , we propose **Domain Specific Knowledge 1** for the input data which has n/m greater than two or more, where $m =$ the number of rows, $n =$ the number of the columns. For such input data at least $n - m$ column variables take value zeros in an optimal basic feasible solution for the LP relaxed Set Covering problem. Hence modifying the LP optimal basic feasible solution to create good zero-one solutions having nearly equal objective values as itself would provide us some good chromosomes in the first starting generation in our genetic algorithm. Set the variables be zeros or ones as they are zeros or ones in the LP relaxed optimal feasible solutions. If x_j has a fractional value between zero and one, round up or round down to get a good feasible zero- one solution. Find any kind of smart heuristics to get enough good feasible zero-one solutions from the LP relaxed optimal solution and/or solutions.

We call a feasible solution of (5.1),(5.2),(5.3) a *cover solution* ,or a *cover* in short. A cover $x = (x_j)$ is a *prime* cover if there exists no $y = (y_j)$ satisfying (5.2),(5.3)
and

$$y_j \leq x_j, \qquad y \neq x \tag{5.5}$$

We easily see that the optimal solution of (5.1),(5.2),(5.3) can be found in the set of prime covers. So,to solve the Set Covering problem, we only have to take the prime cover with minimum cx value. Therefore, solving (5.1),(5.2),(5.3) is mathematically equal to finding all the prime covers. Yet, finding all the prime covers are hard enough to solve the Set Covering problem itself. So, we do not go in this way. M.Fushimi[42] used this , **Domain Specific Knowledge 2**, to get a good prime cover after sorting the columns with respect to cost performance. His method at present gives us just two good prime covers.

Theorem5. 2 *Any prime cover can be expressed as a basic feasible solution of the linear programming problem obtained*
by relaxing

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

into

$$0 \leq x_j \leq 1$$

This theorem tells us that an optimal cover(solution) exists in the basic feasible solutions of the LP relaxed Set Covering problem. But, an optimal basic feasible solution of the LP relaxed Set Covering problem might not be zero-one. If it is zero-one, then we are done, good luck! In case we have no good lucks, we propose, **Domain Specific Knowledge 3**, to go like F.S.Hillier[60]. We search for some adjacent integer vertices and put them in the first generation of our genetic algorithm.

Theorem5. 3 *Let x be any feasible solution of the relaxed linear programming problem derived from (5.1),(5.2),(5.3) and set $y_j =< x_j >$, where $< t >$ takes the smallest integer greater than or equal to t . Then we have a cover solution y of the Set Covering problem.*

Proof $y_j \in \{0, 1\}$ and so $y = (y_j)$ satisfies (5.3) because $0 \leq x_j \leq 1$. Furthermore we have

$$\sum_{j=1}^{n} a_{ij} y_j \geq \sum_{j=1}^{n} a_{ij} x_j \geq 1. \qquad \square$$

After solving Linear Programming problem derived from the Set Covering problem, we try to get as many optimal basic feasible solutions as possible based on the optimal basic feasible LP tableau. If we find an all 0,1 among them, then we are done. Good luck! In case all of them are fractional, using this theorem, we get 0,1 feasible solutions for the Set Covering problem. Transform them into prime cover solutions using the method of Fushimi[42] or any tricks. These would provide us enough good covers to start in the first generation of our genetic algorithm. We call it **Domain Specific Knowledge 4**.

Last but not least, taking probabilistic/statistical characteristics of genetic algorithms, we propose **Domain Specific Knowledge 5** as follows. Re-index the columns after cost -performances cp_j of the column j,where

$$cp_j = \frac{c_j}{\sum_{i=1}^{m} a_{ij}}(1 \leq j \leq n) \tag{5.6}$$

, and just apply simple genetic algorithm in harmony with one point cross over operation. This is just like J.F.Pierce and J.S.Lasky[170], but differs both in its details and in its usages.

5.2 Handling Bitwise Operation and Storing Coefficient Matrix Bitwise

We have stored zero-one information of each column in bitwise. We used

$$< \frac{m}{32} > \tag{5.7}$$

words to store column wise zero-one information for each column in an array form, i.e., in array G. So, for an m by n coefficient matrix, we needs

$$n \times < \frac{m}{32} > \tag{5.8}$$

words to store zero-one coefficient matrix information in G. To speed up feasibility check for each chromosome, we used another array A. For an input data with 640 rows $\times$ 2000 columns, we needed about 360Kbyte memory in all and so all the crucial computations have been done in core. We made full use of bitwise AND, OR, EXCLUSIVE OR operations in our C++ programs. These bitwise operations co-worked well with our Genetic Algorithm.

5.3 A Genetic Algorithm

In this section we design a genetic algorithm to solve the Set Covering problem. We will discuss the initialization process, evaluation function, selection, crossover and mutation operations in turn.

5.3.1 Initialization Process

As usual in Genetic Algorithm, we generate *pop_size* initial feasible chromosomes(solutions). Randomly generate integers between 1 and n, say, j_1 and set $x_{j_1} = 1$. Continue randomly generating integers between 1 and n until $(x_{j_1} = x_{j_2} =, \cdots, x_{j_k} = 1)$ constitute a feasible solution, where k is the smallest integer such that $(x_{j_1} = x_{j_2} =, \cdots, x_{j_k} = 1)$ is a feasible solution. In this way, randomly generate feasible *pop_size* chromosomes(solutions) in total. Let us call them V_i, $(i = 1, 2, \cdots, pop\_size)$.

5.3.2 Evaluation Function

For each chromosome in the population, we calculate its objective function and set its fitness to be the inverse of the objective function, because the objective function of any chromosome is always positive.

5.3.3 Selection Operation

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select a single chromosome for a new population in the following way:

Step 1. Calculate a cumulative probability a_i for each chromosome V_i, $(i = 1, 2, \cdots, pop\_size)$,where

$$a_i = \sum_{j=1}^{i} p_j, \quad p_i = \frac{f_i}{\sum_{j=1}^{pop\_size} f_j} \tag{5.9}$$

and f_i =the inverse value of the fitness of the chromosome V_i, $(i = 1, 2, \cdots, pop\_size)$.

Step 2. Generate a random real number r in $[0, 1]$.

Step 3. If $r \leq a_1$, then select the first chromosome V_1; otherwise select the i-th chromosome V_i $(2 \leq i \leq pop\_size)$ such that $a_{i-1} < r \leq a_i$.

Step 4. Repeat Steps 2 and 3 *pop_size* times and obtain *pop_size* copies of chromosomes.

In this process, the best chromosomes get more copies, the average stay even, and the worst die off.

5.3.4 Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. This probability gives us the expected number $P_c \cdot pop\_size$ of chromosomes which undergo the crossover operation.

Firstly we generate a random real number r in $[0, 1]$; secondly, we select the given chromosome for crossover if $r < P_c$. Repeat this operation *pop_size* times and produce $P_c \cdot pop\_size$ parents, averagely. For each pair of parents (vectors $V_1 = (x_1, x_2, \cdots, x_n)$ and $V_2 = (y_1, y_2, \cdots, y_n)$), the crossover operator on V_1 and V_2 will produce two children as

$$V_1' = (x_1, \cdots, x_s, y_{s+1}, \cdots, y_n), V_2' = (y_1, \cdots, y_s, x_{s+1}, \cdots, x_n)$$

, where s is randomly generated integer between 1 and $n-1$. If the two children are feasible, then select the best two of the four. If either of the children is infeasible, then correct it feasible by randomly adding value one variable step by step. Then select the best two of the four into the population. If none of the children are feasible, do the same as stated.

5.3.5 Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. This probability gives us the expected number $P_m \cdot pop\_size$ of chromosomes which undergo the mutation operation.

Generating a random real number r in $[0, 1]$, we select the given chromosome for mutation if $r < P_m$. Let a parent for mutation, denoted by a vector $V = (x_1, x_2, \cdots, x_n)$, be selected. Assume that $\{j_1, j_2, \cdots, j_z\}$ is a randomly generated subset of $\{1, 2, \cdots, n\}$. Then set $x'_{j_t} = 1 - x_{j_t} (1 \leq t \leq z)$ to get a new chromosome $V' = (x'_1, x'_2, \cdots, x'_n)$ except for the fact that all $x'_j = x_j$ are predetermined for all $j \notin \{j_1, j_2, \cdots, j_z\}$. If the chromosome V' is infeasible, then modify it to a feasible one V'' using the same method as in Crossover Operation. Repeat this operation *pop_size* times.

Following selection, crossover and mutation, the new population is ready for its next evaluation. The algorithm will terminate after a given number of cyclic repetitions of the above steps. The proposed genetic algorithm is shown as follows:

Procedure Genetic Algorithm

Input parameters;

Initialize the solutions (chromosomes);

REPEAT

Update the chromosomes by genetic operators;

Compute fitness of each chromosome by objective function;

Select the chromosomes by spinning the roulette wheel;

UNTIL(*termination_condition*)

5.4 Computational Results

Set Covering problem is an NP-complete problem. Computational experiments have shown that its computing time is heavily input data dependent and explodes as input problem data size goes up. So, we have applied our genetic algorithm with **Domain Specific Knowledge 5** to input problem data A,C,G of H.Morohoshi and M.Fushimi[159], which are Real-World input data. We have got chromosomes with fitness(objective function) value 13,13,4 for A,C,G , which are better or equal to 14,21,4 that have been obtained by applying the algorithms of M.Fushimi[42]. We have made our program in C++ on Celeron 400 MHz with incore memory size 96MB. Its computing time is just 1 to 2 seconds, while our rudimentary enumeration type algorithm took 10 to 15 minutes.

To see how our **Domain Specific Knowledge** affects the performance of the original genetic algorithm, we have made additional two different Initialization Processes. First one is as follows: We randomly generated 15 bits three integers and then concatenated them into one chromosome. We repeated this process until we got *pop_size* starting chromosomes. This way of getting the chromosomes for the starting generation in our genetic algorithm is named **Method 1**. In this paper throughout, we always uses **Domain Specific Knowledge 5**. Second one is just the original Initialization Process in our genetic algorithm which we call **Method 2**. The third one , **Method 3**, can be fully illustrated as follows:

$$
\begin{array}{rcccccc}
 & & x_1 & x_2 & x_3 & x_4 & x_5 \\
 & & 1 & 1 & 0 & 1 & 0 \\
 & & 0 & 1 & 0 & 0 & 1 \\
a_{ij} & = & 1 & 1 & 0 & 1 & 0 \\
 & & 0 & 0 & 1 & 0 & 1 \\
 & & 0 & 1 & 1 & 0 & 0 \\
 & & 1 & 0 & 0 & 1 & 0 \\
 & & & & & & \\
h_j & = & 3 & 4 & 2 & 3 & 2 \\
c_j & = & 2 & 3 & 2 & 4 & 4 \\
cp_j & = & 0.66 & 0.75 & 1.00 & 1.33 & 2.00
\end{array}
\tag{5.10}
$$

First, select the most cost-effective x_j in the table, so set $x_1 = 1$.From top to bottom, search for an uncovered row. The 2-nd row is uncovered and so select the most cost-effective column not picked up yet. So, we select the 2-nd column letting $x_2 = 1$. Combining the column x_1 and the column x_2 ,we see that the 4-th row is uncovered and so select the most cost-effective column not yet picked up. So, we select the 3-rd column setting $x_3 = 1$. We have got one feasible chromosome(solution) with $x_1 = x_2 = x_3 = 1, x_j = 0$ otherwise . Second select the next cost-effective column x_2 and set $x_2 = 1$. The first uncovered row is the 4-th row. Then search for a new column from left to right which first covers the 4-th row. Then we get the column x_3 with $x_3 = 1$. Combining $x_2 = 1$ and $x_3 = 1$, we see that the last row is not covered and so finally we get x_4. Setting $x_4 = 1$, we get another feasible chromosome with $x_2 = x_3 = x_4 = 1, x_j = 0$ otherwise.Third we get the third feasible chromosome $x_3 = x_1 = x_2 = 1, x_j = 0$ otherwise, which is identical to the first feasible chromosome. So, we drop it. But the 5_th feasible chromosome turns out to be $x_5 = x_1 = x_2 = 1, x_j = 0$ otherwise, which is a new one.

In this way we get *pop_size* feasible chromosomes with cost effective columns, while maintaining statistical diversity of the starting population.

We have randomly generated Set Covering input data with unicost, density about 10%, size 200×500. We have applied our Genetic Algorithm(**Method 2**) to this input data , where randomly set first GA parameters are as follows;

- Population size = 50,
- Cross over probability = 0.25,
- Mutation probability = 0.01,
- Final generation number = 1000

and its computational result is

- Best feasible fitness function obtained so far = 26
- Computing time 8 seconds(using floppy disk drive) .

We can see how these **Domain Specific Knowledge** affects the performance in Figure 1, where the vertical axis is log measured. We can judge that **Method 1** is the worst , **Method 2** the second best, **Method 3** the best. We observe that the finer **Domain Specific Knowledge** becomes, the better final feasible function values we can get, which is what we have anticipated in advance.

We have tested **Method 2**, **Method 3** for a randomly generated Set Covering input data with unicost, density about 10% , size 640×2000 . The result in Figure 2 shows that **Method 3** is better than **Method 2** , once again for this input data. The parameter values are the same except mutation probability , which is set to 0.001 . Computing time has been just 20 seconds with final objective function value 27, which is surprisingly small if one compares it with that of some branch and bound type algorithms.

We have also carried out computational experiments to see how our Genetic Algorithm with **Method 3** works as the input problem size goes up. Below are the results, where each input problem data were randomly generated with density nearly 3% and unicost objective coefficients. And still we have set Population size=50, Crossover probability = 0.25, Mutation probability = 0.001, Final generation number = 1000.

Table 5.1: Computing Time when m Varies

Problem size	Computing time
200 X 2000	15 seconds
300 X 2000	16 seconds
400 X 2000	17 seconds
500 X 2000	17 seconds
640 X 2000	18 seconds

Judging from this table, we can say that Computing time of our Genetic Algorithm varies linearly in m = the number of rows in the input problem data.

We have done the same computational experiments as above with just one change. This time we have kept the number of rows at 640 and varied the number of the columns n. Just below are the results.

Here, we can say that Computing time is linear up to problem size 640 $\times$ 2000. Concluding the two , we can say that our Genetic Algorithm works linearly in input problem data size.

We have carried out additional computational experiments to see parameter dependency of our Genetic Algorithm. We have randomly generated size 640 $\times$ 2000 , uni-cost , density 3% set covering input data, which we call

Table 5.2: Computing Time when n Varies

Problem size	Computing time
640 X 500	9 seconds
640 X 1000	13 seconds
640 X 1500	16 seconds
640 X 2000	18 seconds

2000N16C. This time we used NEC PC note PC-LM40H32D6 Celeron 400, which is about 15% slower than the former one. Computational results appear in Table 5.3, where * denotes that we have tried 5 trials for the input data 2000N16C.Mean computing time and mean objective function values without * denotes that we have tried 10 trials. The reason we halved the trials was the fact that fluctuations in computing time was very small. For the most fluctuated one with $N = 800$, its computing time were 381,383,398,396,401with fluctuations less than 6%. In the table, N stands for population size in our Genetic Algorithm, p_c: crossover probability, p_m : mutation probability . Comparing the first two lines, we see that letting the final generation number double makes our GA 's computing time about two times large with a little bit good objective function values. Comparing the first and the third , we see that changing the values of p_c and p_m doesn't affect our GA's computing time, yet worsens the objective function values. To see how our Genetic Algorithm works when we change N with all other parameters fixed, i.e., $p_c = 0.25, p_m = 0.001$, final generation number= 1000, we have got the results from the fourth line to the last line. From these results we can say that computing time is proportional to N and its objective function values improving a little bit.

Table 5.3: Parameter Dependency of our Genetic Algorithm

N	p_c	p_m	Final generation number	Objective function value			Mean computing time
				Best	Worst	Mean	
50	0.25	0.001	1000	64	67	65.4	23.6 sec
50	0.25	0.001	2000	64	66	64.2*	*46.2 sec
50	0.50	0.100	1000	77	80	78.7	21.5 sec
25	0.25	0.001	1000	66	70	67.2	11.6 sec
75	0.25	0.001	1000	63	67	65.2	34.7 sec
100	0.25	0.001	1000	63	67	64.9	46.2 sec
200	0.25	0.001	1000	61	66	63.4*	*93.2 sec
400	0.25	0.001	1000	63	66	64.0*	*196.0 sec
800	0.25	0.001	1000	62	65	63.4*	*391.8 sec

Finally, we would like to summarize our computational results. In addition to small sized input data,we have carried out two randomly generated Set Covering data of size 200 X 500, 640 X 2000 and have got very good near optimal solutions to each of them. We have tried up to ten input parameters for each Set Covering data. Ten is a small one because 640 X 2000 input data uses just 20 seconds for one run of our genetic code with final generation number 1000. We have shown how the performance of our genetic algorithm improves as we take more care of creating first starting population. Applying **Method 3** for 200 X 500 input data, we have got a 0-1 feasible solution with its objective function value 19 trying up to ten input parameters.We have applied LINGO version 3 to the 200 X 500 Set Covering data using Celeron 400 to find that it was unable to finish its computing in two weeks. We thought that it would take another two weeks or more. Considering the fact that there exist some Set Covering data for which there surely exists an optimal 0-1 feasible solution, yet Branch and Bound type algorithm such as LINGO, LINDO and MPS/X cannot find even a feasible integer solution in a reasonable amount of time, we think our results would be useful for decision maker in the real world.

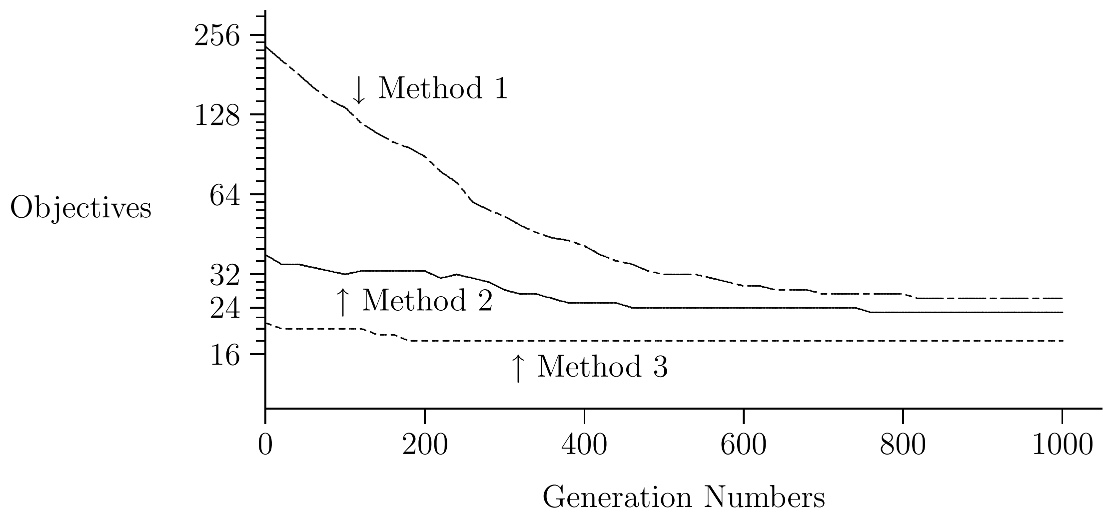

Figure 5.1: Computational Result 200 × 500

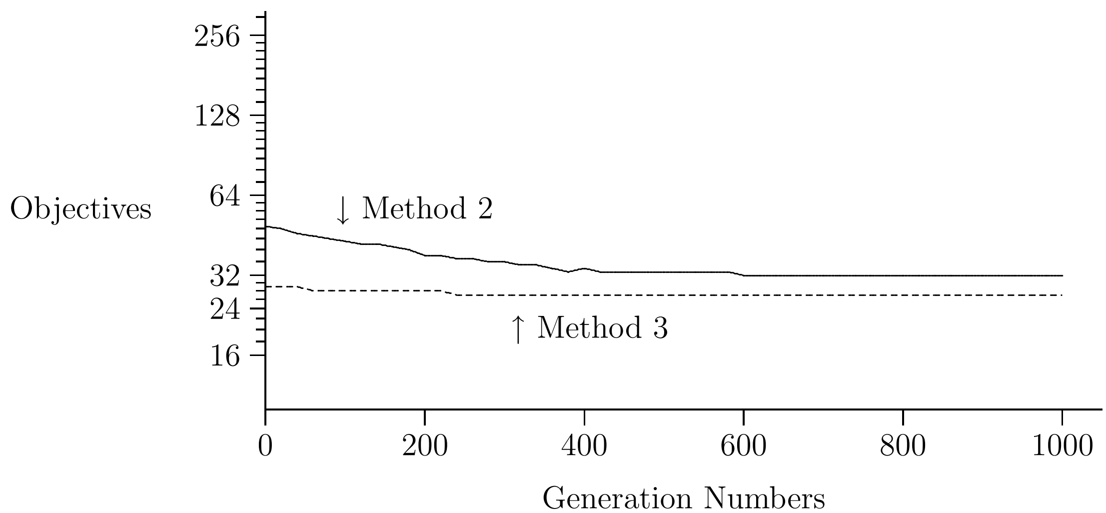

Figure 5.2: Computational Result 640 × 2000

Chapter 6

Conclusion

6.1 Discussion and Conclusion

In this final Chapter, I would like to overview Planning/Decision-Making Problems presented in the preceding Chapters.

First, Knapsack problem. Already in 1970s it was well recognized that Knapsack problem was easily solved to real problem data. We imagined that the situation was the same for knapsack typed integer programming problems(K.Iwamura [83](1972)). Its easiness contributed almost half of success of H.Mukawa, J.Sensui, K.Iwamura and J.Kase[161](1971). Another half of it was because *Capacitated Facilities Location Programming Problem* contained a transportation problem as a subproblem. To solve a transportation problem is much faster than to solve a general linear programming problem.

We find that S.Martello and P.Toth's book ,*Knapsack Problem*, published in 1990, is a fine one. They treat almost all types of knapsack problems. But they don't treat our Knapsack Typed integer programming. They equip the book with some FORTRAN programs to solve some kinds of Knapsack problems, although we can't see whether personal computers other than IBM PCs can read the programs. Using the terminology of S.Martello and P.Toth, Knapsack Typed integer programming is called *Unbounded Multiple Knapsack Problem*. Finally it is author's duty to ask the readers to take notice that they say that they have some input data of some knapsack typed problems which they are not able to solve successfully. E.L. Lawler[1979] treated *Approximation Algorithms for Knapsack Problems*. See also E.Horowitz and S.Sahni[1978], K.Iwamura[1981], and R.Weismantel[1992]. We noticed the existence of Weismantel's work through Internet. But we were not able to get the full content of his work. We hope we will get it in the near future.

As for the research activities carried out by other Japanese, I think that H.Suzuki and K.Iwamura[190](1979), H.Konno and H.Suzuki[116](1982) are still useful. In 1995, M.Futakawa et al.[43](Max-Min Knapsack Problem) reported that their problem was able to be solved very quickly through their heuristic algorithm and its error ratio got lesser and lesser down to 0.001 percent as the problem size increased up to ten thousand. This fact was pointed out already in 1972 by J.G.Lührs[152] for a simple Knapsack Problem. In 1970s researchers thought that knapsack problem was an easy one to devote ourselves in. In fact, they are still fruitful as the piles of the computational results in S.Martello and P.Toth[154](1990) show it to us. See also T.-C. Lai, M.L.Brandeau and S.Chiu[134](1994) and Y.Hayashi[58](1995). Today, we can say that we will find an optimal / a near optimal solution for a Real-World data of Knapsack Problems by the Branch and Bound method within a reasonable amount of time.

When we turn our attention to the Set-Covering and/or Set-Partitioning problems, we realize they are far beyond our ability, particularly finding an optimal solution for a real-world data of the Set Covering/Set-Partitioning problems. Therefore the author thinks it's beneficial to state an algorithmic history of discrete optimization and integer programming.

The importance to solve the integer programming problems by computers were well acknowledged by operations researchers late 1950s (G.B. Dantzig[24] (1963)). R.E.Gomory's "Outline of an Algorithm for Integer Solutions to Linear Programs" appeared in Bull. Amer. Math. Soc. in 1958. Gomory developed two algorithms to solve general linear integer programming problem. They are called "Fractional integer programming algorithm" and " All integer algorithm"(T.C.Hu[66](1970)). In R.S.Garfinkel and G.L.Nemhauser[46](1972), they are called " the method of integer forms " and " dual all integer algorithm". In their book are showed Young's primal all-integer algorithm, too. These algorithms are all based on cutting plane methods because they add a new cutting plane constraint from iteration to iteration. In the early 70s, the author got a rumor that a cutting plane method generated so many cut constraints that the algorithms based on it wasn't able to go further because of both storage and computing time limit. And so, practitioners thought that a general algorithm to solve general integer linear programming problem was useless and helpless. The rumor was in fact a truth. We were able to see it at page 380 in Garfinkel and Nemhauser[46](1972), although there were some problems and some problem instances that could be solved within a reasonable time bound.

Then came a wave of research activities in integer programming(IP) by the Branch and Bound method. This time, we picked up a specific integer programming problem such as Knapsack Problem, Travelling Salesman Problem, Set Covering Problem, Vehicle Routing Problem, Set Partitioning Problem and so on. We carried out computational experiments of the Set

Partitioning Problem with three algorithms. The first one is from E.Balas and M.W.Padberg[8](1972),[9](1975), the second one is from R.S.Garfinkel and G.L.Nemhauser[45](1969), the third one is from J.F.Pierce and J.S.Lasky[170] (1973). These were selected on the basis of their algorithmic efficiency , independently of implementation easiness of their algorithms. Furthermore we gave every improvement to each of the three with as much programming techniques as we were able to pay. To each problem instance, we applied MPS/X to compare the efficiency of the three on a fare standpoint. Up to problem size 100 rows and 200 columns, densities varying from 68% to 3.4% every algorithm was able to get an optimal solution to all problem instances on FACOM 230-38S except E.Balas and M.W.Padberg's one. But when we tried 200 rows and 2000 columns problem instance, then MPS/X stopped computing because it had used up all the external memory. E.Balas and M.W.Padberg's algorithm was unsuccessful because it demanded too much memory size for its Column Generating Procedure. This drawback was also observed by E.Balas's student Prof. Gerritssen. The most promising algorithm of J.F.Pierce and J.S.Lasky continued computing for more than 70 hours, i.e., more than 7days, each day with 10 hours. Finally we were asked to give up computing by our University Computing Center. So, we re-ran the same problem instance with a new counter in the program to see how many subproblems it created. It was not a million but more than a billion. But why so many? This is , still at present, the computational difficulty every NP-Complete /NP-Hard Problem has. In this case, we easily saw that even the computations like

$$z \leftarrow z + c_j$$

and

$$z \leftarrow z - c_j$$

were meaningless because single or double floating point numbers mechanism could not maintain computational accuracy. Declaring z, c_j real 8 byte isn't safe enough, i.e., one has to keep costs c_j and z all in integers. Even if we had improved the original algorithm so that it should re-calculate objective function value each time it got a new feasible solution , it was still incapable of treating an infeasible input problem data. At the worst we hoped that J.F.Pierce and J.S.Lasky's algorithm would over-perform the others, but in fact , it wasn't true. No algorithm showed such uniform efficiency over the others. We were heavily shocked that we once more got the same result as in the case of Travelling Salesman problem, that is to say,*Exponential computing time and its heavy data dependency.* In 1977, Garfinkel and Nemhauser sent us a letter that they had no source program because they had a business company make an assembler program and so they had no right to send us a source program list. In their paper, they wrote that they made a source program in FORTRAN! But we thought they were sincere, anyway they answered us. N. Christofides[22](1974-75) just sent us his new coming book with no letter, no source program. In these cases, we offered them our source program in exchange for theirs to solve the Set Partitioning Problem. E.Balas also wrote to us to try some Set Partitioning data of low density, from 1.5% down to 1%. Surprisingly enough, in Balas and Padberg' paper "On the set covering problem: II An algorithm for set partitioning", *Operations Research 23(1975)*,74-90, there appeared no sentences that their algorithm was devoted to low density Set Partitioning problem data. We confirmed low density data 3.7% caused density 57% when their algorithm got to the Block Pivoting Procedure for the enlarged table. Devising a branch and bound algorithm enthusiastically to solve a specific IP problem lasted about 15 years. Some researchers are still searching for good Branch and Bound type algorithm to solve some *NP hard* problems for real-world large input data. Recently, M.Shindo and E.Tomita[186](1988), E.Tomita et al.[193](1996) showed the same sort of difficulty to find a *Maximum Clique* with both worst-case time complexity and experimental evaluations.

Through this kind of hardness in integer programming, researchers have come to realize the importance of the work of S.A.Cook[23](1971). He has found the problem classes such as *P problems, NP-hard problems, NP complete problems. NP-hard problems* are also called as *NP problems.* R.M.Karp[110](1972) then quickly found that lots of discrete planning problems in Operations Research fell in *NP complete* ones. I noticed this fact in 1978. As for other notions such as *pseudo-polynomial time algorithm, number problem, NP complete in the strong sense, polynomial time approximation scheme, fully polynomial time approximation scheme*, one can consult M.R.Gary and D.S.Johnson[44](1979). Research subjects in these branches are called *Complexity Theory.* They are still taking efforts to finally solve the *P vs. NP* problem which is an extremely important open problem in both Computer Science and Operations Research.

For almost 25 years up to today, there appeared lots of theoretical papers on the Complexity Theory. As we have just stated in the preceeding, we can see its developments in M.R.Garey and D.S.Johnson[44](1979), A.V.Aho, J.E.Hopcroft and J.D.Ullman[4](1974) and T.Ibaraki[69](1994). Yet, at present the author can not think that 1970's Integer/Combinatorial programming problems are completely analyzed and explained through the Turing Machine based Complexity Theory. But the results of M.Li and P.M.B. Vitanyi[139](1989), Kobayashi[115](1992) coincide with the computational experiences in the Integer/Combinatorial Programming. They say the following;

There is an input data distribution such that for any algorithm to solve a specific NP-Complete problem, its mean time complexity and its worst time complexity are of the same order.

So, there is a room for trying to devise some *Non-Turing Machine based* algorithm for *NP-hard* Integer/Combinatorial Planning Problems. We have carried out one such project in **Chapter 5**.

If we limit ourselves to combinatorial optimization problems which have a fine structure such as matroid and/or greedoid , then there are some problems for which we can find a polynomial time algorithm. Minimum Spanning Tree, Shortest Path, Flows in Network fall all in this class. See, R.S.Garfinkel and G.L.Nemhauser[46](1972), M.Iri[76][77](1969), M.Iri et al.[80](1986).

So are optimization problems on Submodular Polyhedron/System. See also K.Iwamura[92][93](1995), S.Fujishige[37](1984), [39](1991). We point out that the theory of Submodular System can be traced back to the work of M.Iri[79](1984),[78](1979), [76](1969),[75](1968). There appears the notion of principal partitioning , some time apparently, some time veiled and unseen, in the theory of Submodular System. The author would like to say that dual supermodular polyhedron naturally comes out through principal partitioning and submodular system's poset. It's also noteworthy that a primal-dual type theorem independent from LP duality is a useful one in combinatorial optimization. To see how Submodular Polyhedron came from network flow, the reader would be advised to consult M.Iri,S.Fujishige and T.Ooyama[80](1986). For much more developments of submodular functions, one can have a look at K.Murota[162](1995), [163](1996)(convexity), S.Fujishige[39](1991)(network flow) and M.Nakamura[164](1988)(principal partitioning).

The reference book[68](1981) written by T.Ibaraki is one of the best books that treat structural classification of combinatorial optimization problems from Dynamic Programming point of view. Dynamic Programming sometimes leads to a polynomial time algorithm and so it is still important. Particularly sequential decision processes(sdp) is very important. S.Iwamoto[81](1987) and M.Sniedovich[187](1992) will be eye-opening to those who want to know how powerful Dynamic Programming is. One can catch the heart of the theory through T.Ibaraki[67](1973), too. In K.Iwamura[91](1993), it is revealed that greedy algorithm over a given greedoid can be captured within a framework of sequential decision processes. And so, the author would like to restate here that the notion of greedoid is a very wonderful one . Today, we can have its whole view in the book written by B.Korte, L.Lovász and R.Schrader[129](1991).The author believes that greedoidal point of view would let the researchers in combinatorial optimization have a clear and better understanding for their problems at hand with efficient algorithms.

Uncertain Programming treats Planning problems under uncertainty. It has been developed by B.Liu and I. We treat almost all kinds of uncertainty; probabilistic/stochastic, of fuzziness/possibility, of reliability, of accuracy and so on. We think we can have much more achievements in this field(Zhao R., Iwamura K. and B.Liu[204][205],G.Wang and K.Iwamura[197]).

After experiencing robust and efficient behavior of the Genetic Algorithm in Uncertain Programming, we have carefully applied Genetic Algorithm to the Set Covering Problem. We have succeeded in finding all the optimal solutions for 3 input data of Fushimi and Morohoshi(see, Fushimi[42]). These data have about 10 % density. We randomly generated two Set Covering input data density about 10% with row and column size 200×500, 640×2000. Even for 640×2000 input data, our Algorithm have finished 1000 generation computation within 20 seconds through Celeron 400. We have applied LINGO Version 3 for the 200×500 input data. LINGO version 3 has found near optimal value 16 in about 3 minutes and then near optimal value 15 in about 90 minutes. But it continued computing 14 days, getting its lower bound value 11.0985 and so we judged that it would take another 14 days or more because its first LP optimal value was 8.67354. LINGO Version 3 iterated 236,430,233 using 36MB in-core memory. So, there might have happened the same problem as we had reported in Iwamura and Okada[105](1999)(floating point number accuracy problem). Our Genetic Algorithm have found near optimal value 19 after 10 trials of 1000generation computation within 5 minutes. This is just 30% worse than the one LINGO Version 3 had found. There are some Set Covering input data for which Branch and Bound/Cut Algorithm cannot find even a feasible solution within a suitable amount of time. So, our Algorithm will be a nice substitute for such Set Covering input data. Additional computational experiments tell us it is a robust algorithm. Furthermore our genetic algorithm doesn't have floating point accuracy problem at all. In case Branch and Bound type algorithm cannot find a first feasible solution within a suitable amount of time, our genetic algorithm can find a near optimal solution if the input data is feasible. This is a good characteristic of Genetic Algorithm even if quality of near optimal solutions is not guaranteed. Yet, it isn't so bad. Therefore Our Genetic Algorithm with Branch and Bound type Algorithm would be strong means for practitioners who want to solve the Set Covering problem in the real world(K.Iwamura, T.Sibahara, M.Fushimi and H. Morohoshi[106], [107]).

We have carried out a systematic computational study of the algorithm to show its efficiency(N.Okada, K.Iwamura and Y.Deguchi[167]), where we have made a computational comparison between the algorithm and the commercial software code LINGO 4 concerning approximation ratio and computing time. Through it, We can say that although our Genetic Algorithm cannot find an optimal solution of the Set Covering problem, it can find an approximate solution whose values are within about 30% worse of the objective function value LINGO 4 finds. The greater the number of the integer variables of the Set Covering problem, the easier can our GA find a good approximation solution and so, more practical becomes our GA.

We have further investigated input data dependency of our Genetic Algorithm, i.e.,dependency on costs and dependency on density(K.Iwamura, M.Horiike and T. Sibahara[97]). We have found that although our Genetic Algorithm still cannot find an optimal solution of the Set Covering problem, for size 999×999 input data, it can find approximate solutions whose objective function values are, this time, within about 8% worse than the objective function values LINGO 4 finds. We have found that for input problem data with density more than or equal to 3%, our GA still keeps its practicality both in computing time and approximation ratio. We have got two new computational results for our GA and LINGO 4. Fist one is that we get a Set Covering input data 999×999 density 10% for which LINGO 4 is not able to find a first feasible solution in *one hour and twelve minutes*, where we have used Gateway Select 800(Athlon 800 MHz). Second one is that for four input data sets 999×999 with density 1%, LINGO 4 has defeated our GA. So, we have carried out another computational experiment to see whether this phenomenon still holds for much bigger input data sets, say 2500×2500 density $2\%, 1.5\%, 1\%, 0.7\%, 0.5\%, 0.2\%$ and so on.We will report it in the near future.

We see that in the U.S. lots of research activities on the Set Covering and Set Partitioning problems have been carried out. In their papers , besides Set Covering/Set partitioning problems they treat several other problems of Airline Scheduling. The interested readers are advised to see Vasquez-Marquez[196](1991),Anbil et al.[5](1991), K.L.Hoffman and M.Padberg[63](1993), R.E.Bixby and E.K.Lee[12](1998), R.Borndoerfer[15]([1998]), E.R.Butchers et al.[17](2001).

In May 2001, E.Gunji et al.[57] have published "A Randomized and Genetic Hybrid Algorithm for the Traveling Salesman Problem" in Japanese. In it, they have carried out computational experiment for seven Traveling Salesman input data taken from a set of benchmark test input data. Their hybrid algorithm shows approximation ratios 0.78% at the worst. See also Y.Ymamoto and M.Kubo[201](1997). The readers who are interested in Parallel Branch and Bound are welcomed to see Y.Shinano[185](2000). M.X.Goemans and D.P. Williamson[52](1994) gave a probabilistic argument to create 0.878-Approximation Algorithm for MAX CUT and MAX 2SAT, which once more shows us the importance of probabilistic/statistic/randomized algorithms to solve the basic planning problems in Operations Research.

Bibliography

[1] De P.K., Acharya D. and K.C. Sahu, A chance-constrained goal programming model for capital budgeting, *Journal of the Operational Research Society*, Vol.33, 635-638, 1982.

[2] Aggarwal K.K., Chopra Y.C. and J.S. Bajwa, Topological layout of links for optimizing the s-t reliability in a computer communication system, *Microelectronics & Reliability*, Vol.22, No.3, 341-345,1982.

[3] Aggarwal K.K., Chopra Y.C. and J.S. Bajwa, Topological layout of links for optimizing the overall reliability in a computer communication system, *Microelectronics & Reliability*, Vol.22, No.3, 347-351,1982.

[4] Aho A.V., Hopcroft J.E. and J.D.Ullman,*The Design and Analysis of Computer Algorithms*, Addison- Wesley,1974.

[5] Anbil R., German E., Patty B. and R. Tanga, Recent Advances in Crew-Pairing Optimization at American Airlines, *Interfaces*, Vol.21, 62-74,1991.

[6] Applegate, Bixby, Chvatal and Cook, On the solution of traveling salesman problems, *Proc. of the International Congress of Mathematics*, Documenta Mathematica:Extra Volume ICM 1998:III, 645-656,1998.

[7] Balas E., An additive algorithm for solving linear programs with zero-one variables, *Operations Research*, Vol.13, 517-545, 1965.

[8] Balas E. and M.W.Padberg, On the Set-Covering Problem,*Operations Research*,Vol.20, 1152-1161,1972.

[9] Balas E. and M.W.Padberg, On the Set-Covering Problem II : An Algorithm for Set-Partitioning,*Operations Research*, Vol.23,74-90,1975.

[10] Bellman R.E. and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton, 1962.

[11] Birkhoff G., Rings of sets, *Duke Mathematical Journal*, Vol.3, 443-454,1937.

[12] Bixby R.E. and E.K.Lee, Solving a Truck Dispatching Scheduling Problem using Branch-and-Cut,*Operations Research*, Vol.46, 355-367,1998.

[13] Björner A. , On matroids, groups and exchange languages, *Colloquia Mathematica Societatis Janós Bolyai*, Vol.40, 25-60,1985.

[14] Björner A. Korte B. and L. Lovász , Homotopy properties of greedoids, *Advances in Applied Mathematics*, Vol.6, 447-494,1985.

[15] Borndoerfer R.,, *Aspects of Set Packing, Partitioning, and Covering*, Technishe Universitaet Berlin, 1998.

[16] Bouchet A., Greedy algorithm and symmetric matroids, *Mathematical Programming*, Vol.38, 147-159, 1987.

[17] Butchers E.R., Day P.R. ,Goldie A.P. ,Miller S. ,Meyer J.A. ,Ryan D.M. , Scott A.C. and C.A. Wallace, Optimized Crew Scheduling at Air New Zealand, *Interfaces*, Vol.31, 30-56,2001.

[18] Charnes A. and W.W. Cooper, Chance-constrained programming, *Management Science*, Vol.6, No.1, 73-79, 1959.

[19] Charnes A. and W. W. Cooper, *Management Models and Industrial Applications of Linear Programming,* New York: John Wiley & Sons, Inc., 1961.

[20] Charnes A. and W.W. Cooper, Goal programming and multiple objective optimizations: part I, *European J. of Operational Research*, Vol.1, No.1, 39-54, 1977.

[21] Chopra Y.C., Sohi B.S., Tiwari R.K.and K.K. Aggarwal, Network topology for maximizing the terminal reliability in a computer communication network, *Microelectronics & Reliability*, Vol.24, 911-913,1984.

[22] Christofides N. and S. Korman,A computational survey of methods for the set covering problem, *Management Science 21*,591-599,1974-75.

[23] Cook S.A.,The Complexity of Theorem-proving Procedures,*Proc. 3rd Ann. ACM Symposium on Theory of Computing*,ACM,151-158,1971.

[24] Dantzig G.B. ,*Linear Programming and Extensions*, Princeton University Press,1963.

[25] Dempster M.A.H.(ed), *Stochastic Programming*, Academic Press, London, 1980.

[26] Dengiz B., Altiparmak F. and A.E. Smith, Efficient optimization of all-terminal reliable networks, Using an evolutionary approach, *IEEE Transactions on Reliability*, Vol.46, No.1, 18-26,1977.

[27] Ding L.-Y. and M.-Y.Yue , On a generalization of the Rado-Hall theorem to greedoids, *Preprint, Institute für Operations Research, Universität Bonn*,1986.

[28] Dreyfus S.E. and K.L. Prather, Improved algorithms for Knapsack problems, *ORC 70-30*, Univ. of California, Berkeley, August, 1970.

[29] Dubois D. and H. Prade, *Possibility Theory*, Plenum, New York, 1988.

[30] Dubois D. and H. Prade, Fuzzy numbers: an overview, in *Analysis of Fuzzy Information*, Vol.2, Bezdek J.C., Ed., CRC Press, Boca Raton, 3-39, 1988.

[31] Even S.,*Graph Algorithms*, Computer Science Press,1979.

[32] Ermoliev Y. and R.J.-B. Wets (eds.), *Numerical Techniques for Stochastic Optimization*, Springer-Verlag, Berlin, 1988.

[33] Faigle U. , Goecke O. and R. Schrader , Church-Rosser decomposition in combinatorial structures,*Preprint, Institute für Operations Research, Universität Bonn*,1986.

[34] Fogel D. B., An introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks,* Vol.5, 3-14, 1994.

[35] Fujishige S., Algorithms for solving the independent-flow problems, *J. of the Operations Research Society of Japan*, Vol.21, No.2, 189-202, 1978.

[36] Fujishige S. , Lexicographically optimal base of a polymatroids with respect to a weight vector, *Math. of O.R.*, Vol.5, No.2, 186-196,1980.

[37] Fujishige S. , Submodular systems and related topics, *Mathematical Programming Study*, Vol.22, 113-131,1984.

[38] Fujishige S. , private communication, 1987.

[39] Fujishige S. , *Submodular functions and optimization*, North-Holland,1991.

[40] Fujishige S. and N. Tomizawa , A note on submodular functions on distributive lattices, *J. of the Operations Research Society of Japan*, Vol.26, No.4, 309-317,1983.

[41] Fukuhara T. , Structural Study of a Graph Drawing Problem on k - Parallel Lines(in Japanese), Graduate Course - Tokyo Institute of Technology,1990.

[42] Fushimi M. , Designing a uniform random number generator whose subsequences are k_distributed, *SIAM Journal on Computing*, Vol.17, No.1, 89-99,1988.

[43] Futakawa M. , Yamada T. and S.Kataoka, A heuristic algorithm to solve Max-Min knapsack problem, Fall Meeting of Japanese Operations Research Society,1995.

[44] Garey M.R. and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*,Freeman,1979.

[45] Garfinkel R.S. and G.L.Nemhauser, The Set Partitioning Problem:Set Covering with Equality Constraints,*Operations Research*,Vol.17,848-856,1969.

[46] Garfinkel R.S. and G.L.Nemhauser,*Integer Programming*,John Wiley & Sons, 1972.

[47] Gilmore P.C. and R.E. Gomory, A linear programming approach to the cutting stock problem, *Operations Research*, Vol.9, 849-859, 1961.

[48] Gilmore P.C. and R.E. Gomory, A linear programming approach to the cutting stock problem, Part II, *Operations Research*, Vol.11, 863-888, 1963.

[49] Gilmore P.C. and R.E. Gomory, Multistage cutting stock problems of two and more dimensions, *Operations Research*, Vol.13, 94-120, 1965.

[50] Gilmore P.C. and R.E. Gomory, The Theory and Computation of Knapsack Function, *Operations Research*, Vol.14, 1045-1074, 1966.

[51] Goecke O. and R. Schrader , Minor characterization of undirected branching greedoids - a short proof, *Preprint, Institute für Operations Research, Universität Bonn*,1986.

[52] Goemans M.X. and D.P.Williamson, .878-Approximation Algorithms for MAX CUT and MAX 2SAT, *Proc. of 26th STOC*, ACM,422-431,1994.

[53] Goetschel R. H. Jr. , Linear objective functions on certain classes of greedoids,*Discrete Applied Mathematics*, Vol.14, 11-16,1986.

[54] Goldberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning,* ,Addison-Wesley, 1989.

[55] Gomory R.E., Some polyhedra related to combinatorial problems, *Linear Algebra and Its Applications*, Vol.2, 451-558, 1969.

[56] Greenberg H., An algorithm for the computation of Knapsack functions, *Journal of Mathematical Analysis and Applications*, Vol.26, 159-162, 1969.

[57] Gunji E., Tomita E. and M.Wakatsuki,A Randomized and Genetic Hybrid Algorithm for the Traveling Salesman Problem(in Japanese), in *Mathematical Modeling and Problem Solving(Suuri Moderuka to Mondai Kaiketsu)*,Information Processing Society of Japan, 61-64, March 16, 2001.

[58] Hayashi Y. , An efficient algorithm to solve the 0-1 knapsack problem when $J = 2^n - 1$, *Abstracts of the 1995 Fall National Conference of the Operations Research Society of Japan*,244-245(in Japanese),1995.

[59] Hayashi Y. , *Theory and Algorithm of 0-1 Knapsack Problem* (in Japanese), Kinki University, Japan,2000.

[60] Hillier F. S. , Efficient Heuristic Procedures for Integer Linear Programming with an Interior,*Operations Research*, Vol. 17, 600-637,1969.

[61] Hirabayashi R., Ikebe Y.T., Iwamura K. and T.Nakayama, Drawing a Tree on Parallel Lines, *Computers and Mathematics with Applications*, Vol.37,171-176,1999.

[62] Hoffman A.J. , On simple linear programming problems,*Convexity*, American Mathematical Society,1963.

[63] Hoffman K.L. and M. Padberg, Solving Airline Crew Scheduling Problems by Branch and Cut, *Management Science*, Vol.39,657-682,1993.

[64] Holland J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[65] Horowitz E. and S.Sahni, *Algorithms: Design and Analysis*, Computer Science Press, Maryland, 1978.

[66] Hu T.C., *Integer Programming and Network Flows*, Addison-Wesley, 1970.

[67] Ibaraki T. , Solvable classes of discrete dynamic programming, *J. of Mathematical Analysis and Applications*, Vol.43, 642-693, 1973.

[68] Ibaraki T., *Theory of Combinatorial Optimization*(in Japanese) , Corona-shya,1981.

[69] Ibaraki T. , Combinatorial Optimization and Its Complexity (in Japanese), *Journal of Japanese Society for Artificial Intelligence(Jinkouchinougakkaishi)*, Vol.9 No.3,335-341,1994.

[70] Ibaraki T. and M.Fukushima, *Method of Optimization*(in Japanese), Kyouritu-syuppan ,1993.

[71] Ikegami A. and A.Niwa,The Vehicle Routing Problem with Time Constraints ,*J. of the Operations Research Society of Japan*, Vol.38, No.1, 107-123,1995.

[72] Ikura Y. , Private communication,1976.

[73] Ikura Y. , On the set partitioning problem, Master's thesis, Faculty of Engineering , University of Tokyo, 1976.

[74] Imai H., New Trends in Design and Analysis of Approximation Algorithms: Some Hard Problems in Sets and Logics as Examples,*The Journal of Institute of Electronics, Information and Communication Engineers*, Vol.79, No.6,920-926,1996.

[75] Iri M. , An algebraic approach to the problem of topological degrees of freedom of a network, *Trans. Inst. Electronics and Communication Engineers of Japan*, Vol.51A, No.5,180-187,1968.

[76] Iri M. ,The Maximum-Rank Minimum-Term-Rank Theorem for the Pivotal Transforms of a Matrix,*Linear Algebra and Its Applications*, Vol.2, 427-446,1969.

[77] Iri M. ,*Network Flows, Transportation and Scheduling - Theory and Algorithms*, Academic Press, New York,1969.

[78] Iri M. , A review of recent work in Japan on principal partitions of matroids and their applications,*Annals of the New York Academy of Sciences*, Vol.319,306-319,1979.

[79] Iri M. , Structural Theory for the Combinatorial Systems Characterized by Submodular Functions,in *W.R.Pulleyblank(ed.): Progress in Combinatorial Optimization*, Academic Press, 197-219,1984.

[80] Iri M., Fujishige S. and T.Ohoyama, *Graph-Network-Matroid*, Sangyohotoshyo(in Japanese),1986.

[81] Iwamoto S. , *Dynamic Programming*, Kyushyudaigaku-Shyppankai (in Japanese),1987.

[82] Iwamura K., A solution procedure of an all integer linear programming appearing in CFLP (based on Boolean branch and bound), (in Japanese), *Research Meeting of Mathematical Programming*, Operations Research Society of Japan, October, 1972.

[83] Iwamura K., On Some Theorems of Knapsack Function , *Journal of the Operations Research Society of Japan*, Vol.17,No.4,173-183, 1974.

[84] Iwamura K. , Developing an efficient program code to solve the set partitioning problem,part I(in Japanese),*Abstracts of Spring Conference of the Operations Research Society of Japan*, 107-108,1978.

[85] Iwamura K. , Developing an efficient program code to solve the set partitioning problem, part II(in Japanese),*Abstracts of Spring Conference of the Operations Research Society of Japan*,57-58,1979.

[86] Iwamura K. , On integrality preserving pivot and cycling of the set partitioning problem(in Japanese), *Abstracts of Fall Conference of the Operations Research Society of Japan*,142-143,1980.

[87] Iwamura K. , Knapsack typed ILP has a fully polynomial time approximation scheme when its cost coefficients are all bounded(in Japanese), *Abstracts of Spring Conference of the Operations Research Society of Japan* ,25-26,1981.

[88] Iwamura K. , Developing an efficient program code to solve the set partitioning problem, part III(in Japanese),*Abstracts of Spring Conference of the Operations Research Society of Japan*,57-58, 1982.

[89] Iwamura K., Greedy algorithm for primal dual matroids (in Japanese), *Proceedings of Symposium in Language and Automata Theory*,1985.

[90] Iwamura K., Contracting Greedoids and a Rado-Hall Type Theorem, *Institut für Ökonometrie und Operations Research*, Universität Bonn,1988.

[91] Iwamura K. , Discrete Decision Process Model Involves Greedy Algorithm Over Greedoid, *Journal of Information and Optimization Sciences*, Vol.14, 83-86 ,1993.

[92] Iwamura K. , Lexicographically optimal base of a submodular system with respect to a weight vector, *Journal of Information and Optimization Sciences*, Vol.16, 49-60,1995.

[93] Iwamura K. , Primal Dual Algorithms for the Lexicographically Optimal Base of a Submodular Polyhedron and its Relation to a Poset Greedoid, *Journal of Systems Science and Systems Engineering*,Vol.4, No.3,193-203,1995.

[94] Iwamura K. and Y.Deguchi, An inverse problem for two-dimensional discrete constant spring system,(in Japanese) *Suurikaisekikennkyujo-koukyuuroku798*, 219-225,1992.

[95] Iwamura K., Deguchi Y. and N.Okada, A remark on solving the set-partitioning problem by dual all integer algorithm,*Journal of Systems Science and Systems Engineering*, Vol. 7, No. 3,363-367, 1998.

[96] Iwamura K. and O. Goecke, An application of greedoid to matroid theory, *Preprint, Josai University, Sakado, Saitama, Japan*,1985.

[97] Iwamura K., Horiike M. and T. Sibahara, Input Data Dependency of a Genetic Algorithm to Solve the Set Covering Problem, *Tsinghua Science and Technology*, Vol.8, No.1, 14-18, 2003.

[98] Iwamura K. and B. Liu, A genetic algorithm for chance constrained programming, *Journal of Information & Optimization Sciences*, Vol.17 , No.2, 409-422,1996.

[99] Iwamura K. and B. Liu, Chance constrained integer programming models for capital budgeting in fuzzy environments, *Journal of the Operational Research Society*, Vol.49,854-860,1998.

[100] Iwamura K. and B.Liu,Dependent Chance Integer Programming Applied to Capital Budgeting, *Journal of the Operations Research Society of Japan*, Vol.42,117-127,1999.

[101] Iwamura K. and Y. Maeda, Core Saving Criterion for The MCGP of The Set Partitioning Algorithm Proposed by Balas and Padberg, preprint,1977.

[102] Iwamura K. and Y. Maeda, A computational Experiment of The Set Partitioning Algorithm Proposed by Balas and Padberg,preprint,1977.

[103] Iwamura K. and G.Nakamura, An inverse problem for two-dimensional elastic body,*Research Report of Faculty of Sciences, Josai University*, Vol.2,25-31,1994.

[104] Iwamura,K. and T.Nakayama, Drawing a Tree on Parallel Lines, *Proc. of International Workshop on Intelligent Systems and Innovative Computations*–The 6th Bellman Continuum–235-238,1994.

[105] Iwamura K. and N.Okada, Knapsack problem, set-partitioning problem and some algorithmic prospect to solve real-world NP-complete problems, in*Proceedings of the 3rd Young Chinese Conference in Operations Research*, China Higher Education Press Beijing and Springer -Verlag Heidelberg ,283-289,1999.

[106] K.Iwamura, T.Sibahara, M.Fushimi and H.Morohoshi[2000], Set Covering Problem, Genetic Algorithm and Its Domain Specific Knowledge, in *Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications*, May 3-5, 2000, Global-Link Publishing Company, Hong Kong, 2000,pp.250-257.

[107] K.Iwamura, T.Sibahara, M.Fushimi and H.Morohoshi[2001], Set Covering Problem, Genetic Algorithm and Its Domain Specific Knowledge, *Information*, vol.4,no.3,pp.279-286, 2001.

[108] Jan R.-H., Hwang F.-J. and S.-T. Chen, Topological optimization of a communication network subject to a reliability constraint, *IEEE Transactions on Reliability*, Vol.42, 63-70,1993.

[109] Kall P. and S.W. Wallace, *Stochastic Programming,* John Wiley & Sons, 1994.

[110] Karp R.M., Reducibility among Combinatorial Problems, in R.E. Miller and J.W. Thatcher(eds.), *Complexity of Computer Computations*, Plenum Press, 85-103, 1972.

[111] Keown A.J. and J.D. Martin, A chance constrained goal programming model for working capital management, *Engng Econ.*, Vol.22, 153-174, 1977.

[112] Keown A.J. and B.W. Taylor, A chance-constrained integer goal programming model for capital budgeting in the production area, *Journal of the Operational Research Society*, Vol.31, No.7, 579-589, 1980.

[113] Kiuchi M., Shinano Y., Saruwatari Y. and R.Hirabayashi, An exact parallel branch and bound algorithm for capacitated edge traveling problem : Part 2(in Japanese), Faculty of Engineering, Tokyo Science University,1995.

[114] Klein C.M. , A Submodular approach to discrete dynamic programming *European Journal of Operations Research*, Vol.80,147-155,1995.

[115] Kobayashi K., On existence of non-"a priori" malign measures, Faculty of Science, Tokyo Institute of Technology,1992.

[116] Konno H. and H.Suzuki,*Integer Programming and Combinatorial Optimization(in Japanese)*, Nikkagiren-syuppansha, 1982.

[117] Korte B. and L. Lovász , Structural properties of greediods, *Combinatorica*, Vol.3, Nos.3-4, 359-374,1983.

[118] Korte B. and L. Lovász , Greedoids – a structural framework for the greedy algorithm, in W.R. Pulleyblank (ed.): *Progress in Combinatorial Optimization*, Academic Press, 221-243,1984.

[119] Korte B. and L. Lovász, Shelling structures, convexity and a happy end, in B. Bollobás (ed.): *Graph Theory and Combinatorics*, Academic Press, 219-232,1984.

[120] Korte B. and L. Lovász, Greedoids and linear objective functions, *SIAM J. Alg. Disc. Math.*, Vol.5, No.2, 229-238,1984.

[121] Korte B. and L. Lovász, A note on selectors and greedoids, *Europ. J. of Combinatorics*, Vol.6, 59-67,1985.

[122] Korte B. and L. Lovász, Poset, matroids and greedoids, *Colloquia Mathematica Societatis Janoś Bolyai*, Vol.40, 239-265,1985.

[123] Korte B. and L. Lovász, Polymatroid greedoid, *J. of Combinatorial Theory, Series B*, Vol.38, No.1, 41-72,1985.

[124] Korte B. and L. Lovász , Relations between subclasses of greedoids, *Zeitschrift für Operations Research*, Series A 29, 249-267,1985.

[125] Korte B. and L. Lovász , Basis graphs of greedoids and two-connectivity,*Mathematical Programming Study*, No.24, 158-165,1985.

[126] Korte B. and L. Lovász , Intersections of matroids and antimatroids, *Preprint, Institute für Operations Research, Universität Bonn*,1985.

[127] Korte B. and L. Lovász , Homomorphisms and Ramsey properties of antimatroids, *Discrete Applied Mathematics*, Vol.15, 283-290,1986.

[128] Korte B. and L. Lovász , Non interval greedoids and the transposition property, *Discr. Math.*, 59, 297-314,1986.

[129] Korte B. Lovász L. and R.Schrader, *Greedoids*, Springer,1991.

[130] Kubo M. ,Meta heuristics , in *Discrete Structures and Algorithms IV(ed. K.Murota, in Japanese)*, 171-230, Kindaikagakushya,1995.

[131] Kumar A., Pathak R.M. and Y.P. Gupta, Genetic-algorithm-based reliability optimization for computer network expansion, *IEEE Transactions On Reliability*, Vol.44, 63-72,1995.

[132] Kumar A., Pathak R.M. , Gupta Y.P. and H.R. Parsaei, A Genetic algorithm for distributed system topology design, *Computers and Industrial Engineering*, Vol.28, 659-670,1995.

[133] Kuroda, Private communication,1976.

[134] Lai T.-C., Brandeau M.L. and S. Chiu, An Approach for Worst Case Analysis of Heuristics:Analysis of a Flexible 0-1 Knapsack Problem,*J. of the Operations Research Society of Japan*,Vol.37 No.3, 197-210,1994.

[135] Lawler E.L., *Combinatorial Optimization: Networks and Matroids*, Holt,Rinehart&Winston, New York,1976.

[136] Lawler E.L., Fast Approximation Algorithms for Knapsack Problems, *Mathematics of Operations Research*, Vol.4, 339-356, 1979.

[137] Lawler E. L. , Lenstera J.K. , Rinnooy Kan A.H.G. and D.B.Shmoys, *The Traveling Salesman Problem*, John Wiley & Sons,1987.

[138] Lemk C.,Salkin H. and K. Spielberg,Set covering by Single Branch Enumeration with Linear Programming Subproblems, *Operations Research* Vol.19, 998-1022,1971.

[139] Li M. and P.M.B. Vitanyi, A Theory of Learning Simple Concepts Under Simple Distributions and Average Case Complexity for the Universal Distribution,*Proc. of the 30th FOCS*, 34-39,1989.

[140] Lin S.,Computer solutions of the traveling salesman problem,*Bell System Tech. J.*,Vol.44,2245-2269,1965.

[141] Lin S. and B.W.Kernighan,An Effective Heuristic Algorithm for the Traveling-salesman problem,*Operations Research*,Vol.21,498-516,1973.

[142] Little J.D.C., Murty K.G., Sweeney D.W. and C.Karel, An Algorithm for the Traveling Salesman Problem,*Operations Research* Vol.11,979-989,1963.

[143] Liu B., Dependent-chance goal programming and its genetic algorithm based approach, *Mathematical and Computer Modelling*, Vol.24, No.7, 43-52,1996.

[144] Liu B., Dependent-chance programming: A class of stochastic optimization, *Computers & Mathematics with Applications*, Vol.34, No.12, 89-104,1997.

[145] Liu B., Dependent-chance programming in fuzzy environments, to appear in *Fuzzy Sets and Systems*.

[146] Liu B., Dependent-chance programming with fuzzy decisions, *IEEE Transactions on Fuzzy Systems*, Vol.7, No.3, 354-360,1999.

[147] Liu B., *Uncertain Programming*, John Wiley & Sons, New York, 1999.

[148] Liu B. and K. Iwamura, Modelling stochastic decision systems using dependent-chance programming, *European Journal of Operational Research*, Vol.101,193-203,1997.

[149] Liu B. and K. Iwamura, Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems*, Vol.94, No.2, 227-237,1998.

[150] Liu B. and K. Iwamura, A note on chance constrained programming with fuzzy coefficients, *Fuzzy Sets and Systems*, Vol.100, Nos.1-3, 229-233,1998.

[151] Liu B. and K. Iwamura, Topological Optimization Models for Communication Network with Multiple Reliability Goals, *Computers and Mathematics with Applications*, Vol.39,59-69,2000.

[152] Lührs J.G., Ein Einschliessungssatz zum Knapsack-problem, *Computing*, Vol.9, 101-105, 1972.

[153] Maeda E. and K.Iwamura ,Set Covering and Set Partitioning Problems,in*Integer Programming and Combinatorial Optimization*(in Japanese,eds. H.Konno and H.Suzuki)185-209 ,Nikkagiren-syuppansha,Tokyo,1982.

[154] Martello S. and P.Toth, *Knapsack Problems*,John Wiley & Sons,1990.

[155] Megiddo N., Optimal flows in networks with multiple sources and sinks. *Math. Programming*, 7, 97-107,1974.

[156] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd ed., Springer-Verlag, New York, 1996.

[157] Mineno Y. ,Private communication,1978.

[158] Mirsky L. , Transversal theory, Academic Press,1971.

[159] Morohoshi H. and M.Fushimi, Designing a uniform random number generator with approximate asymptotic randomness(in Japanese),*Nihon Ouyou Suuri Gakkai Ronbunshi*, Vol.8, No.1,25-33,1998.

[160] Morton G., von Randow R. and K. Ringwald, A greedy algorithm for solving a class of convex programming problems and its connection with polymatroid theory, *Math. Prog.*, Vol.32, 238-241,1985.

[161] Mukawa H.,Sensui J., Iwamura K. and J.Kase, An algorithm to solve the Capacitated Facilities Location Programming Problem,in Japanese,1971.

[162] Murota K. ,*Convexity and Steinitz's Exchange Property*, RIMS-1023,Research Institute for Mathematical Sciences, Kyoto Univ.,1995.

[163] Murota K. , *Discrete Convex Analysis* , RIMS-1065, Kyoto Univ.,1996.

[164] Nakamura M. , Structural Theorems for Submodular Functions, Polymatroids and Plymatroid Intersections,*Graphs and Combinatorics* ,Vol.4,257-284,1988.

[165] Nemhauser G.L. and L.A.Wolsey, *Integer and Combinatorial Optimization* , Wiley, 1988.

[166] Odanaka T. and K.Iwamura, Prediction Theory of Discrete Multiple Time Series, *Beijing Mathematics*, Vol.4,284-288,1998.

[167] Okada N., Iwamura K. and Y. Deguchi, A Computational Study of a Genetic Algorithm to Solve the Set Covering Problem, to appear in *Journal of Interdisciplinary Mathematics*.

[168] Painton L. and J. Campbell, Genetic algorithms in optimization of system reliability, *IEEE Transactions on Reliability*, Vol.44, 172-178,1995.

[169] Papadimitriou C.H. and K. Steiglitz,*Combinatorial Optimization*, Prentice-Hall, Englewood Cllifs, N.J.,1982.

[170] Pierce J.F. and J.S.Lasky, Improved combinatorial programming algorithms for a class of all-zero-one integer programming problems,*Management Science*, Vol.19, 528-543,1973.

[171] Ravi V., Murty B.S.N. and P.J. Reddy, Non equilibrium simulated annealing-algorithm applied to reliability optimization of complex systems, *IEEE Transactions on Reliability*, Vol.46, 233-239,1977.

[172] Reinelt G. ,*The Traveling Salesman*, Springer,1994.

[173] Rubinstein R.Y., *Simulation and the Monte Carlo Method*, John Wiley & Sons, Inc., 1981.

[174] Salkin H.M. ,*Integer Programming*,Addison-Wesley, 1975

[175] Salkin H.M. and R.Koncal, A pseudo dual all-integer algorithm for the set covering problem, Technical memorandum, No.204, Case Western Reserve University,1970.

[176] Salkin H.M. and R.Koncal, A dual all-integer algorithm(in revised simplex form) for the set covering problem. Technical memorandum No.250, Case Western Reserve University,1971.

[177] Salkin H.M. and R.Koncal, Set covering by an all integer algorithm; Computational experience, *J. ACM*, Vol.20, 189-193,1973.

[178] Saruwatari Y., Hirabayashi R. and N. Nishida, Node Duplication Lower Bounds for the Capacitated Arc Routing Problem,*J. of the Operations Research Society of Japan*, Vol.35, No.2, 119-133,1992.

[179] Schmid B., Ganzzahlige lineare programmierung und das Knapsack-problem, *Z. Wahrscheinlichkeitstheorie view. Geb.*, Vol.23, 255-259, 1972.

[180] Schmidt W. , A characterization of undirected branching greedoids, *Preprint, Institute für Operations Research, Universität Bonn*,1985.

[181] Schmidt W. , Greedoids and searches in directed graphs, *Preprint, Institute für Operations Research, Universität Bonn*,1985.

[182] Schmidt W. , A mini-max theorem for greedoids, *Preprint, Institute für Operations Research, Universität Bonn*,1985.

[183] Schrijver A.,*Theory of Linear and Integer Programming*, Wiley, Chichester,1986.

[184] Shapiro J.F. and H.M. Wagner, A finite renewal algorithm for the Knapsack and Turnpike models, *Operations Research*, Vol.15, 319-341, 1967.

[185] Shinano Y.,Parallelization Utility for Branch and Bound Algorithms PUBB(in Japanese), *Communications of the Operations Research Society of Japan(Opereesyonzu Risaachi)*, Vol.45,No.3,112-117,2000.

[186] Shindo M. and E.Tomita, A Simple Algorithm for Finding a Maximum Clique and Its Worst-Case Time Complexity(in Japanese), *The Transactions of the Institute of Electronics, Information and Communication Engineers D*,Vol.J71-D,No.3, 472-481,1988.

[187] Sniedovich M. , *Dynamic Programming*, Dekker,1992.

[188] Sorimachi Y., Optimal Planning of Distribution Center,*Suurikagaku*, September,23-27(in Japanese),1970.

[189] *Suugaku Seminar*, Nippon Hyoronsha, Tokyo, Japan, September, 6-13, 1973.

[190] Suzuki H. and K. Iwamura,Knapsack Problem and Set Covering (Set Partitioning) Problem, *Communications of the Operations Research Society of Japan(Opereesyonzu Risaachi)*, 359-368(in Japanese),June 1979.

[191] Tagawa K. ,Okada D. ,Kanzaki Y. , Inoue K. and H.Haneda, Distance Based Hybrid Genetic Algorithm for Symmetric and Asymmetric Travelling Salesman Problems, *Beijing Mathematics*,Vol.4,No.2,42-49,1998.

[192] Tang X. and J.Gu, Soft System Approach to Management Support System Development, A lecture at the Dept. Math., Josai University,1996.

[193] Tomita E., Imamatsu K., Kohata Y. and M.Wakatsuki, A Simple and Efficient Branch and Bound Algorithm for Finding a Maximum Clique with the Experimental Evaluations(in Japanese),*The Transactions of the Institute of Electronics, Information and Communication Engineers D-I*, Vol.J79-D-I, No.1,1-8,1996.

[194] Trevisan L., When Hamming Meets Euclid:The Approximability of Geometric TSP and Steiner Tree,*SIAM J. COMPUT.*, Vol.30,No.2,475-485,2000.

[195] Van de Panne C. and W. Popp, Minimum cost cattle feed under probabilistic protein constraints, *Management Science,* 9, 405-430, 1963.

[196] Vasquez-Marquez, American Airlines Arrival Slot Allocation System(ASAS), *Interface*, Vol.21,42-61,1991.

[197] G. Wang and K.Iwamura, Yield Management with Random Fuzzy Demand, in *Proceedings of the First International Conference on Information and Management Sciences*, Xi'an, China, May 27-31, 2002, pp.192-195.

[198] Weismantel R., *Knapsack Problems, Test Sets and Polyhedra*, Technishe Universitaet Berlin, 1992.

[199] Welsh D.J.A. , *Matroid theory*, Academic Press,1976.

[200] Xie J. and W.Xing,Incorporating Domain-Specific Knowledge into Evolutionary Algorithms, *Beijing Mathematics*, Vol.4, No.2, 131-139,1998.

[201] Yamamoto Y. and M.Kubo,*An Invitation to the Traveling Salesman Problem*(in Japanese) ,Asakura Shoten, Tokyo, Japan,1997.

[202] Young R.D., A primal (all integer) integer programming algorithm, *J. of Res. of N.B.S.*, Vol.69B,No. 3, 213-250, 1965.

[203] Zadeh L.A., Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, Vol.1, 3-28, 1978.

[204] Zhao R., Iwamura K. and B. Liu, Chance Constrained Integer Programming and Stochastic Simulation Based Genetic Algorithm, *Journal of Systems Science and Systems Engineering*, Vol.7,96-102,1998.

[205] Zhao R., Iwamura K. and B.Liu, A genetic Algorithm for multivariate isotonic regression,*Journal of Information & Optimization Sciences*,Vol.19,273-284,1998.

[206] Zimmermann U.,*Linear and Combinatorial Optimization in Ordered Algebraic Structures*, North Holland,1981.

Author Biography

Kakuzo Iwamura was born in Tochigi, Japan, on September 25, 1945. He received the B.S. degree from the Tokyo Institute of Technology, Tokyo, Japan in March 1968, and the M.S. degree from the same university in March 1970. From April 1970 to March 1972, he worked for Mitsubishi Research Institute. Then from April 1972 to March 1975 he worked at the Department of Mathematics, Josai University, Japan as an assistant. From April 1975 to present, he has been working for the same department as lecturer.

His research interests include Discrete Optimization, Dynamic Programming, Greedy Algorithm, Planning under Uncertainty , Fuzzy and/or Genetic Algorithms.

Mr. Iwamura is a member of the Operations Research Society of Japan, the Institute of Electronics, Information and Communication Engineers, the Japan Society for Industrial and Applied Mathematics, the European Association for Theoretical Computer Science and Language & Automata(Japan).