

國立交通大學

資訊管理研究所

博士論文

整數規劃之高效率求解方法及其運用

An Effective Method for Solving Large Binary Programs and
Its Applications



研究生：盧浩鈞

指導教授：黎漢林 講座教授

中華民國九十七年六月

整數規劃之高效率求解方法及其運用

An Effective Method for Solving Large Binary Programs and
Its Applications

研究生：盧浩鈞

Student : Hao-Chun Lu

指導教授：黎漢林

Advisor : Han-Lin Li

國立交通大學



A Dissertation

Submitted to Institute of Information Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Information Management
June 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

國立交通大學

論文口試委員會審定書

本校 資訊管理 研究所博士班 盧浩鈞 君

所提論文 (中) 整數規劃之高效率求解方法及其運用

(英) An Effective Method for Solving Large Binary Programs and Its Applications

合於博士資格水準、業經本委員會評審認可。

口試委員：

曾國雄

陳汝生

黎漢林

溫子平

李永銘

林妙聰

指導教授：黎漢林 教授

研究所所長：劉敦仁 教授

中華民國 九十七年 六月 二日

整數規畫之高效率求解方法及其運用

學生：盧浩鈞

指導教授：黎漢林

國立交通大學資訊管理研究所博士班

摘要

許多非線性問題需要逐段線性技術(Piecewise Linearization)將原始問題線性化以求得全域最佳解，而這過程需要加入許多二進位變數。近四十年來發展許多逐段線性之技術，而本研究就是發展出一套二進位變數的超級展現法(SRB)以便降低逐段線性技術所需之二進位變數其限制式。當一個擁有 $m+1$ 個中斷點的線性化函數時，現有之逐段線性技術必須用到 m 個二進位變數及 $4m$ 個限制式，而本研究提出之方法只需 $\lceil \log_2 m \rceil$ 個二進位變數及 $8 + 8\lceil \log_2 m \rceil$ 個限制式。同時本研究也展現數個實際問題之應用以證明它的高效率。

關鍵字：二進位變數, 逐段線性化.

An Effective Method for Solving Large Binary Programs and Its Applications

Student : Hao-Chun Lu

Advisor: Han-Lin Li

Institute of Information Management

National Chiao Tung University

ABSTRACT

Many nonlinear programs can be linearized approximately as piecewise linear programs by adding extra binary variables. For the last four decades, several techniques of formulating a piecewise linear function have been developed. This study develops a Superior Representation of Binary Variables (SRB) and applied in many applications. For expressing a piecewise linear function with $m+1$ break points, the existing methods, which incurs heavy computation when m is large, require to use m additional binary variables and $4m$ constraints while the SRB only uses $\lceil \log_2 m \rceil$ binary variables and $8+8\lceil \log_2 m \rceil$ additive constraints. Various numerical experiments of applications demonstrate the proposed method is more computationally efficient than the existing methods.

Keywords: Binary variable, Piecewise Linearization

誌 謝

首先感謝 黎漢林老師的費心指導，這些成果完全仰賴指導教授，及 Yinyu Ye 教授 及 Christodoulos A. Floudas 教授的鼓勵，還有論文口試委員的指導使本論文更加完善。寫論文時，很多想法來自於實驗室的傳承，其中類 0-1 變數的概念來自於黎老師與胡念祖學長的概念，GGP 的動機完全來自於蔡榮發學長的研究，Piecewise 也是來自於張錦特學長的論文。自己的想法不多加上我的英文並不好，所以要感謝大家與上天，我只是最後那一根稻草時間到了研究結果就水到渠成。

在求學的歲月中，研究室的同學也給我許多的幫助與關懷。耀輝常幫忙我，在此向你致謝；也常麻煩昶瑞開車；宇謙學長，實驗室的大好人，常常關心我；還有明賢、大姐等。感謝你們陪我走過這段過程，不管是在做學問上、生活上及娛樂上，有了你們才使我在這段過程中常保有愉快的笑聲。

最後要感謝家人及女友的體諒，使我能夠專心一意完成學業，總之謝謝你們。



浩鈞 2008/06/10.

Contents

摘要	iv
ABSTRACT	v
誌謝	vi
Contents.....	vii
Tables	viii
Figures	ix
Chapter 1 Introduction	1
1.1 Research Background	1
1.2 Research Motivation and Purpose	4
1.3 Examples	5
1.4 Structure of the Dissertation	6
Chapter 2 A Superior Representation Method for Piecewise Linear Functions.....	8
2.1 Introduction to Piecewise Linear Functions	8
2.2 Superior Representation of Binary Variables (SRB)	10
2.3 The Proposed Linear Approximation Method	15
2.4 Numerical Examples.....	18
Chapter 3 Extension 1 – Solving Generalized Geometric Programs Problems.....	28
3.1 Introduction to Generalized Geometric Programming	28
3.2 The Proposed Linear Approximation Method	33
3.3 Treatment of Discrete Function in GGP (Approach 2).....	40
3.4 Treatment of Continuous Function in GGP.....	44
3.5 Numerical Examples.....	49
Chapter 4 Extension 2 – Solving Task Allocation Problem.....	59
4.1 Introduction to Task Allocation Problem.....	59
4.2 The Proposed Mathematical Programming Formula.....	60
4.3 Experiment Results.....	65
Chapter 5 Discussion and Concluding Remarks	66
5.1 Discussion.....	66
5.2 Concluding Remarks	67
References.....	69

Tables

Table 2.1	List of $ G(\theta) $, $c_{\theta,j}$, and $A_{\theta}(\theta')$	11
Table 2.2	Experiment 1 ($\alpha_1 = 0.4$, $\beta_1 = 2$, $\alpha_2 = 1.85$, $\beta_2 = 2$, $b = 5$)	20
Table 2.3	Experiment 2 (for different combinations of α_1 , β_1 , α_2 , β_2 and b)	22
Table 2.4	Experiment result of Example 2.2	24
Table 2.5	Experiment results with discrete function in ILOG CPLEX	26
Table 2.6	Experiment results with continuous function in ILOG CPLEX	27
Table 3.1	Way of expressing $y \in \{d_1, \dots, d_r\}$	43
Table 3.2	Experiment result of Example 3.3	56
Table 3.3	Comparisons of existing methods with proposed method	58
Table 4.1	Experiment result of TAP	65



Figures

Figure 1.1	Structure of the dissertation	7
Figure 2.1	Piecewise linear function $f(x)$	9
Figure 2.2	Contour map of the three-humped camelback function.....	23



Chapter 1 Introduction

1.1 Research Background

Optimization has many applications in various fields, such as finance (Sharpe 1971; Young 1998), allocation and location problem (Chen et al. 1995; Faina 2000), engineering design (Duffin et al. 1967; Fu et al. 1991; Sandgren 1990), system design and database problems (Lin and Orłowska 1995; Rotem et al. 1993; Sarathy et al. 1997), chemical engineering (Floudas 1999; Ryoo and Sahinidis 1995), and molecular biology (Ecker et al. 2002). Those optimization problems most are either nonlinear program or mixed-integer program and involve continuous and/or discrete variables. Guaranteed to obtain global optimization and improve the efficiency of algorithm are two important and necessary areas of research in optimization.



There are two categories of approaches for optimization field: deterministic and heuristic. Deterministic approaches take advantage of analytical properties of the problem to generate a sequence of points that converge to obtain a global optimization. There are three methods for deterministic approaches from recent literatures as follows.

- (i) Difference of Convex functions (DC): The DC optimization method is important in solving optimal problems and has been studied extensively by Horst and Tuy (1996), Floudas and Pardalos (1996), Tuy (1998), and Floudas (2000), etc. By utilizing logarithmic or exponential transformations, the initial GGP program can be reduced to a DC program. A global optimum can be found by successively refining a convex relaxation and the subsequent solutions of a series of nonlinear convex problems. A

major difficulty in applying this method for solving optimal problems is that the lower bounds of variables are not permitted to less than zero.

(ii) Multilevel Single Linkage Technique (MSLT): Rinnooy and Timmer (1987) proposed MSLT for finding the global optimum of a nonlinear program. Li and Chou (1994) also solved a design optimization problem using MSLT to obtain a global solution at a given confidence level. However, the difficulty of MSLT is that it has to solve numerous nonlinear optimization problems based on various starting points.

(iii) Reformation Linearization Technique (RLT): RLT was developed by Sherali and Tuncbilek (1992) to solve a polynomial problem. Their algorithm generate nonlinear implied constraints by taking the products of the bounding terms in the constraint set to a suitable order. The resulting problem is subsequently linearized by defining new variables. By incorporating appropriate bound factor products in their RLT scheme, and employing a suitable partitioning technique, they developed a convergent branch and bound algorithm has been developed. Although the RLT algorithm is very promising in solving optimization problems, a major associate difficulty is that it may generate a huge amount of new constraints.

The other category approaches, heuristic method, can get solution quickly, but the quality of the solution cannot be guaranteed as a global optimization. Meta-heuristic is a advanced heuristic method for solving a very general class of computational problems by

combining user given black-box procedures in a hopefully efficient way. Meta-heuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method. Most commonly used meta-heuristics are targeted to combinatorial optimization problems. Those popular meta-heuristic methods are listed as follows.

(i) Genetic Algorithm (GA): GA (Goldberg 1989) is based on the idea of the survival of the fittest, as observed in nature. It maintains a pool of solutions rather than just one. The process of finding superior solutions mimics that of evolution, with solutions being combined or mutated to alter the pool of solutions, with solutions of inferior quality being discarded.

(ii) Simulated Annealing (SA): SA is a related global optimization technique which traverses the search space by generating neighboring solutions of the existing solution. The Hide-and-Seek algorithm (Romeijn and Smith 1994) is the first SA algorithm for solving optimization problems.

(iii) Tabu Search (TS): The idea of the TS (Glover and Laguna 1997) is similar to SA, in which both traverse the solution space by testing mutations of an individual solution. While SA generates only one mutated solution, TS generates many mutated solutions and moves to the solution with the lowest fitness of those generated.

(iv) Ant Colony Optimization (ACO): ACO first introduced by Marco Dorigo in his PhD thesis and published in Colomi et al. (1992), is a probabilistic technique for solving

computational problems which can be reduced to finding good paths through graphs.

They are inspired by the behaviors of ants in finding paths from the colony to food.

The main challenge of the above meta-heuristic methods is to guarantee convergence in the solution; accordingly, the quality of the solution is not ensured.

Those two categories of optimum approaches have different advantages and weaknesses; consequently, integrating deterministic and heuristic approaches is an appropriate way to solve large scale optimal problems. For example, heuristic algorithms are applied to find an initial solution to enhance the efficiency of finding the optimal solution; then, deterministic approaches are utilized to transcend the incumbent solution.

1.2 Research Motivation and Purpose

Traditional deterministic optimization approaches have been very successful in obtaining global optimization; nevertheless, there are still lacks of efficient method to some nonlinear problems, especially for large mixed-integer problems. Consider the following equation:

$$\sum_{\theta=1}^m u_{\theta} = 1, u_{\theta} \in \{0,1\}. \quad (1.1)$$

It means that there is one and only one binary variable can be active in the all given binary variable. Equation (1.1) is frequently appeared in numerous classic optimal problems, especially in the integer and combinatorial optimal problems. For instance, Task Allocation Problem (TAP) is one of the fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics. The primitive form of TAP is listed in Example 1.2, which first constraint (1.4) is similar as Equation (1.1). Another example is to decompose the separated discrete nonlinear term, introduced by Floudas (2000), in Example 1.1. The constraint (1.3) is same as Equation (1.1). There are also numerous applications of

using Equation (1.1) in optimization problem.

This study focuses on developing a Superior Representation of Binary Variables (SRB) method to replace the traditionally binary variables and apply in the various applications. The major advantages of the proposed approach over existing optimization methods are reducing the traditional binary variables number from $O(m)$ to $O(\log_2 m)$ with $4\lceil \log_2 m \rceil + 1$ new additive constraints and easily applying in any mixed-integer problems. We also applied SRB in some applications, such as Generalized Geometric Programming (GGP) and Task Assignment Problem (TAP), and obtained the global optimization efficiently.

1.3 Examples

EXAMPLE 1.1 Linearization Form of Discrete Nonlinear Function

This example is a linearization result of discrete nonlinear function (1.2)

$$g(y) = y^\alpha, y \in \{d_1, \dots, d_m\}. \quad (1.2)$$

The decomposing program (DP), introduced in Floudas (2000), utilizes m binary variables to linearization (1.2). The mathematical form is listed below.

DP

$$\begin{aligned} \text{Min} \quad & \sum_{\theta=1}^m u_\theta d_\theta^\alpha \\ \text{s.t.} \quad & \sum_{\theta=1}^m u_\theta = 1, u_\theta \in \{0,1\}, \forall \theta, \end{aligned} \quad (1.3)$$

EXAMPLE 1.2 Task Assignment Problem

This example is primitive form of Task Assignment Problem (TAP), which has n tasks that must be accomplished by using only m agents. Each agent performs better at some tasks and worse at others and obviously some agents are better than others at certain tasks.

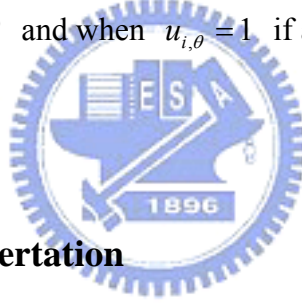
The goal is to minimize the total cost for accomplishing all tasks or, stated differently, to maximize the overall output of all the agents as a whole.

TAP

$$\begin{aligned}
 \text{Min } & \sum_{i=1}^n \sum_{\theta=1}^m d_{i,\theta} u_{i,\theta} \\
 \text{s.t. } & \sum_{\theta=1}^m u_{i,\theta} = 1, \quad i = 1, \dots, n, \\
 & \sum_{i=1}^n a_{i,\theta} u_{i,\theta} \leq b_{\theta}, \quad \theta = 1, \dots, m,
 \end{aligned} \tag{1.4}$$

where

- (i) $d_{i,\theta}$ be the execution cost of task i if it is assigned to agent θ ,
- (ii) b_{θ} be the capacity for agent θ ,
- (iii) $a_{i,\theta}$ be the resource required of task i if it is assigned to agent θ ,
- (vi) $u_{i,\theta} \in \{0,1\} \forall i, \theta$ and when $u_{i,\theta} = 1$ if and only if task i is allocated to agent θ .



1.4 Structure of the Dissertation

The structure of this dissertation is depicted in Figure 1.1 and briefly introduced as follows.

Chapter 2 develops the core technique of this dissertation, Superior Representation of Binary Variables (SRB), which only uses $\lceil \log_2 m \rceil$ binary variables to replace m original traditional binary variables (u_1, \dots, u_m) with $\sum_{\theta=1}^m u_{\theta} = 1$. It can effectively reduce the traditional binary variables number from $O(m)$ to $O(\log_2 m)$ and easily applies in any mixed-integer problems. By using SRB, this chapter proposes a superior way of expressing the same piecewise linear function, where only $\lceil \log_2 m \rceil$ binary variables and $8 + 8\lceil \log_2 m \rceil$ additive constraints are used. Various numerical experiments demonstrate that the proposed method is more computationally efficient than existing methods.

Chapter 3 extends SRB to apply Generalized Geometric Programs Problems. Based on the SRB, this chapter proposes a novel method for handling a GGP problem with free-signed continuous and discrete variables. We first use $\lceil \log_2 r \rceil$ binary variables to express a discrete function containing r discrete values, and utilize convexification and concavification strategies to treat the continuous signomial function. Comparing with existing GGP methods, the proposed method can solve the problem to reach approximate global optimum using less number of binary variables and constraints.

Chapter 4 is another extension of SRB for solving the task allocation problem (TAP). For example, a TAP with n agents and m tasks requires nm binary variables in existing deterministic approaches to obtain global solution while our method, based on SRB, only needs $\lceil \log_2 nm \rceil$ binary variables. We try to find an efficient method to conquer the classic TAP.

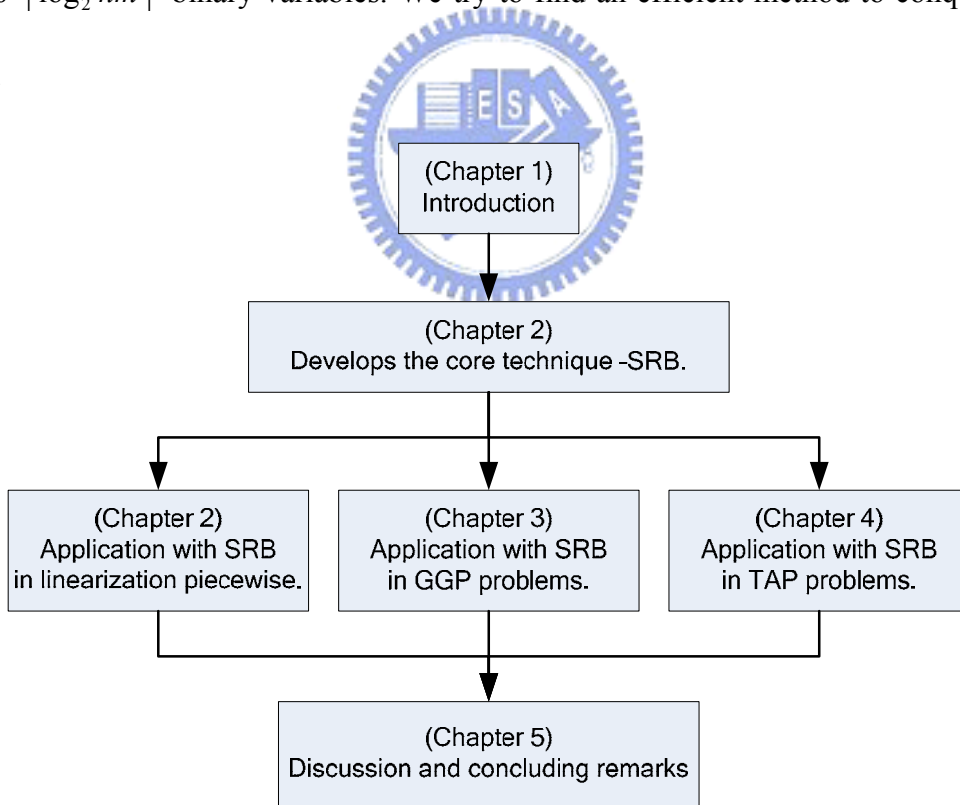


Figure 1.1 Structure of the dissertation

Chapter 2 A Superior Representation Method for Piecewise Linear Functions

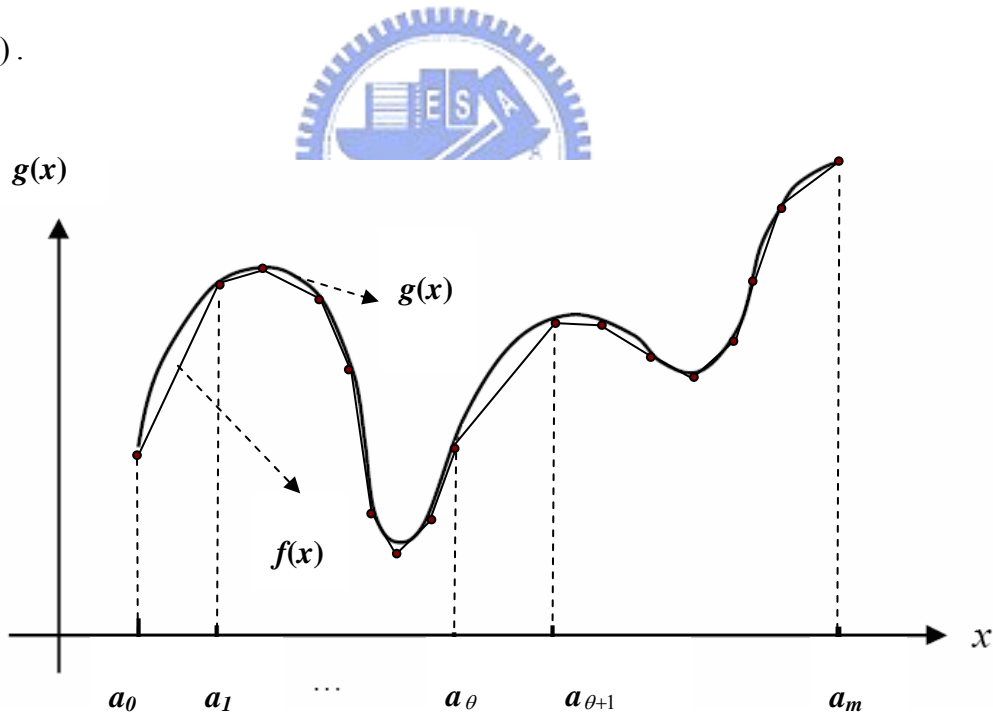
Many nonlinear programs can be linearized approximately as piecewise linear programs by adding extra binary variables. For the last four decades, several techniques of formulating a piecewise linear function have been developed. For expressing a piecewise linear function with $m+1$ break points, existing methods require to use m additional binary variables and $4m$ constraints, which causes heavy computation when m is large. This chapter proposes a superior way of expressing the same piecewise linear function, where only $\lceil \log_2 m \rceil$ binary variables and $8+8\lceil \log_2 m \rceil$ additive constraints are used. Various numerical experiments demonstrate that the proposed method is more computationally efficient than existing methods.

2.1 Introduction to Piecewise Linear Functions

Piecewise linear functions have been applied in numerous optimization problems, such as data fitting, network analysis, logistics, and production planning (Bazaraa et al. 1993). Since 1950, it has been widely recognized that a nonlinear program can be piecewisely linearized as a mixed 0-1 program by introducing extra 0-1 variables (Dantzing 1960). Most textbooks (Bazaraa et al. 1999, Floudas 2000, Vajda 1964) in Operations Research offer some methods of expressing piecewise linear functions. Many methods of piecewisely linearizing general nonlinear functions have also been developed in recent literature (Croxtton et al. 2003, Li 1996, Li and Lu 2008, Padberg 2000, Topaloglu and Powell 2003). Some methods of piecewisely linearizing a specific concave function have also been discussed (Kontogiorgis 2000). For expressing a piecewise linear function $f(x)$ of a single variable x with $m+1$ break points, most of the methods in the textbooks and

literatures mentioned above require to add extra m binary variables and $4m$ constraints, which may cause heavy computational burden when m is large. This study proposes a superior method for representing $f(x)$, which only requires to add extra $\lceil \log_2 m \rceil$ binary variables and $8 + 8\lceil \log_2 m \rceil$ constraints.

Consider a general nonlinear function $g(x)$ of a single variable x , $g(x)$ is continuous and x is within the interval $[a_0, a_m]$. Let $f(x)$ be a piecewise linear function of $g(x)$, where the interval $[a_0, a_m]$ is partitioned into m small intervals via the break points a_0, \dots, a_m , where $a_0 < a_1 < \dots < a_m$ as shown in Figure 2.1. It is assumed in this study that the locations of the break points are pre-specified by the user. This study does not consider how to pick break points that result in small approximation error between $g(x)$ and $f(x)$.



$f(x)$ is a piecewise linear function of $g(x)$

Figure 2.1 Piecewise linear function $f(x)$

As indicated by Croxton et al. (2003), most models of expressing a piecewise linear

function are equivalent with each other. Here, a general form of pressing a piecewise linear function by the existing methods, as in the textbook of Bazaraa et al. (1993), Floudas (2000), and Vajda (1964) or in the articles of Croxton et al. (2003), Li (1996), Li and Lu (2008), Padberg (2000), and Topaloglu and Powell (2003), are formulated below:

First introducing m binary variables $\lambda_0, \lambda_1, \dots, \lambda_{m-1}$ to express the inequalities:

$$a_\theta - (a_m - a_0)(1 - \lambda_\theta) \leq x \leq a_{\theta+1} + (a_m - a_0)(1 - \lambda_\theta), \quad \theta = 0, \dots, m-1, \quad (2.1)$$

where

$$\sum_{\theta=0}^{m-1} \lambda_\theta = 1, \quad \lambda_\theta \in \{0,1\}. \quad (2.2)$$

There is one and only one $\theta = k$ (over $\theta \in \{0, \dots, m-1\}$), in which $\lambda_k = 1$. That means, only one interval $[a_k, a_{k+1}]$ is activated, which results in $a_k \leq x \leq a_{k+1}$.

Following Expression (2.1), the piecewise linear function $f(x)$ can be expressed as:

$$f(a_\theta) + s_\theta(x - a_\theta) - M(1 - \lambda_\theta) \leq f(x) \leq f(a_\theta) + s_\theta(x - a_\theta) + M(1 - \lambda_\theta), \quad \theta = 0, \dots, m-1 \quad (2.3)$$

$$\text{where } M = \max_{\theta} \{f(a_\theta) + s_\theta(a_m - a_\theta), f(a_\theta) + s_\theta(a_0 - a_\theta)\}$$

$$- \min_{\theta} \{f(a_\theta) + s_\theta(a_m - a_\theta), f(a_\theta) + s_\theta(a_0 - a_\theta)\},$$

$$\text{and } s_\theta = \frac{f(a_{\theta+1}) - f(a_\theta)}{a_{\theta+1} - a_\theta}. \quad (2.4)$$

REMARK 2.1 Expressing $f(x)$ by (2.1) through (2.4) requires m binary variables $\lambda_0, \dots, \lambda_{m-1}$ and $4m$ constraints (i.e., inequalities (2.1) – (2.3)), where $m+1$ is the number of break points.

2.2 Superior Representation of Binary Variables (SRB)

Let θ be an integer, $0 \leq \theta \leq m-1$, expressed as:

$$\theta = \sum_{j=1}^h 2^{j-1} u_j, \quad h = \lceil \log_2 m \rceil, \quad u_j \in \{0,1\}. \quad (2.5)$$

Given a $\theta \in \{0, \dots, m-1\}$, there is a unique set of $\{u_j\}$ meets (2.5). Let $G(\theta) \subseteq \{1, \dots, h\}$

be the subset of indices such that:

$$\sum_{j \in G(\theta)} 2^{j-1} = \theta. \quad (2.6)$$

For instance, $G(0) = \emptyset$, $G(1) = \{1\}$, $G(2) = \{2\}$, $G(3) = \{1,2\}$, etc. Let $|G(\theta)|$ denote the number of elements in $G(\theta)$. For instance, $|G(0)| = 0$, $|G(3)| = 2$, etc. Define m functions $A_\theta(\theta')$ ($\theta = 0, \dots, m-1$) based on a binary vector $\mathbf{u}' = (u'_1, \dots, u'_h)$ as follows:

$$A_\theta(\theta') = |G(\theta)| + \sum_{j=1}^h c_{\theta,j} u'_j, \quad \theta' = \sum_{j=1}^h 2^{j-1} u'_j. \quad (2.7)$$

where the coefficients $c_{\theta,j}$, $\theta = 0, \dots, m-1$, $j = 1, \dots, h$ are defined by

$$c_{\theta,j} = \begin{cases} -1, & j \in G(\theta), \\ 1, & \text{otherwise.} \end{cases} \quad (2.8)$$

Table 2.1 List of $|G(\theta)|$, $c_{\theta,j}$, and $A_\theta(\theta')$

θ	$ G(\theta) $	$c_{\theta,1}$	$c_{\theta,2}$	$c_{\theta,3}$	$c_{\theta,4}$	$A_\theta(\theta')$
0	0	1	1	1	1	$0 + u'_1 + u'_2 + u'_3 + u'_4$
1	1	-1	1	1	1	$1 - u'_1 + u'_2 + u'_3 + u'_4$
2	1	1	-1	1	1	$1 + u'_1 - u'_2 + u'_3 + u'_4$
3	2	-1	-1	1	1	$2 - u'_1 - u'_2 + u'_3 + u'_4$
4	1	1	1	-1	1	$1 + u'_1 + u'_2 - u'_3 + u'_4$
5	2	-1	1	-1	1	$2 - u'_1 + u'_2 - u'_3 + u'_4$
6	2	1	-1	-1	1	$2 + u'_1 - u'_2 - u'_3 + u'_4$
7	3	-1	-1	-1	1	$3 - u'_1 - u'_2 - u'_3 + u'_4$
8	1	1	1	1	-1	$1 + u'_1 + u'_2 + u'_3 - u'_4$
9	2	-1	1	1	-1	$2 - u'_1 + u'_2 + u'_3 - u'_4$
10	2	1	-1	1	-1	$2 + u'_1 - u'_2 + u'_3 - u'_4$
11	3	-1	-1	1	-1	$3 - u'_1 - u'_2 + u'_3 - u'_4$
12	2	1	1	-1	-1	$2 + u'_1 + u'_2 - u'_3 - u'_4$

For instance, given an arbitrary number $m = 13$, a list of all values of $|G(\theta)|$, $c_{\theta,j}$, and $A_\theta(\theta')$ for given θ are shown in Table 2.1. $|G(\theta)|$ and $c_{\theta,j}$ in (2.7) are not intuitive forms, then it can be rewritten more intuitive as follows.

PROPOSITION 2.1 $|G(\theta)|$ and $c_{\theta,j}$ in (2.7) can be rewritten as follows:

$$|G(\theta)| = 0.5 \sum_{j=1}^h (1 - (-1)^{\lfloor \frac{\theta}{2^{j-1}} \rfloor}), \theta = 0, \dots, m-1, h = \lceil \log_2 m \rceil, \quad (2.9)$$

$$c_{\theta,j} = (-1)^{\lfloor \frac{\theta}{2^{j-1}} \rfloor}, \theta = 0, \dots, m-1, j = 1, \dots, \lceil \log_2 m \rceil. \quad (2.10)$$

PROOF In (2.7), where $c_{\theta,j} = -1$ if $j \in G(\theta)$, otherwise $c_{\theta,j} = 1$.

Obviously:

$$\begin{aligned} c_{\theta,1} &= (-1)^{\lfloor \frac{\theta}{1} \rfloor} = (-1)^{\lfloor \frac{\theta}{2^0} \rfloor} \\ c_{\theta,2} &= (-1)^{\lfloor \frac{\theta}{2} \rfloor} = (-1)^{\lfloor \frac{\theta}{2^1} \rfloor} \\ &\vdots \\ c_{\theta,h} &= (-1)^{\lfloor \frac{\theta}{2^{h-1}} \rfloor} \end{aligned}$$

$$|G(\theta)| = \frac{(1 - c_{\theta,1}) + (1 - c_{\theta,2}) + \dots + (1 - c_{\theta,h})}{2} = 0.5 \sum_{j=1}^h (1 - (-1)^{\lfloor \frac{\theta}{2^{j-1}} \rfloor})$$

This proposition is then proven. □

Then we have following main theorem:

THEOREM 2.1 For an integer θ in (2.5) and (2.6), given a binary vector $\mathbf{u}' = (u'_1, \dots, u'_h)$, there is one and only one k , $0 \leq k \leq m-1$, such that

(i) $A_k(\theta') = 0$ if $\theta' = k$,

(ii) $A_k(\theta') \geq 1$, otherwise.

PROOF There is one and only one k , such that $k = \sum_{j=1}^h 2^{j-1} u_j + 1$.

(i) If $k = \theta'$ that $u'_j = u_j \forall j$, which results in

$$\sum_{j \in G(k)} u'_j = \sum_{j \in G(k)} u_j = G(k) \text{ and } \sum_{j \notin G(k)} u'_j = \sum_{j \notin G(k)} u_j = 0.$$

Thus to have $A_k(\theta') = 0$.

(ii) If $k \neq \theta'$ then

$$|G(\theta)| \geq \sum_{j \in G(k)} u'_j \text{ and } \sum_{j \notin G(k)} u'_j \geq 1,$$

therefore $A_\theta(\theta') \geq 1$. □

For instance, given $\mathbf{u}' = (u'_1, \dots, u'_h) = (0, 1, 0, 1)$ in Table 2.1, we have

$$A_0(\theta') = |G(0)| + u'_1 + u'_2 + u'_3 + u'_4 = 0 + 0 + 1 + 0 + 1 = 2,$$

$$A_1(\theta') = |G(1)| - u'_1 + u'_2 + u'_3 + u'_4 = 1 - 0 + 1 + 0 + 1 = 3,$$

⋮

$$A_{10}(\theta') = |G(10)| + u'_1 - u'_2 + u'_3 - u'_4 = 2 + 0 - 1 + 0 - 1 = 0,$$

$$A_{11}(\theta') = |G(11)| - u'_1 - u'_2 + u'_3 - u'_4 = 3 - 0 - 1 + 0 - 1 = 1,$$

$$A_{12}(\theta') = |G(12)| + u'_1 + u'_2 - u'_3 - u'_4 = 2 + 0 + 1 - 0 - 1 = 2,$$

Here $A_{10}(\theta') = 0$; $A_\theta(\theta') \geq 0$ for $0 \leq \theta \leq 12$ and $\theta \neq 10$.

PROPOSITION 2.2 For a set of non-negative continuous variables (r_0, \dots, r_{m-1}) , if

$$\sum_{\theta=0}^{m-1} r_\theta = 1, \tag{2.11}$$

$$\sum_{\theta=0}^{m-1} r_{\theta} A_{\theta}(\theta') = 0, \quad (2.12)$$

then the values of $r_{\theta}, \theta = 0, \dots, m-1$, are either 0 or 1, where $A_{\theta}(\theta')$ are defined by (2.6) – (2.8).

PROOF By referring to THEOREM 2.1, there is one and only one $k, 0 \leq k \leq m-1$, such that $A_k(\theta') = 0$ and $A_{\theta}(\theta') \geq 1$ for $0 \leq \theta \leq m-1$ and $\theta \neq k$. Since $r_{\theta} \geq 0$, $0 \leq \theta \leq m-1$, the only solution to satisfy (2.11) and (2.12) is $r_k = 1$ and $r_{\theta} = 0$ for $\theta \neq k$. The proposition is then proven. \square

Expression (2.12) is nonlinear and needs to be linearized. We have

$$\begin{aligned} & \sum_{\theta=0}^{m-1} r_{\theta} A_{\theta}(\theta') \\ &= r_0 (|G(0)| + c_{0,1} u'_1 + \dots + c_{0,h} u'_h) + \dots \\ & \quad + r_{m-1} (|G(m-1)| + c_{m-1,1} u'_1 + \dots + c_{m-1,h} u'_h) \\ &= \sum_{\theta=0}^{m-1} r_{\theta} |G(\theta)| + u'_1 \sum_{\theta=0}^{m-1} r_{\theta} c_{\theta,1} + \dots + u'_h \sum_{\theta=0}^{m-1} r_{\theta} c_{\theta,h} = 0 \end{aligned} \quad (2.13)$$

Denote

$$z_j = u'_j \sum_{\theta=0}^{m-1} r_{\theta} c_{\theta,j}, \quad \forall j \quad (2.14)$$

We then have following proposition:

PROPOSITION 2.3 *The nonlinear expression $\sum_{\theta=0}^{m-1} r_{\theta} A_{\theta}(\theta') = 0$ in (2.12) can be replaced*

by the following linear system:

$$\sum_{\theta=0}^{m-1} r_{\theta} |G(\theta)| + \sum_{j=1}^h z_j = 0, \quad (2.15)$$

$$-u'_j \leq z_j \leq u'_j, \quad u'_j \in \{0,1\}, \quad j=1, \dots, h, \quad (2.16)$$

$$\sum_{\theta=0}^{m-1} r_\theta c_{\theta,j} - (1-u'_j) \leq z_j \leq \sum_{\theta=0}^{m-1} r_\theta c_{\theta,j} + (1-u'_j), \quad j=1, \dots, h. \quad (2.17)$$

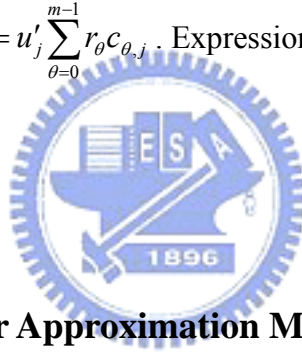
PROOF By referring to (2.14), (2.13) can be converted into (2.15). According to (2.8) and PROPOSITION 2.2 we have $c_{\theta,j} \in \{1,-1\}$ and $r_\theta \in \{0,1\}$, which cause $-1 \leq z_j \leq 1$. Expression (2.14) therefore can be linearized by (2.16) and (2.17). Two cases can be check for each u'_j :

(i) If $u'_j = 0$, then $z_j = 0$ referring to (2.16),

(ii) If $u'_j = 1$, then $z_j = \sum_{\theta=0}^{m-1} r_\theta c_{\theta,j}$ referring to (2.17).

Both (i) and (ii) imply that $z_j = u'_j \sum_{\theta=0}^{m-1} r_\theta c_{\theta,j}$. Expression (2.14) can be replaced by

(2.15)–(2.17). □



2.3 The Proposed Linear Approximation Method

Consider the same piecewise linear function $f(x)$ described in Figure 2.1, where x is within the interval $[a_0, a_m]$, and there are $m+1$ break points $a_0 < a_1 < \dots < a_m$ within $[a_0, a_m]$. By referring to PROPOSITION 2.2 and PROPOSITION 2.3, $f(x)$ with division of the interval $[a_0, a_m]$ into m subintervals can be expressed as:

$$\sum_{\theta=0}^{m-1} r_\theta a_\theta \leq x \leq \sum_{\theta=0}^{m-1} r_\theta a_{\theta+1}, \quad (2.18)$$

$$\begin{aligned} f(x) &= \sum_{\theta=0}^{m-1} r_\theta (f(a_\theta) + s_\theta (x - a_0 + a_0 - a_\theta)) \\ &= \sum_{\theta=0}^{m-1} (f(a_\theta) - s_\theta (a_\theta - a_0)) r_\theta + \sum_{\theta=0}^{m-1} s_\theta r_\theta (x - a_0) \end{aligned} \quad (2.19)$$

where s_θ are slopes of $f(x)$ as defined in (2.4).

By referring to PROPOSITION 2.2, $r_\theta \in \{0,1\}$ and $\sum_{\theta} r_\theta = 1$, Expressions (2.18) and

(2.19) imply that: if $r_k = 1, k \in \{1, \dots, m-1\}$, then $a_k \leq x \leq a_{k+1}$ and

$f(x) = f(a_k) + s_k(x - a_k)$. The nonlinear term $\sum_{\theta=1}^{m-1} a_\theta r_\theta (x - a_1)$ in (2.19) can then be

linearized as follows.

Denote $w_\theta = r_\theta(x - a_0)$, since there is one and only one $k, k \in \{0, \dots, m-1\}$ such that $r_k = 1$ and $A_k(\theta') = 0$, it is clear that:

$$\sum_{\theta=0}^{m-1} w_\theta = x - a_0, \quad (2.20)$$

$$\sum_{\theta=0}^{m-1} w_\theta A_\theta(\theta') = 0. \quad (2.21)$$

By referring to (2.15), we then have $\sum_{\theta=0}^{m-1} w_\theta A_\theta(\theta') = \sum_{\theta=0}^{m-1} w_\theta |G(\theta)| + \sum_{j=1}^h u'_j \left(\sum_{\theta=0}^{m-1} w_\theta c_{\theta,j} \right) = 0$.

Denote $\delta_j = u'_j \sum_{\theta=0}^{m-1} w_\theta c_{\theta,j}$ for all j , the term $\sum_{\theta=0}^{m-1} w_\theta A_\theta(\theta')$ can be expressed as below

referring to PROPOSITION 2.2:

$$\sum_{\theta=0}^{m-1} w_\theta |G(\theta)| + \sum_{j=1}^h \delta_j = 0, \quad (2.22)$$

$$-(a_m - a_0)u'_j \leq \delta_j \leq (a_m - a_0)u'_j, \quad j = 1, \dots, h, \quad (2.23)$$

$$\sum_{\theta=0}^{m-1} w_\theta c_{\theta,j} - (a_m - a_0)(1 - u'_j) \leq \delta_j \leq \sum_{\theta=0}^{m-1} w_\theta c_{\theta,j} + (a_m - a_0)(1 - u'_j), \quad j = 1, \dots, h. \quad (2.24)$$

We then deduce our main result:

THEOREM 2.2 *Let $f(x)$ be a piecewise linear function defined in a given interval $x \in [a_0, a_m]$ with m subintervals by $m+1$ break points: $a_0 < a_1 < \dots < a_m$. $f(x)$ can be expressed by the following linear system:*

$$\sum_{\theta=0}^{m-1} r_{\theta} a_{\theta} \leq x \leq \sum_{\theta=0}^{m-1} r_{\theta} a_{\theta+1}, \quad (2.25)$$

$$f(x) = \sum_{\theta=0}^{m-1} (f(a_{\theta}) - s_{\theta}(a_{\theta} - a_0)) r_{\theta} + \sum_{\theta=0}^{m-1} s_{\theta} w_{\theta}, \quad (2.26)$$

$$\sum_{\theta=0}^{m-1} r_{\theta} = 1, r_{\theta} \geq 0, \quad (2.27)$$

$$\sum_{\theta=0}^{m-1} r_{\theta} |G(\theta)| + \sum_{j=1}^h z_j = 0, \quad (2.28)$$

$$-u'_j \leq z_j \leq u'_j, \quad j = 1, \dots, h, \quad (2.29)$$

$$\sum_{\theta=0}^{m-1} r_{\theta} c_{\theta,j} - (1 - u'_j) \leq z_j \leq \sum_{\theta=0}^{m-1} r_{\theta} c_{\theta,j} + (1 - u'_j), \quad j = 1, \dots, h, \quad (2.30)$$

$$\sum_{\theta=0}^{m-1} w_{\theta} = x - a_0, \quad (2.31)$$

$$\sum_{\theta=0}^{m-1} w_{\theta} |G(\theta)| + \sum_{j=1}^h \delta_j = 0, \quad (2.32)$$

$$-(a_m - a_0) u'_j \leq \delta_j \leq (a_m - a_0) u'_j, \quad j = 1, \dots, h, \quad (2.33)$$

$$\sum_{\theta=0}^{m-1} w_{\theta} c_{\theta,j} - (a_m - a_0)(1 - u'_j) \leq \delta_j \leq \sum_{\theta=0}^{m-1} w_{\theta} c_{\theta,j} + (a_m - a_0)(1 - u'_j), \quad j = 1, \dots, h, \quad (2.34)$$

$$\sum_{j=1}^h 2^{j-1} u'_j \leq m. \quad (2.35)$$

where (2.25) and (2.26) come from (2.18) and (2.19), (2.27) from (2.11), (2.28)–(2.30) from PROPOSITION 2.3, and (2.31)–(2.34) from (2.21)–(2.24).

REMARK 2.2 Expressing $f(x)$ with $m+1$ break points by THEOREM 2.2 uses $\lceil \log_2 m \rceil$ binary variables, $8+8\lceil \log_2 m \rceil$ constraints, $2m$ non-negative continuous variables, and $2\lceil \log_2 m \rceil$ free-signed continuous variables.

Comparing REMARK 2.1 with REMARK 2.2 to know that, in expressing $f(x)$ with large number of break points, the proposed method uses much less number of binary variables and constraints than being used in the existing methods.

2.4 Numerical Examples

EXAMPLE 2.1

Consider the following nonlinear programming problem:

P1

$$\text{Min } x_1^{\alpha_1} - x_2^{\beta_1}$$

$$\text{s.t. } x_1^{\alpha_2} - 6x_1 + x_2^{\beta_2} \leq b,$$

$$x_1 + x_2 \leq 8,$$

$$1 \leq x_1 \leq 7.4, \quad 1 \leq x_2 \leq 7.4,$$

where $\alpha_1, \beta_1, \alpha_2, \beta_2$ and b are fixed constants.

Various $\alpha_1, \beta_1, \alpha_2, \beta_2$ and b values are specified in this experiment to compare the computational efficiency of expressing the piecewise linear functions by the existing method (i.e., (2.1)–(2.4)) and by our proposed method (i.e., THEOREM 2.2).

For the first experiment given, $\alpha_1 = 0.4$, $\beta_1 = 2$, $\alpha_2 = 1.85$, $\beta_2 = 2$, $b = 5$, the nonlinear terms $x_1^{0.4}$ and $-x_2^2$ are concave and required to be linearized, while the other convex terms $x_1^{1.85}$ and x_2^2 do not need linearization. Suppose $m+1$ equal-distance break points are chosen to linearize each of $x_1^{0.4}$ and $-x_2^2$, denoted as $a_0 = 1$, $a_\theta = 1 + 6.4\theta/m$, $a_m = 7.4$, for $\theta = 1, \dots, m-1$. Then existing methods (i.e., (2.1)–(2.4)) require to use $2m$ binary variables and $8m$ extra constraints to piecewisely linearize $x_1^{0.4}$ and $-x_2^2$. However, by piecewisely linearize the same convex terms, our proposed method (i.e., THEOREM 2.2)

only uses $2\lceil \log_2 m \rceil$ binary variables, $14 + 16\lceil \log_2 m \rceil$ extra constraints, $4m$ non-negative continuous variables, and $2\lceil \log_2 m \rceil$ free-signed continuous variables. The linearization of $x_1^{0.4}$ by the proposed method is described below:

Denote $f(x_1) = x_1^{0.4}$. By THEOREM 2.2, $f(x_1)$ with $m = 64$ is expressed by the following linear equations and inequalities:

$$\sum_{\theta=0}^{63} r_{\theta} a_{\theta} \leq x_1 \leq \sum_{\theta=0}^{63} r_{\theta} a_{\theta+1},$$

$$f(x_1) = \sum_{\theta=0}^{63} (a_{\theta}^{0.4} - s_{\theta}(a_{\theta} - a_0)) r_{\theta} + \sum_{\theta=0}^{63} s_{\theta} w_{\theta},$$

$$\sum_{\theta=0}^{63} r_{\theta} = 1, r_{\theta} \geq 0,$$

$$\sum_{\theta=0}^{63} w_{\theta} = x - 1,$$

$$\sum_{\theta=0}^{63} r_{\theta} |G(\theta)| + \sum_{j=1}^6 z_j = 0,$$

$$-u'_j \leq z_j \leq u'_j, j = 1, \dots, 6,$$

$$\sum_{\theta=0}^{63} r_{\theta} c_{\theta,j} - (1 - u'_j) \leq z_j \leq \sum_{\theta=0}^{63} r_{\theta} c_{\theta,j} + (1 - u'_j), j = 1, \dots, 6,$$

$$\sum_{\theta=0}^{63} w_{\theta} |G(\theta)| + \sum_{j=1}^6 \delta_j = 0,$$

$$-6.4u'_j \leq \delta_j \leq 6.4u'_j, j = 1, \dots, 6,$$

$$\sum_{\theta=0}^{63} w_{\theta} c_{\theta,j} - 6.4(1 - u'_j) \leq \delta_j \leq \sum_{\theta=0}^{63} w_{\theta} c_{\theta,j} + 6.4(1 - u'_j), j = 1, \dots, 6,$$

where $s_{\theta} = \frac{a_{\theta+1}^{0.4} - a_{\theta}^{0.4}}{a_{\theta+1} - a_{\theta}}$, $u'_j \in \{0,1\}$, $\forall j$.

Similarly, $-x_2^2$ can be piecewisely linearized by THEOREM 2.2. The original program

is then converted into a mixed 0-1 convex program. We solve this program on LINGO (2004) by both the existing methods and the proposed method. The related solutions, CPU time, number of iterations, number of binary variables, number of constraints, and gap to the upper and lower bound objective value are listed in Table 2.2.

Table 2.2 Experiment 1 ($\alpha_1 = 0.4$, $\beta_1 = 2$, $\alpha_2 = 1.85$, $\beta_2 = 2$, $b = 5$)

$m + 1$	Item	Proposed method	Existing method	Gap to the upper and lower bound objective value
65	CPU Time (second)	28	57	
	Number of iterations	103,090	229,521	
	Number of binary variables	12	128	
	Number of constraints	110	514	
	Solution (x_1, x_2)	(3.849203, 3.998877)		
	Objective Value	-14.27649		0.00001
257	CPU Time (second)	329	4298	
	Number of iterations	452,721	5,153,692	
	Number of binary variables	16	512	
	Number of constraints	142	2050	
	Solution (x_1, x_2)	(3.852642, 3.998955)		
	Objective Value	-14.27648		<0.00001

* The reference upper bound solution is $(x_1, x_2) = (3.852642, 3.998955)$ with objective value -14.27648

The gaps between the upper and lower bounds can be computed as follows.

- (i) Solve the nonlinear program **P1** directly by a nonlinear optimization software (such as LINGO). The solution is $\mathbf{x}^0 = (x_1^0, x_2^0) = (3.852642, 3.998955)$ with objective value -14.27648 . \mathbf{x}^0 is one of the local optima of **P1**. But it is unknown about the gap between \mathbf{x}^0 and the global optimum. \mathbf{x}^0 is regarded as the upper bound solution.
- (ii) Divide the ranges of both x_1 and x_2 into 65 break points. Let (x_1^0, x_2^0) replace the nearest break point (3.8, 4.0) in the existing 65 break points. Solving the program by both the proposed method and the existing method to obtain

$\mathbf{x}^\Delta = (3.849203, 3.998877)$ with objective value -14.27649 . \mathbf{x}^Δ is a lower bound solution of **P1**. The gap between the objective values of \mathbf{x}^0 and \mathbf{x}^Δ is 0.00001.

(iii) Reiterate to (ii), divide the ranges of both x_1 and x_2 into 257 break points. Solving the program by both methods to obtain the solution (3.852642, 3.998955), which is the same as \mathbf{x}^Δ .

Various combinations of α_1 , β_1 , α_2 , β_2 and b are specified to test the computation results of the existing methods and the proposed method. Parts of the results are displayed in Table 2.3. All these experiments demonstrate that the proposed method is much faster than the existing method especially when m becomes large.



EXAMPLE 2.2

This example, modified from Dixon and Szegő (1975), is to find the approximated global optimum of the following program:

P2

$$\text{Min } 2x_1^2 - 1.081x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2 + 0.01x_1$$

$$\text{s.t. } -2 \leq x_i \leq 2, \quad i=1,2,$$

where the object function is a non-symmetrical three-humped camelback function. Figure 2.2 shows a contour map of this function under $-2 \leq x_1 \leq 2$ and $-2 \leq x_2 \leq 2$. Here a reference upper bound solution is $(x_1, x_2) = (-1.802271, -0.9011357)$ with objective value -0.02723789 . For solving this program, let $x'_1 = x_1 + 2.1 > 0$ and $x'_2 = x_2 + 2.1 > 0$ to have $x_1x_2 = x'_1x'_2 - 2.1x'_1 - 2.1x'_2 + 4.41$. Replace $x'_1x'_2$ by $\exp(y)$, **P2** then becomes **P3** below:

Table 2.3 Experiment 2 (for different combinations of α_1 , β_1 , α_2 , β_2 and b)

$\alpha_1, \beta_1, \alpha_2,$ β_2, b	$m+1$	Item	Proposed method	Existing method	Gap to the upper and lower bound objective value
A reference upper bound solution is $(x_1, x_2)=(2.289561, 5.710439)$ with objective value -0.8765232					
$\alpha_1 = 0.5,$ $\beta_1 = 0.5,$ $\alpha_2 = 0.8,$ $\beta_2 = 0.9,$	65	CPU Time (second)	5	13	
		Number of iterations	15,616	84,036	
		Solution (x_1, x_2)	(2.289561, 5.710439)		
		Objective Value	-0.8765229		0.0000003
	129	CPU Time (second)	12	54	
		Number of iterations	43,830	216,577	
		Solution (x_1, x_2)	(2.289561, 5.710439)		
		Objective Value	-0.876523		0.0000002
	257	CPU Time (second)	69	464	
		Number of iterations	113,591	1,296,464	
		Solution (x_1, x_2)	(2.28956, 5.71044)		
		Objective Value	-0.8765232		<0.0000001
A reference upper bound solution is $(x_1, x_2)=(4.087383, 3.8)$ with objective value -12.68378					
$\alpha_1 = 0.4,$ $\beta_1 = 2,$ $\alpha_2 = 0.8,$ $\beta_2 = 2,$ $b = -7$	65	CPU Time (second)	15	17	
		Number of iterations	68,759	141,519	
		Solution (x_1, x_2)	(4.153399, 3.846601)		
		Objective Value	-13.03136		0.34758
	129	CPU Time (second)	62	428	
		Number of iterations	207,894	1,243,751	
		Solution (x_1, x_2)	(4.153402, 3.846598)		
		Objective Value	-13.02896		0.34518
	257	CPU Time (second)	299	1350	
		Number of iterations	557,043	2,490,405	
		Solution (x_1, x_2)	(4.153401, 3.846599)		
		Objective Value	-13.02889		0.34511

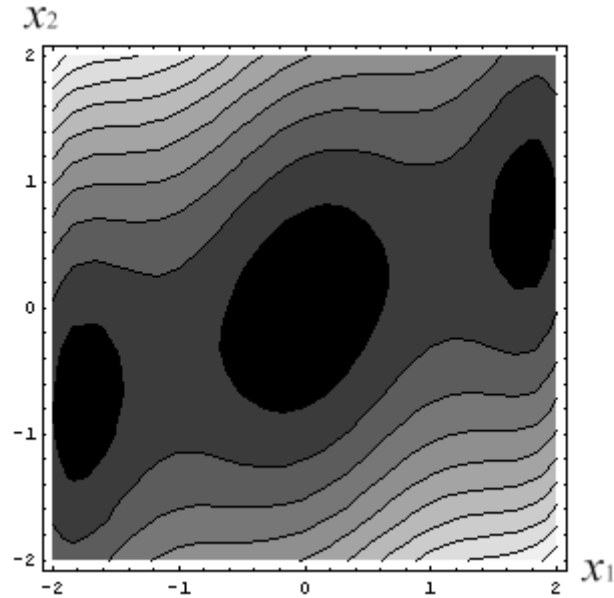


Figure 2.2 Contour map of the three-humped camelback function

P3

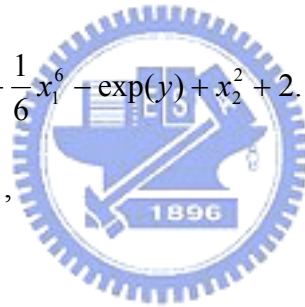
$$\text{Min } 2x_1^2 - 1.081x_1^4 + \frac{1}{6}x_1^6 - \exp(y) + x_2^2 + 2.11x_1 + 2.1x_2 + 4.41$$

$$\text{s.t. } y = \ln x'_1 + \ln x'_2,$$

$$x_1 = x'_1 - 2.1,$$

$$x_2 = x'_2 - 2.1,$$

$$-2 \leq x_i \leq 2, \quad i = 1, 2.$$



The terms $-x_1^4$, $-\exp(y)$, $\ln x'_1$ and $\ln x'_2$ in **P3** are non-convex, here we use $-z_1$, $-z_2$, y_1 and y_2 to respectively as piecewise linear function to approximate those non-convex terms. We then obtain the following mixed-integer program:

P4

$$\text{Min } 2x_1^2 - 1.081z_1 + \frac{1}{6}x_1^6 - z_2 + x_2^2 + 2.11x_1 + 2.1x_2 + 4.41$$

s.t. The same constraints as in P3, but replace $\ln x'_1$ and $\ln x'_2$ by y_1 and y_2 respectively.

We take 41 equal-distance break points for the value range of x_1 and x_2 ($[-2, 2]$), denoted as $a_0 = -2, a_1 = -1.9, a_2 = -1.8, \dots, a_{40} = 2$. It is clear that the break points for x'_1 and x'_2 are $(0.1, 0.2, \dots, 4.1)$. Base on THEOREM 2.2, each of $z_1, y_1,$ and y_2 is linearized with 6 binary variables and 55 constrains. Since $y = y_1 + y_2$, the region of y is $[\ln 0.1 + \ln 0.1, \ln 4.1 + \ln 4.1] = [\ln 0.01, \ln 16.81]$. We take $m+1$ equal-distance break points for the value range of y , denoted as $b_0 = \ln 0.01, b_\theta = \ln(0.01 + 16.8\theta/m), b_m = \ln 16.81$, for $\theta = 1, 2, \dots, m-1$. $z_2 = e^y$ can then be piecewisely linearized under various y values.

Problem **P4** is converted into a mixed 0-1 convex program. We solve this program using LINGO (2004) with the existing methods and the proposed method. The related solutions, CPU time, number of iterations, number of binary variables, number of constraints, and gaps to the upper and lower bound objective values are reported in Table 2.4.

Table 2.4 Experiment result of Example 2.2

$m+1$	Item	Proposed method	Existing method	Gap to the upper and lower bound objective value
33	CPU Time (second)	239	834	
	Number of iterations	2,387,868	3,675,444	
	Number of binary variables	17	112	
	Number of constraints	161	532	
	Solution (x_1, x_2)	(-1.849207, -0.9925797)		
	Objective Value	-0.2269442		0.19970631
65	CPU Time (second)	882	2,277	
	Number of iterations	1,610,422	7,467,149	
	Number of binary variables	18	144	
	Number of constraints	169	660	
	Solution (x_1, x_2)	(-1.7657, -0.8942029)		
	Objective Value	-0.08396288		0.05672499

* The reference upper bound solution is $(x_1, x_2) = (-1.802271, -0.9011357)$ with objective value -0.02723789

EXAMPLE 2.3

This example is used to illustrate the process of solving discrete functions, consider following program:

$$\begin{aligned} \text{Min } & x_1^3 - 1.8x_1^{2.8} + 0.8x_2^{2.2} - x_2^{2.1} + x_3^{0.5} - 3.5x_4^{0.8} - 0.3x_5^{1.1} \\ \text{s.t. } & x_1^{1.2} + x_2^{0.8} \leq 8, \\ & x_1^{1.2} - x_3^{1.7} \leq 2, \\ & x_2^{2.1} - x_4^{1.7} \geq 4.5, \\ & x_4^{0.8} - x_5^{0.96} \geq -3, \\ & x_2^{2.2} - x_5^{1.1} \geq -0.1, \end{aligned}$$

where x_i are discrete variables for all i , $x_i \in \{1, 1+c, \dots, 1+mc\} = \{d_0, \dots, d_m\}$, where c is the discreteness of x_i , and $m+1$ is the number of discrete points.

The existing method solves this example by linearizing a nonlinear function $x^\alpha, x \in \{d_0, \dots, d_m\}$ as

$$x^\alpha = d_0^\alpha + \sum_{\theta=1}^m \lambda_\theta (d_\theta^\alpha - d_0^\alpha),$$

$$\sum_{\theta=1}^m \lambda_\theta \leq 1, \lambda_\theta \in \{0,1\},$$

where m binary variables are used.

THEOREM 2.2 can be applied to solve a discrete function $x^\alpha, x \in \{d_0, \dots, d_m\}$ where $f(x)$ in (2.26) is replaced by x^α as

$$x^\alpha = \sum_{\theta=0}^m d_\theta^\alpha r_\theta,$$

and all other constraints in (2.27)–(2.30) and (2.35) are kept as the same with

$\theta = 0, \dots, m$. For treating a discrete function, the proposed method uses $\log_2(m+1)$ binary variables, $4 + 4\lceil \log_2(m+1) \rceil$ constraints, $m+1$ non-negative continuous variables, and $\lceil \log_2(m+1) \rceil$ free-signed continuous variables.

Table 2.5 is the computational result with ILOG CPLEX 9.0. It illustrates that the proposed method needs to use more constraints than the existing method; however, the proposed method is computationally more efficient than existing method for large m .

Table 2.5 Experiment results with discrete function in ILOG CPLEX

$(m+1)/d$	Item	Proposed method	Existing method
256 0.025	CPU Time (second)	3.89	1.27
	Number of iterations	17,181	2,935
	Number of binary variables	40	1,275
	Number of constraints	187	22
	Solution $(x_1, x_2, x_3, x_4, x_5)$	(3.725, 4.2, 1.85, 5.075, 7.2)	
	Objective Value	-35.49859275	
512 0.0125	CPU Time (second)	3.58	7.85
	Number of iterations	12,418	14,103
	Number of binary variables	45	2,555
	Number of constraints	207	22
	Solution $(x_1, x_2, x_3, x_4, x_5)$	(3.75, 4.1375, 1.875, 4.9625, 7.1375)	
	Objective Value	-35.51962643	
1024 0.00625	CPU Time (second)	7.45	66.92
	Number of iterations	17,671	60,603
	Number of binary variables	50	5,115
	Number of constraints	227	22
	Solution $(x_1, x_2, x_3, x_4, x_5)$	(3.6688, 4.35, 1.8188, 5.3688, 7.3937)	
	Objective Value	-35.55043719	

EXAMPLE 2.4

This example uses the same program in Example 2.3 but specifying all x_i as continuous variables, where $1 \leq x_i \leq 7.4$ with $m+1$ break points for all i . The comparison of the proposed method and the existing method, solving with ILOG CPLEX 9.0, is shown in Table 5. It demonstrates that the proposed method is also superior to the existing method for the continuous case.

Table 2.6 Experiment results with continuous function in ILOG CPLEX

$m+1$	Item	Proposed method	Existing method	Gap to the upper and lower bound objective value
33	CPU Time (second)	4.23	39.86	
	Number of iterations	47,896	155,309	
	Number of binary variables	25	160	
	Number of constraints	347	1,098	
	Solution (x_1, x_2)	(3.67117699, 4.34398332)		
	Solution (x_3, x_4, x_5)	(1.8176404, 5.35913745, 7.4)		
	Objective Value	-35.56181842		0.00766842
65	CPU Time (second)	40.29	919.22	
	Number of iterations	250,747	1,437,056	
	Number of binary variables	30	320	
	Number of constraints	407	2,186	
	Solution (x_1, x_2)	(3.67115433, 4.34402387)		
	Solution (x_3, x_4, x_5)	(1.81765758, 5.3591112, 7.4)		
	Objective Value	-35.56160283		0.00745283
129	CPU Time (second)	454.21	4354.26	
	Number of iterations	2,128,203	8,690,141	
	Number of binary variables	35	640	
	Number of constraints	467	4,362	
	Solution (x_1, x_2)	(3.67116282, 4.34402387)		
	Solution (x_3, x_4, x_5)	(1.81767335, 5.3591112, 7.4)		
	Objective Value	-35.56112235		0.00697235

* The reference upper bound solution is $x_1 = 3.668$, $x_2 = 4.352$, $x_3 = 1.816$, $x_4 = 5.373$, $x_5 = 7.4$ with objective value -35.55415

Chapter 3 Extension 1 – Solving Generalized Geometric Programs Problems

Many optimization problems are formulated as Generalized Geometric Programming (GGP) containing signomial terms $f(\mathbf{x}) \cdot g(\mathbf{y})$ where \mathbf{x} and \mathbf{y} are continuous and discrete free-sign vectors respectively. By effectively convexifying $f(\mathbf{x})$ and linearizing $g(\mathbf{y})$, this chapter globally solves a GGP with less number of binary variables than are used in existing GGP methods. Numerical experiments demonstrate the computational efficiency of the proposed method.

3.1 Introduction to Generalized Geometric Programming

Generalized Geometric Programming (GGP) methods have been applied to solve problems in various fields, such as heat exchanger network design (Duffin and Peterson 1966), capital investment (Hellinckx and Rijckaert 1971), optimal design of cooling towers (Ecker and Wiebking 1978), batch plant modeling (Salomone and Iribarren 1992), competence sets expansion (Li 1999), smoothing splines (Cheng et al. 2005), and digital circuit (Boyd 2005). These GGP problems often contain continuous and discrete functions where the discrete variables may represent the sizes of components, thicknesses of steel plates, diameters of pipes, lengths of springs, and elements in a competence set etc. Many local optimization algorithms for solving GGP problems have been developed, which include linearization method (Duffin 1970), separable programming (Kochenberger et al. 1973), and a concave simplex algorithm (Beck and Ecker 1975). Pardalos and Romeijn (2002) provided an impressive overview of these GGP algorithms. Among existing GGP algorithms, the techniques developed by Maranas and Floudas (1997), Floudas et al. (1999), Floudas (2000) (these three methods are called Floudas's methods in this study) are the most popular approaches for solving GGP problems. Floudas's methods, however, can not be applied to

treat non-positive variables and discrete functions. Recently, Li and Tsai (2005) developed another method to modify Floudas's methods thus to treat continuous variables containing zero values; however, Li and Tsai's method is incapable of effectively handling mixed-integer variables. This study proposes a novel method to globally solve a GGP program with mixed integer free-sign variables. A free-sign variable is one which can be positive, negative, or zero.

The GGP program discussed in this study is expressed below:

GGP

$$\text{Min } \sum_{p=1}^{T_0} f_p(\mathbf{x}) \cdot G_p(\mathbf{y})$$

$$\text{s.t. } \sum_{q=1}^{T_w} f_{w,q}(\mathbf{x}) \cdot G_{w,q}(\mathbf{y}) \leq l_w, \quad w = 1, \dots, s,$$

where

$$(C1) \quad f_p(\mathbf{x}) = c_p \cdot \prod_{i=1}^n x_i^{\alpha_{p,i}}, \quad p = 1, \dots, T_0,$$

$$(C2) \quad f_{w,q}(\mathbf{x}) = h_{w,q} \cdot \prod_{i=1}^n x_i^{\alpha_{w,q,i}}, \quad w = 1, \dots, s, \quad q = 1, \dots, T_w,$$

$$(C3) \quad G_p(\mathbf{y}) = \prod_{j=1}^m g_{p,j}(y_j), \quad p = 1, \dots, T_0,$$

$$(C4) \quad G_{w,q}(\mathbf{y}) = \prod_{j=1}^m g_{w,q,j}(y_j), \quad w = 1, \dots, s, \quad q = 1, \dots, T_w,$$

(C5) $\mathbf{x} = (x_1, \dots, x_n)$, is a vector with free-sign continuous variables x_i , where x_i

can be positive, negative, or zero, $\underline{x}_i \leq x_i \leq \bar{x}_i$, \underline{x}_i and \bar{x}_i are constants,

(C6) $\mathbf{y} = (y_1, \dots, y_m)$, is a vector with free-sign discrete variables

$y_j, y_j \in \{d_{j,1}, \dots, d_{j,r_j}\}$ where $d_{j,k}$ can be positive, negative, or zero,

(C7) $g_{p,j}(y_j)$ and $g_{w,q,j}(y_j)$ are nonlinear functions of y_j ,

(C8) $\alpha_{p,i}$, $\alpha_{w,q,i}$, c_p , $h_{w,q}$, and l_w are constants,

(C9) $\alpha_{p,i}$ and $\alpha_{w,q,i}$ are integers if the lower bounds of x_i are negative.

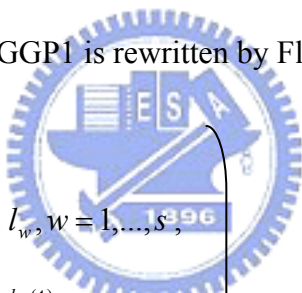
Floudas's method can solve a specific GGP problem containing continuous functions with positive variables, as illustrated below:

GGP1 (with continuous functions and positive variables)

$$\begin{aligned} \text{Min } & \sum_{p=1}^{T_0} f_p(\mathbf{x}) \\ \text{s.t. } & \sum_{q=1}^{T_w} f_{w,q}(\mathbf{x}) \leq l_w, \quad w = 1, \dots, S, \quad \mathbf{x} = (x_1, \dots, x_n), \quad \underline{x}_i \leq x_i \leq \bar{x}_i, \quad \forall \underline{x}_i > 0, \end{aligned}$$

where conditions (C1) and (C2) in GGP hold.

For solving GGP1, Floudas's method denotes $x_i = e^{t_i}$ with $t_i = \ln x_i$ and group all monomials with identical sign. GGP1 is rewritten by Floudas's method as follows:



$$\begin{aligned} \text{Min } & f_0^+(\mathbf{t}) - f_0^-(\mathbf{t}) \\ \text{s.t. } & f_w^+(\mathbf{t}) - f_w^-(\mathbf{t}) \leq l_w, \quad w = 1, \dots, S, \\ & f_0^+(\mathbf{t}) = \sum_{c_p > 0} c_p \cdot e^{h_p(\mathbf{t})}, \\ & f_0^-(\mathbf{t}) = \sum_{c_p < 0} -c_p \cdot e^{h_p(\mathbf{t})}, \\ & h_p(\mathbf{t}) = \sum_i \alpha_{p,i} \cdot t_i, \quad \forall p, \\ & f_w^+(\mathbf{t}) = \sum_{h_{w,q} > 0} h_{w,q} \cdot e^{h_{w,q}(\mathbf{t})}, \\ & f_w^-(\mathbf{t}) = \sum_{h_{w,q} < 0} -h_{w,q} \cdot e^{h_{w,q}(\mathbf{t})}, \\ & h_{w,q}(\mathbf{t}) = \sum_i \alpha_{w,q,i} \cdot t_i, \quad \forall w, q. \end{aligned} \tag{3.1}$$

GGP1 is a signomial geometric program containing posynomial functions $f_0^+(\mathbf{t})$,

$f_w^+(\mathbf{t})$ and signomial functions $-f_0^-(\mathbf{t})$, $-f_w^-(\mathbf{t})$ where $\mathbf{t}=(t_1, \dots, t_n)$ is a positive variable vector. A posynomial term is a monomial with positive coefficient, while a signomial term is a monomial with negative coefficient (Bazaraa et al. 1993; Floudas 2000). Supports all signomial terms in GGP1 are removed then we have a posynomial geometric program below:

$$\begin{aligned} & \text{Min } f_0^+(\mathbf{t}) \\ & \text{s.t. } f_w^+(\mathbf{t}) \leq l_w, w = 1, \dots, s. \end{aligned}$$

Posynomial geometric program laid the foundation for the theory of generalized geometric program. Duffin and Peterson (1966) pioneered the initial work on posynomial geometric programs, which derived the dual based on the arithmetic geometric inequality. This dual involves the maximization of a separable concave function subject to linear constraints. Unlike posynomial geometric problems, signomial geometric problems remain nonconvex and much more difficult to solve. Floudas's method employs an exponential variable transformation to the initial signomial geometric program to reduce it into a decomposition program. A convex relaxation is then obtained based on the linear lower bounding of the concave parts. Floudas's method can reach ε -convergence to the global minimum by successively refining a convex relaxation of a series of nonlinear convex optimization problems. The ε -convergence to the global minimum (ε -global minimum), as defined in Floudas (2000, page 58) is stated below: Suppose that \mathbf{x}^* is a feasible solution, $\varepsilon \geq 0$ is a small prescribed tolerance, and $f(\mathbf{x}) \geq f(\mathbf{x}^*) - \varepsilon$ for all feasible \mathbf{x} , then \mathbf{x}^* is an ε -global minimum. However, the usefulness of Floudas's method is limited by the difficulty that x_i must be strictly positive. This restriction prohibits many applications where x_i can be zero or negative values (such as temperature, growth rate, etc.).

In order to overcome the difficulty of Floudas's methods, recently, Li and Tsai (2005) proposed another method for solving GGP1 where x_i may be negative or zero. Li and Tsai

first transfer $\prod_{i=1}^n x_i^{\alpha_{p,i}}$ and $\prod_{i=1}^n x_i^{\alpha_{w,q,i}}$ into convex and concave functions, then approximate the concave functions by piecewise linear techniques. Li and Tsai's method can also reach finite ε -convergence to the global minimum. Both Floudas's methods and Li & Tsai's method use the exponential-based decomposition technique to decompose the objective function and constraints into convex and concave functions. Decomposition programs have good properties for finding a global optimum (Horst and Tuy, 1996). A convex relaxation of the decomposition can be computed conveniently based on the linear lower bound of the concave parts of the objective functions and constraints.

Two difficulties of direct application of Floudas's method or Li & Tsai method to globally solve a GGP problem are discussed below:

- (i) The major difficulty is that $r - 1$ binary variables are used in expressing a discrete variable with r values. For instance, to linearize a signomial term $G(\mathbf{y}) = g_1(y_1) \cdot g_2(y_2) = y_1^{\alpha_1} \cdot y_2^{\alpha_2}$ where $y_j \in \{d_{j,1}, d_{j,2}, \dots, d_{j,r_j}\} \forall j, r_j$, by taking the logarithm of $G(\mathbf{y})$ one obtains

$$\ln G(\mathbf{y}) = \alpha_1 \ln y_1 + \alpha_2 \ln y_2 = \sum_{j=1}^2 \alpha_j [\ln d_{j,1} + \sum_{k=2}^{r_j} u_{j,k} (\ln d_{j,k} - \ln d_{j,k-1})]$$

$$\text{where } \sum_{k=2}^{r_j} u_{j,k} \leq 1, u_{j,k} \in \{0,1\}.$$

It require $r_1 + r_2 - 2$ binary variables to linearly decompose $G(\mathbf{y})$. If the number of r_j is large then it will cause a heavy computational burden.

- (ii) Another difficulty is the treatment of taking logarithms of y_j . If y_j may take on a negative or zero value then we can not take logarithmic directly.

This study proposes a novel method to globally solve GGP programs. The advantages of the proposed method are listed as below:

Less number of binary variables and constraints are used in solving a GGP program. Only $\sum_{r=1}^m \lceil \log_2 r_j \rceil$ binary variables are required to linearize $g_{p,j}(y_j)$ and $g_{w,q,j}(y_j)$ in GGP. It is capable of treating non-positive variables in the discrete and continuous functions in GGP.

This study is organized as follows. Section 3.2 develops the first approach of treating discrete functions in GGP. Section 3.3 describes another approach of treating discrete functions. Section 3.4 proposes a method for handling continuous functions. Numerical examples are analyzed in Section 3.5.



3.2 The Proposed Linear Approximation Method

Form the basic of **THEOREM 2.1**, we develop two approaches to express a discrete variable y with r values. The first approach, as described in this section, use $\lceil \log_2 r \rceil$ binary variables and $2r$ extra constraints to express y . The second approach, as described in the next section, use $\lceil \log_2 r \rceil$ binary variables but only $3 + 4\lceil \log_2 r \rceil$ constraints to express y . The first approach is good at treating product terms $f(\mathbf{x}) \prod_{j=1}^m g_j(y_j)$, while the second approach is more effective in treating additive terms $\sum_j g_j(y_j)$.

REMARK 3.1 A discrete free-sign variable $y, y \in \{d_1, d_2, \dots, d_r\}$ can be expressed as:

$$d_k - M \cdot A_k \leq y \leq d_k + M \cdot A_k, \quad k = 1, \dots, r \quad (3.2)$$

where

(i) k is same as θ in (2.5) and A_k is same as $A_\theta(\theta')$ in (2.7).

(ii) M is a big enough positive value,

$$M = \max\{1, d_1, \dots, d_r\} - \min\{0, d_1, \dots, d_r\}. \quad (3.3)$$

PROOF (i) If $A_k = 0$ then $y = d_k$,

(ii) If $A_k \geq 1$ then $d_k - M \cdot A_k \leq \min\{0, d_1, \dots, d_r\} \leq y \leq \max\{0, d_1, \dots, d_r\} \leq d_k + M \cdot A_k$.

Therefore (3.2) is still correct. □

Take Table 2.1 for instance, for $y \in \{-1, 0, 1, 4, 5, 6, 7.5, 8, 9, 10\}$, y can be expressed by linear inequalities below:

$$-1 - M(0 + u'_1 + u'_2 + u'_3 + u'_4) \leq y \leq -1 + M(0 + u'_1 + u'_2 + u'_3 + u'_4),$$

$$0 - M(1 - u'_1 + u'_2 + u'_3 + u'_4) \leq y \leq 0 + M(1 - u'_1 + u'_2 + u'_3 + u'_4),$$

$$10 - M(2 - u'_1 + u'_2 + u'_3 - u'_4) \leq y \leq 10 + M(2 - u'_1 + u'_2 + u'_3 - u'_4),$$

where $M = 10 + 1 = 11$.

In this case $r=10$ and there are $2 \times 10 = 20$ constraints being used to express y . Since depending on the final u'_1, u'_2, u'_3 , and u'_4 assignments, only two of 20 inequalities will turn into equalities which indicate the discrete choice made of y , while the rest of 18 inequalities turn into redundant constraints.

REMARK 3.2 Expression (3.2) uses $\lceil \log_2 r \rceil$ binary variables and $2r$ constraints to express a discrete variable with r values.

PROPOSITION 3.1 Given a function $g(y)$ where $y \in \{d_1, d_2, \dots, d_r\}$, d_k are discrete free-sign values, $g(y)$ can be expressed by following linear inequalities:

$$g(d_k) - M \cdot A_k \leq g(y) \leq g(d_k) + M \cdot A_k, \quad k = 1, \dots, r$$

where y, A_k , and k are specified in REMARK 3.1, and

$$M = \max\{1, g(d_1), \dots, g(d_r)\} - \min\{0, g(d_1), \dots, g(d_r)\}. \quad (3.4)$$

PROPOSITION 3.2 A product term $z = f(\mathbf{x}) \cdot g(y)$, where $f(\mathbf{x}) = c \prod_{i=1}^n x_i^{\alpha_i}$ as being specified in GGP, $\underline{x}_i \leq x_i \leq \bar{x}_i$, c is a free-sign constant, and $y \in \{d_1, \dots, d_r\}$, can be expressed by following inequalities:

$$f(\mathbf{x}) \cdot g(d_k) - M' \cdot A_k \leq z \leq f(\mathbf{x}) \cdot g(d_k) + M' \cdot A_k, \quad k = 1, \dots, r$$

where

(i) y, A_k , and k are specified in PROPOSITION 3.1.

(ii) $M' = \overline{f(\mathbf{x})} \cdot M$, for

$$\overline{f(\mathbf{x})} = \max\{1, |c| \prod_{i=1}^n x_i^U\}, \text{ where } x_i^U = \max\{|x_i^{\alpha_i}|, \underline{x}_i \leq x_i \leq \bar{x}_i \forall i\}. \quad (3.5)$$

PROOF It is clear that $M' \geq M |f(\mathbf{x})|$ for $\underline{x}_i \leq x_i \leq \bar{x}_i$. Since $A_k \geq 0$,

$-M'A_k \leq -Mf(\mathbf{x})A_k$ and $M'A_k \geq Mf(\mathbf{x})A_k$. Two cases are discussed.

(i) Case 1 for $f(\mathbf{x}) \geq 0$. From Proposition 3 to have

$$f(\mathbf{x}) \cdot g(d_k) - Mf(\mathbf{x}) \cdot A_k \leq f(\mathbf{x})g(y) \leq f(\mathbf{x}) \cdot g(d_k) + Mf(\mathbf{x}) \cdot A_k.$$

We then have

$$f(\mathbf{x})g(d_k) - M' \cdot A_k \leq f(\mathbf{x})g(y) \leq f(\mathbf{x})g(d_k) + M' \cdot A_k.$$

(ii) Case 2 for $f(\mathbf{x}) < 0$. From Proposition 3 to have

$$f(\mathbf{x}) \cdot g(d_k) - Mf(\mathbf{x}) \cdot A_k \geq f(\mathbf{x})g(y) \geq f(\mathbf{x}) \cdot g(d_k) + Mf(\mathbf{x}) \cdot A_k.$$

Similar to Case 1, it is clear that

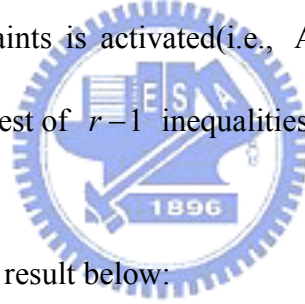
$$f(\mathbf{x})g(d_k) - M' \cdot A_k \leq f(\mathbf{x})g(y) \leq f(\mathbf{x})g(d_k) + M' \cdot A_k.$$

The proposition is then proven. □

REMARK 3.3 For a constraint $z = f(\mathbf{x}) \cdot g(y) \leq a$ where a is constant, $f(\mathbf{x})$ and $g(y)$ are the same as in PROPOSITION 3.2. The constraint $z \leq a$ can be expressed by following inequalities:

$$f(\mathbf{x}) \cdot g(d_k) - M' \cdot A_k \leq a, \quad k = 1, \dots, r,$$

where M' is same in PROPOSITION 3.2. There are r constraints used to describe $z \leq a$; where only one of the constraints is activated (i.e., $A_k = 0$) which indicates the discrete choice made by y , while the rest of $r-1$ inequalities turn into redundant constraints.



We then deduce the main result below:

THEOREM 3.1 Denote $z_1 = f(\mathbf{x}) \cdot g_1(y_1)$ and $z_\sigma = z_{\sigma-1} \cdot g_\sigma(y_\sigma) = f(\mathbf{x}) \cdot \prod_{j=1}^{\sigma} g_j(y_j)$ for

$\sigma = 2, \dots, m$ where $f(\mathbf{x}) = c \prod_{i=1}^n x_i^{\alpha_i}$, $\underline{x}_i \leq x_i \leq \bar{x}_i$, c is a free-sign constant, and y_j are

discrete free-sign variables, $y_j \in \{d_{j,1}, d_{j,2}, \dots, d_{j,r_j}\}$. The terms z_1, z_2, \dots, z_m can be

expressed by the following inequalities:

$$(C1) \quad f(\mathbf{x}) \cdot g_1(d_{1,k}) - M'_1 \cdot A_{1,k} \leq z_1 \leq f(\mathbf{x}) \cdot g_1(d_{1,k}) + M'_1 \cdot A_{1,k}, \quad k = 1, \dots, r_1,$$

$$(C2) \quad z_1 \cdot g_2(d_{2,k}) - M'_2 \cdot A_{2,k} \leq z_2 \leq z_1 \cdot g_2(d_{2,k}) + M'_2 \cdot A_{2,k}, \quad k = 1, \dots, r_2,$$

...

$$(Cm) \quad z_{m-1} \cdot g_m(d_{m,k}) - M'_m \cdot A_{m,k} \leq z_m \leq z_{m-1} \cdot g_m(d_{m,k}) + M'_m \cdot A_{m,k}, \quad k = 1, \dots, r_m.$$

where

$$(i) \quad d_{j,k} - M_j \cdot A_{j,k} \leq y_j \leq d_{j,k} + M_j \cdot A_{j,k}, \quad j = 1, \dots, m, \quad k = 1, \dots, r_j,$$

$$(ii) \quad A_{j,k} = 0.5 \sum_{w=1}^{\theta} (1 - (-1)^{\lfloor \frac{k-1}{2^{w-1}} \rfloor}) + \sum_{w=1}^{\theta} ((-1)^{\lfloor \frac{k-1}{2^{w-1}} \rfloor}) \cdot u_{j,w}, \quad \theta = 1, \dots, \lceil \log_2 r_j \rceil \text{ for all } j, k,$$

$$(iii) \quad k = 1 + \sum_{w=1}^{\lceil \log_2 r_j \rceil} 2^{w-1} u_{j,w} \leq r_j,$$

$$(iv) \quad M'_j = \overline{f(\mathbf{x})} \cdot \prod_{i=1}^j \sigma_i, \quad \overline{f(\mathbf{x})} \text{ is specified in (3.5),}$$

$$\left. \begin{aligned} \sigma_i &= \max_k \{1, g_i(d_{i,k})\} - \min_k \{0, g_i(d_{i,k})\}, \quad (3.6) \\ j &= 1, \dots, m, \quad k = 1, \dots, r_j. \end{aligned} \right\}$$

PROOF (i) Consider (C1). Since $M'_1 \geq M' \geq M = \sigma_1$, where M' and M are specified in PROPOSITION 3.2, (C1) is true.

$$(ii) \quad \text{Consider (C2). It is clear } M'_2 = \overline{f(\mathbf{x})} \cdot \sigma_1 \sigma_2.$$

$$\text{If } f(\mathbf{x}) \geq 0 \text{ then } z_1 \cdot g_2(d_{2,k}) - M'_2 \cdot A_{2,k} \leq z_1 \cdot g_2(d_{2,k}) - f(\mathbf{x}) \sigma_1 \sigma_2 \cdot A_{2,k} \leq z_1 \cdot g_2(d_{2,k})$$

$$\text{and } z_1 \cdot g_2(d_{2,k}) \leq z_1 \cdot g_2(d_{2,k}) + f(\mathbf{x}) \sigma_1 \sigma_2 \cdot A_{2,k} \leq z_1 \cdot g_2(d_{2,k}) + M'_2 \cdot A_{2,k}.$$

If $f(\mathbf{x}) < 0$

$$z_1 \cdot g_2(d_{2,k}) + M'_2 \cdot A_{2,k} \geq z_1 \cdot g_2(d_{2,k}) - f(\mathbf{x}) \sigma_1 \sigma_2 \cdot A_{2,k} \geq z_1 \cdot g_2(d_{2,k}) \text{ and}$$

$$z_1 \cdot g_2(d_{2,k}) \geq z_1 \cdot g_2(d_{2,k}) + f(\mathbf{x}) \sigma_1 \sigma_2 \cdot A_{2,k} \geq z_1 \cdot g_2(d_{2,k}) - M'_2 \cdot A_{2,k}.$$

We then have $z_1 \cdot g_2(d_{2,k}) - M'_2 \cdot A_{2,k} \leq z_2 \leq z_1 \cdot g_2(d_{2,k}) + M'_2 \cdot A_{2,k}$ for $A_{2,k} \geq 1$.

(iii) Similar for (C3) to (Cm).

The theorem is then proven. □

REMARK 3.4 A Constraint $\sum_{s=1}^p f_s(\mathbf{x}) \prod_{j=1}^m g_{s,j}(y_j) \leq a$, where a is constant

and $f_s(\mathbf{x}) = c_s \prod_{i=1}^n x_i^{\alpha_{i,s}}$, $\underline{x}_i \leq x_i \leq \bar{x}_i$, c is a free-sign constant, can be expressed by

following inequalities:

$$\sum_{s=1}^p z_{s,m} \leq a,$$

$$z_{s,m-1} \cdot g_{s,m}(d_{m,k}) - M_{s,m} \cdot A_{s,m,k} \leq z_{s,m} \forall s, k,$$

$$z_{s,m-2} \cdot g_{s,m-1}(d_{m-1,k}) - M_{s,m-1} \cdot A_{s,m-1,k} \leq z_{s,m-1} \leq z_{s,m-2} \cdot g_{s,m-1}(d_{m-1,k}) + M_{s,m-1} \cdot A_{s,m-1,k} \forall s, k,$$

⋮

$$z_{s,2} \cdot g_{s,2}(d_{2,k}) - M_{s,2} \cdot A_{s,2,k} \leq z_{s,2} \leq z_{s,2} \cdot g_{s,2}(d_{2,k}) + M_{s,2} \cdot A_{s,2,k} \forall s, k,$$

$$f_s(X) \cdot g_{s,1}(d_{1,k}) - M_{s,1} \cdot A_{s,1,k} \leq z_{s,1} \leq f_s(X) \cdot g_{s,1}(d_{1,k}) + M_{s,1} \cdot A_{s,1,k} \forall s, k,$$

where

$$\left. \begin{aligned} M_{s,i} &= \overline{f_s(\mathbf{x})} \cdot \prod_{j=1}^i \sigma_{s,i}, \quad \overline{f_s(\mathbf{x})} = \max\{1, |C_s| \prod_{i=1}^n x_{s,i}^{U_i}\}, \quad x_{s,i}^U = \max\{x_i^{\alpha_{s,i}} \mid \underline{x}_i \leq x_i \leq \bar{x}_i\} \forall s, i, \\ \sigma_{s,i} &= \max\{1, g_{s,i}(d_{i,k})\} - \min\{0, g_{s,i}(d_{i,k})\}. \end{aligned} \right\} (3.7)$$

For instance, a single nonlinear constraint $xy_1^2 + y_1y_2 \leq a$ where $0 \leq x \leq 5$, y_1, y_2 are integer variables, $1 \leq y_1, y_2 \leq 32$, can be converted into a linear system as follows.

(i) Denote $y_1 \in \{1, \dots, 32\} = \{d_1, \dots, d_{32}\}$. Since $\lceil \log_2 32 \rceil = 5$, five binary variables

u_1, u_2, u_3, u_4 , and u_5 are used to express y_1 . Specifying k as:

$$k = 1 + u_1 + 2u_2 + 2^2u_3 + 2^3u_4 + 2^4u_5.$$

Referring to (3.2), y_1 is expressed as

$$d_k - M_1 \cdot A_{1,k} \leq y_1 \leq d_k + M_1 \cdot A_{1,k}, k = 1, \dots, 32, \quad (3.8)$$

where $M_1 = 32$ and $A_{1,k}$ are specified as

$$A_{1,1} = u_1 + u_2 + u_3 + u_4 + u_5,$$

$$A_{1,2} = 1 - u_1 + u_2 + u_3 + u_4 + u_5,$$

⋮

$$A_{1,32} = 5 - u_1 - u_2 - u_3 - u_4 - u_5.$$

Similarly, denote $y_2 \in \{1, \dots, 32\} = \{b_1, \dots, b_{32}\}$, five binary variables v_1, v_2, v_3, v_4 , and v_5 are used to express y_2 . Specifying q as:

$$q = 1 + v_1 + 2v_2 + 2^2v_3 + 2^3v_4 + 2^4v_5.$$

y_2 is expressed in the same way as in y_1 :

$$b_q - M_2 \cdot A_{2,q} \leq y_2 \leq b_q + M_2 \cdot A_{2,q}, q = 1, \dots, 32, \quad (3.9)$$

where $M_2 = 32$. Specifying $A_{2,q}$ as

$$A_{2,1} = v_1 + v_2 + v_3 + v_4 + v_5,$$

$$A_{2,2} = 1 - v_1 + v_2 + v_3 + v_4 + v_5,$$

⋮

$$A_{2,32} = 5 - v_1 - v_2 - v_3 - v_4 - v_5.$$

- (ii) Treating the first term xy_1^2 in the constraint. Denote $z_3 = xy_1^2$. From Theorem 1 to have:

$$x \cdot d_k^2 - M'_1 \cdot A_{1,k} \leq z_3, k = 1, \dots, 32,$$

where $M'_1 = 5 * (32^2 - 0) = 5120$

- (iii) The second term y_1y_2 in constraint is treated similarly. Denote $z_4 = y_1y_2$. To form

the inequalities as

$$y_1 \cdot b_q - M'_2 \cdot A_{2,q} \leq z_4, q = 1, \dots, 32,$$

where $M'_2 = 32 * 32 = 1024$.

The original constraint then becomes $z_3 + z_4 \leq a$ where there are additional 10 binary variables and two continuous variables. By utilizing Theorem 1 to express this constraint in linear form requires 192 additive constraints (where 64 come from (3.8), 64 come from (3.9)). Another approach, developed based on Theorem 1, can be used to reduce the number of constraints from 192 to $64 + 2(3 + 4\lceil \log_2 32 \rceil) = 110$.

3.3 Treatment of Discrete Function in GGP (Approach 2)

PROPOSITION 3.1 uses $\lceil \log_2 r \rceil$ binary variables and $2r$ additive constraints to express a discrete function $g(y)$ variable with r values. Here we use PROPOSITION 2.2 and PROPOSITION 2.3 to express the same discrete variable where only $3 + 4\lceil \log_2 r \rceil$ additive constraints are required. Such an expression is computationally more efficient to treat $g(y)$ in a GGP program.

We use p_k and δ_w to instead of r_θ and z_j in PROPOSITION 2.2 and PROPOSITION 2.3 in this section respectively. For instance, given $y \in \{-1, 0, 1, 4, 5, 6, 7.5, 8, 9, 10\}$, referring to Table 2.1 to have

$$\begin{aligned}
\sum_{k=1}^{10} p_k A_k &= p_1(u_1 + u_2 + u_3 + u_4) + p_2(1 - u_1 + u_2 + u_3 + u_4) + \dots + p_{10}(2 - u_1 + u_2 + u_3 - u_4) \\
&= u_1(p_1 - p_2 + p_3 - p_4 + p_5 - p_6 + p_7 - p_8 + p_9 - p_{10}) \\
&\quad + u_2(p_1 + p_2 - p_3 - p_4 + p_5 + p_6 - p_7 - p_8 + p_9 + p_{10}) \\
&\quad + u_3(p_1 + p_2 + p_3 + p_4 - p_5 - p_6 - p_7 - p_8 + p_9 + p_{10}) \\
&\quad + u_4(p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 - p_9 - p_{10}) = 0.
\end{aligned}$$

The discrete variable y can be expressed by following linear equation and linear inequalities:

$$\left. \begin{aligned}
y &= -p_1 + p_3 + 4p_4 + 5p_5 + 6p_6 + 7.5p_7 + 8p_8 + 9p_9 + 10p_{10}, \\
\delta_1 + \delta_2 + \delta_3 + \delta_4 + p_2 + p_3 + \dots + 2p_{10} &= 0, \\
-u_w &\leq \delta_w \leq u_w, w = 1, \dots, 4, \\
p_1 - p_2 + p_3 - \dots - p_{10} - 1 + u_1 &\leq \delta_1 \leq p_1 - p_2 + p_3 - \dots - p_{10} + 1 - u_1, \\
p_1 + p_2 - p_3 - \dots - p_{10} - 1 + u_2 &\leq \delta_2 \leq p_1 + p_2 - p_3 - \dots - p_{10} + 1 - u_2, \\
&\vdots \\
p_1 + p_2 + p_3 + \dots - p_{10} - 1 + u_4 &\leq \delta_4 \leq p_1 + p_2 + p_3 + \dots - p_{10} + 1 - u_4, \\
\sum_{k=1}^{10} p_k &= 1, p_k \geq 0.
\end{aligned} \right\} (3.10)$$

It is convenient to check: If $u_2 = 1$ and $u_1 = u_3 = u_4 = 0$ then $\delta_1 = \delta_3 = \delta_4 = 0$ and

$$\delta_2 = p_1 + p_2 - p_3 - p_4 + p_5 + p_6 - p_7 - p_8 + p_9 + p_{10}.$$

Since $\delta_2 + p_2 + p_3 + 2p_4 + p_5 + 2p_6 + 2p_7 + 3p_8 + p_9 + 2p_{10} =$

$$p_1 + 2p_2 + p_4 + 2p_5 + 3p_6 + p_7 + 2p_8 + 2p_9 + 3p_{10} = 10, p_k \geq 0, \text{ and } \sum_{k=1}^{10} p_k = 1, \text{ it is clear}$$

$p_1 = p_2 = p_4 = p_5 = p_6 = p_7 = p_8 = p_9 = p_{10} = 0, \delta_2 + p_3 = 0, \delta_2 = -1$ and $p_3 = 1$. We

then have $y = p_3 = 1$.

REMARK 3.5 $\lceil \log_2 r \rceil$ binary variables, $3 + 4\lceil \log_2 r \rceil$ constraints, and $r + \lceil \log_2 r \rceil$ continuous variables are used in PROPOSITION 2.2 and PROPOSITION 2.3 to express a discrete variable with r values.

REMARK 3.6 Given a function $g(y)$ where $y \in \{d_1, \dots, d_r\}$, $g(y)$ can be expressed by following expressions.

$$g(d_k) = \sum_{k=1}^r g(d_k) p_k$$

where $\sum_{k=1}^r p_k = 1$, $p_k \geq 0$, and $\sum_{k=1}^r p_k A_k = 0$ are specified similarly.

For instance, a function $g(y) = (10 - y)/(1 + y^2)$ can be expressed by the following expressions:

$$g(y) = \sum_{k=1}^{10} \left(\frac{10 - d_k}{1 + d_k^2} \right) p_k,$$

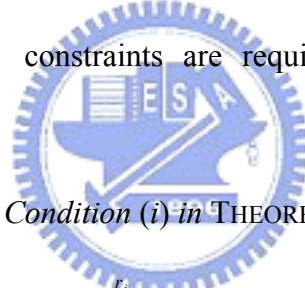
Suppose $u_2 = u_3 = u_4 = 0$ and $u_1 = 1$ then $k = 2, p_2 = 1, y = 0$ and $g(y) = 10$.

Table 3.1 is a comparison of the three ways for expressing a discrete variable y with r values. A numerical example is given in Example 3.4 to compare their computational efficiency.

Table 3.1 Way of expressing $y \in \{d_1, \dots, d_r\}$

	Floudas's Method, Li&Tsai's Method	Approach 1 REMARK 3.1	Approach 2 PROPOSITION 2.2
Expression	$y = \sum_{k=1}^r u_k d_k$ $\sum_{k=1}^r u_k = 1, \forall u_k \in \{0,1\}$	$y \geq d_k - MA_k,$ $y \leq d_k + MA_k,$ $k = 1, \dots, r.$	$y = \sum_{k=1}^r d_k p_k, \sum_{k=1}^r p_k = 1,$ $\sum_{k=1}^r p_k A_k = 0.$
No. of binary variables	r	$\lceil \log_2 r \rceil$	$\lceil \log_2 r \rceil$
No. of additional constraints	2	$2r$	$3 + 4\lceil \log_2 r \rceil$
No. of additional continuous variables	0	0	$r + \lceil \log_2 r \rceil$

Now consider Condition (i) in THEOREM 3.1, where y_j are expressed by Approach 1 using $2r_j$ constraints. That condition can be replaced by PROPOSITION 2.2 and PROPOSITION 2.3 where only $3 + 4\lceil \log_2 r \rceil$ constraints are required. This is described as following proposition:



PROPOSITION 3.3 Replacing Condition (i) in THEOREM 3.1 by following expressions:

$$y_j = \sum_{k=1}^{r_j} d_{j,k} p_{j,k}, j = 1, \dots, m$$

where $\sum_{k=1}^{r_j} p_{j,k} = 1, p_{j,k} \geq 0,$ and $\sum_{k=1}^{r_j} p_{j,k} A_{j,k} = 0$ for all j, k . The total number of variables and constraints used to express y, z_1, \dots, z_m using (C1), (C2), ..., (Cm) in THEOREM 3.1 are listed below:

- (i) no. of binary variables: $\sum_{j=1}^m \lceil \log_2 r_j \rceil,$
- (ii) no. of constraints for $g_j(y_j): 3m + \sum_{j=1}^m \lceil \log_2 r_j \rceil,$
- (iii) no. of constraints for z_1, z_2, \dots, z_m in (C1), (C2), ..., (Cm): $4 \sum_{j=1}^m r_j,$
- (iv) no. of continuous variables: $\sum_{j=1}^m (r_j + 1) + \sum_{j=1}^m \lceil \log_2 r_j \rceil.$

3.4 Treatment of Continuous Function in GGP

Consider the signomial form $f(\mathbf{x}) = c \prod_{i=1}^n x_i^{\alpha_i}$, according to Li and Tsai (2005), we illustrate the technique of treating $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \dots, x_n)$, x_i as free-sign variables, $\underline{x}_i \leq x_i \leq \bar{x}_i$, and c is a constant. Introducing new binary variables θ_i, λ_i and a positive continuous variable $0 \leq x_i^0 \leq \max\{-\underline{x}_i, \bar{x}_i\}$ which are specified below:

- (i) $x_i = 0$ if and only if $\lambda_i = 0$,
- (ii) $x_i > 0, x_i = x_i^0, 0 < x_i^0 \leq \bar{x}_i$ if and only if $\lambda_i = 1$ and $\theta_i = 0$,
- (iii) $x_i < 0, x_i = -x_i^0, \underline{x}_i \leq -x_i^0 < 0$ if and only if $\lambda_i = 1$ and $\theta_i = 1$

The above conditions can be represented by a set of linear inequalities below:

$$\begin{aligned}
 & 0 \leq x_i^0 \leq \max\{-\underline{x}_i, \bar{x}_i\}, \\
 & x_i = x_i^0 \lambda_i (1 - 2\theta_i).
 \end{aligned}
 \Leftrightarrow
 \begin{cases}
 (i) & 0 \leq x_i^0 \leq M\lambda_i, \\
 (ii) & 0 \leq x_i^0 \leq \bar{x}_i(1 - \theta_i) - \underline{x}_i\theta_i, \\
 (iii) & x_i^0 - M\theta_i \leq x_i \leq x_i^0 + M\theta_i, \\
 (iv) & -x_i^0 - M(1 - \theta_i) \leq x_i \leq -x_i^0 + M(1 - \theta_i), \\
 (v) & \lambda_i, \theta_i \in \{0, 1\}.
 \end{cases}$$

Here $M = 2 \max\{-\underline{x}_i, \bar{x}_i\}$. Now denote $f(\mathbf{x})$ and $f^0(\mathbf{x})$ as below:

$$f(\mathbf{x}) = c \prod_{i=1}^n x_i^{\alpha_i} \quad \text{and} \quad f^0(\mathbf{x}) = c \prod_{i=1}^n (x_i^0)^{\alpha_i} \quad \text{where } x_i^0 \text{ are positive variables mentioned}$$

above, it is clear that

$$f(\mathbf{x}) = f^0(\mathbf{x}) \cdot \prod_{i=1}^n \lambda_i^{\alpha_i} (1 - 2\theta_i)^{\alpha_i}, \quad \lambda_i, \theta_i \in \{0, 1\} \quad (3.11)$$

Expression (3.11) implies:

- (i) If $\lambda_i = 0 \forall i$ then $f(\mathbf{x}) = 0$,

(ii) If $\lambda_i = 1, \theta_i = 1$, and α_i is odd for all $i \in \{1, 2, \dots, n\}$ then $f(\mathbf{x}) = -f^0(\mathbf{x})$.

By specifying three new variables λ, θ , and t defined below:

(i) $\lambda \leq \lambda_i$,

(ii) $\theta = \sum_{i \in I} \theta_i - 2t$, $I = \{i \mid x_i < 0 \text{ and } \alpha_i \text{ is odd for all } i\}$,

(iii) $0 \leq t \leq 0.5(\sum_{i \in I} \theta_i + 1)$,

where $\lambda, \theta \in \{0, 1\}$, and t is an integer variable.

It is clear that if $\lambda = 0$ then $f(\mathbf{x}) = 0$; if $\theta = 0$ then $f(\mathbf{x}) = f^0(\mathbf{x})$; and if $\theta = 1$ then $f(\mathbf{x}) = -f^0(\mathbf{x})$. We then have following proposition:

PROPOSITION 3.4 Consider a signomial continuous function $f(\mathbf{x}) = c \prod_{i=1}^n x_i^{\alpha_i}$ where

$\mathbf{x} = (x_1, x_2, \dots, x_n)$, x_i are free-sign variables, $\underline{x}_i \leq x_i \leq \bar{x}_i$, and c is a constant. $f(\mathbf{x})$ can be expressed with the following inequalities:

$$-M'\lambda \leq f(\mathbf{x}) \leq M'\lambda,$$

$$f^0(\mathbf{x}) - M'\theta \leq f(\mathbf{x}) \leq f^0(\mathbf{x}) + M'\theta,$$

$$-f^0(\mathbf{x}) - M'(1 - \theta) \leq f(\mathbf{x}) \leq -f^0(\mathbf{x}) + M'(1 - \theta),$$

$$\theta = \sum_{i \in I} \theta_i - 2t,$$

$$0 \leq t \leq 0.5 \sum_{i \in I} \theta_i,$$

$$\sum_{i=1}^n \lambda_i - (n-1) - \lambda \leq 0,$$

$$\lambda \leq \lambda_i, i = 1, \dots, n,$$

$$0 \leq x_i^0 \leq M\lambda_i, i = 1, \dots, n,$$

$$0 \leq x_i^0 \leq \bar{x}_i(1 - \theta_i) - \underline{x}_i\theta_i, \forall i \in J,$$

$$\begin{aligned}
x_i^0 - M\theta_i &\leq x_i \leq x_i^0 + M\theta_i, \forall i \in J, \\
-x_i^0 - M(1-\theta_i) &\leq x_i \leq -x_i^0 + M(1-\theta_i), \forall i \in J, \\
0 &< x_i^0 \leq \bar{x}_i, i \notin J, \\
x_i &= x_i^0, i \notin J,
\end{aligned}$$

where

$$(i) f^0(\mathbf{x}) = c \prod_{i=1}^n (x_i^0)^{\alpha_i},$$

$$(ii) \lambda, \theta, \lambda_i, \theta_i \in \{0, 1\}, M = 2 \max\{-\underline{x}_i, \bar{x}_i\},$$

(iii) t is an integer variable,

(iv) J is a set containing all i where the lower bound of x_i are negative, and I

denotes a set of all i where $i \in J$ and α_i is odd, J and I are expressed as

$$J = \{i \mid \underline{x}_i < 0 \forall i\} \text{ and}$$

$$I = \{i \mid \underline{x}_i < 0 \text{ and } \alpha_i \text{ is odd } \forall i\}.$$



PROOF Referring to Li and Tsai (2005). □

In order to globally solve a GGP problem, $f^0(\mathbf{x})$ in PROPOSITION 3.4 should be approximately linearized. The most convenient approach to linearizing $f^0(\mathbf{x})$ is to take logarithms to reformulate $f^0(\mathbf{x})$ as the differences of some convex functions in (3.1) as proposed by Floudas's method. Such an approach, however, requires that we add numerous new binary variables. For instance, to treat a constraint

$$z = x_1^{-5} x_2^{-1} x_3^2 \leq 10 \text{ where } 1 \leq x_1, x_2, x_3 \leq 5. \quad (3.12)$$

Floudas's method first denotes $z_1 = \log x_1$, $z_2 = \log x_2$, and $z_3 = \log x_3$, then to reformulates

$z \leq 10$ as the following linear inequalities:

$$-0.5z_1 - z_2 + 2z_3 \leq \log 10,$$

$$L(\log x_i) \leq z_i \leq \log x_i, \quad i = 1, 2, 3.$$

Since $\log x_i$ are concave functions, linearizing $L(\log x_i)$ with m break points requires the addition of $3(m-1)$ binary variables for piecewise linearization. Here we propose some convexification and concavification strategies to reduce the number of binary variables required to linearize $L(\log x_i)$. Consider the following propositions:

PROPOSITION 3.5 *A piecewise linearized function $L(f(x))$ of a concave function $f(x)$ with m break points $b_1 < b_2 < \dots < b_m$ can be expressed as*

$$(C1) \quad b_k - M \cdot A_k \leq x \leq b_{k+1} + M \cdot A_k, \quad k = 1, \dots, m-1,$$

$$(C2) \quad f(b_k) + s_k(x - b_k) - M \cdot A_k \leq L(f(x)) \leq f(b_k) + s_k(x - b_k) + M \cdot A_k, \quad k = 1, \dots, m-1$$

where

$$(i) \quad s_k = \frac{f(b_{k+1}) - f(b_k)}{b_{k+1} - b_k},$$

$$(ii) \quad A_k \text{ are specified as the similar way in (2.7),}$$

$$(iii) \quad M \text{ is a large enough positive value.}$$

Obviously, only $\lceil \log_2(m-1) \rceil$ binary variables are required to create the piecewise linear function $L(f(x))$.

PROOF Referring to Tsai et al. (2002). □

PROPOSITION 3.6 *A twice-differentiable function $f(\mathbf{x}) = cx_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}, x_i \geq 0 \forall i = 1, \dots, n$, is a convex function in one of the following conditions:*

$$(C1) \quad c \geq 0, \quad \alpha_i < 0 \forall i,$$

$$(C2) \quad c < 0, \quad \alpha_i > 0 \forall i, \quad \sum_{i=1}^n \alpha_i \leq 1.$$

PROOF Referring to Tsai et al. (2002). □

To simplify the expression, here we use a signomial term with three variables as instances to illustrate the convexification and concavefication techniques.

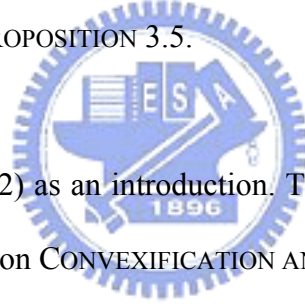
CONVEXIFICATION AND CONCAVIFICATION STRATEGIES 1

Consider a constraint $z = x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} \leq \text{constant}$ with $x_i \geq 0 (i = 1, 2, 3)$, $\alpha_1, \alpha_2 < 0$, $\alpha_3 > 0$. This constraint can be converted into the following inequalities:

$$x_1^{\alpha_1} x_2^{\alpha_2} w_3^{-\alpha_3} \leq \text{constant},$$

$$x_3^{-1} \leq w_3 \leq L(x_3^{-1})$$

where $x_1^{\alpha_1} x_2^{\alpha_2} w_3^{-\alpha_3}$, x_3^{-1} are convex (following (C1) of PROPOSITION 3.6, and $L(x_3^{-1})$ is a piecewise linear function by PROPOSITION 3.5.



Take the example in (3.12) as an introduction. The constraint $z \leq 10$ is converted into the following inequalities base on CONVEXIFICATION AND CONCAVIFICATION STRATEGIES 1:

$$x_1^{-0.5} x_2^{-1} w_3^{-2} \leq 10,$$

$$x_3^{-1} \leq w_3 \leq L(x_3^{-1}),$$

where only $\lceil \log_2(m-1) \rceil$ binary variables are used instead of the $3(m-1)$ binary variables required by conventional methods.

CONVEXIFICATION AND CONCAVIFICATION STRATEGIES 2

A constraint $z = -x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} \leq \text{constant}$ with $x_i \geq 0 (i = 1, 2, 3)$, $\alpha_1, \alpha_2, \alpha_3 < 0$. This constraint can be converted into following inequalities:

$$-w_1^{\frac{1}{3}} w_2^{\frac{1}{3}} w_3^{\frac{1}{3}} \leq \text{constant},$$

$$x_i^{3\alpha_i} \leq w_i \leq L(x_i^{3\alpha_i}) \quad \text{for } i = 1, 2, 3,$$

where $-w_1^{\frac{1}{3}}w_2^{\frac{1}{3}}w_3^{\frac{1}{3}}, x_i^{3\alpha_i}$ are convex (following (C2) of PROPOSITION 3.6), and $L(x_i^{3\alpha_i})$ is a piecewise linear function by PROPOSITION 3.5.

3.5 Numerical Examples

EXAMPLE 3.1 GGP with Linear Continuous Function

Consider following program with one continuous free-sign variable x and two discrete variables y_1 and y_2 :

PROGRAM 1

$$\begin{aligned} \text{Min } & xy_1^3y_2 + xy_1y_2^2 \\ \text{s.t. } & \text{(C1) } xy_1^2 + y_1y_2 \leq -500, \\ & \text{(C2) } -xy_1 + y_1^2y_2 \leq 500, \\ & \text{(C3) } -5 \leq x \leq 5, \\ & \text{(C4) } y_1 \in \{-1, 0, 1, 4, 5, 6, 7.5, 8, 9, 10\}, \\ & \text{(C5) } y_2 \in \{-27, -18, -9, -7, -4, -1, 1, 3, 4, 5\}. \end{aligned}$$

Program 1 can not be solved by Floudas's methods (1997, 1999, 2000) or Li and Tasi's (2005) approach, since there are zero and negative values for continuous and discrete variables. Our solution proceeds as follows:

- (i) Denote $y_1 \in \{-1, 0, 1, 4, 5, 6, 7.5, 8, 9, 10\} = \{d_1, \dots, d_{10}\}$. Since $\lceil \log_2 10 \rceil = 4$, four binary variables u_1, u_2, u_3 , and u_4 are used to express y_1 . Specifying k as:

$$k = 1 + u_1 + 2u_2 + 2^2u_3 + 2^3u_4 \leq 10. \quad (3.13)$$

y_1 can be expressed as in (3.10) where y is replaced by y_1 and p_k are replaced

by . Specifying $A_{1,k}$ as

$$A_{1,1} = u_1 + u_2 + u_3 + u_4, A_{1,2} = 1 - u_1 + u_2 + u_3 + u_4, \dots, A_{1,10} = 2 - u_1 + u_2 + u_3 - u_4. \quad (3.14)$$

Similarly, denote $y_2 \in \{-27, -18, -9, -7, -4, -1, 1, 3, 4, 5\} = \{b_1, \dots, b_{10}\}$, four binary

variables v_1, v_2, v_3 , and v_4 are used to express y_2 . Specifying q as:

$$q = 1 + v_1 + 2v_2 + 2^2v_3 + 2^3v_4 \leq 10. \quad (3.15)$$

y_2 is expressed in the same way as in y_1 :

$$\left. \begin{aligned} y_2 &= -27p_{2,1} - 18p_{2,2} - 9p_{2,3} - 7p_{2,4} - 4p_{2,5} - p_{2,6} + p_{2,7} + 3p_{2,8} + 4p_{2,9} + 5p_{2,10}, \\ \delta_{2,1} + \delta_{2,2} + \delta_{2,3} + \delta_{2,4} + p_{2,2} + p_{2,3} + \dots + 2p_{2,10} &= 0, \\ -v_w &\leq \delta_{2,w} \leq v_w, w = 1, \dots, 4, \\ p_{2,1} - p_{2,2} + p_{2,3} - \dots - p_{2,10} - 1 + v_1 &\leq \delta_1 \leq p_{2,1} - p_{2,2} + p_{2,3} - \dots - p_{2,10} + 1 - v_1, \\ &\vdots \\ p_{2,1} + p_{2,2} + p_{2,3} + \dots - p_{2,10} - 1 + v_4 &\leq \delta_4 \leq p_{2,1} + p_{2,2} + p_{2,3} + \dots - p_{2,10} + 1 - v_4, \\ \sum_{q=1}^{10} p_{2,q} &= 1, p_{2,q} \geq 0. \end{aligned} \right\} \quad (3.16)$$

Specify $A_{2,q}$ as

$$A_{2,1} = v_1 + v_2 + v_3 + v_4, A_{2,2} = 1 - v_1 + v_2 + v_3 + v_4, \dots, A_{2,10} = 2 - v_1 + v_2 + v_3 - v_4. \quad (3.17)$$

(ii) Now treat the term $xy_1^3y_2$ in the objective function.

Denote $z_1 = xy_1^3y_2$ and $z_{11} = xy_1^3$. From Theorem 1 to have:

$$\left. \begin{aligned} z_{11} \cdot b_q - M'_2 \cdot A_{2,q} &\leq z_1, q = 1, \dots, 10, \\ x \cdot d_k^3 - M'_1 \cdot A_{1,k} &\leq z_{11} \leq x \cdot d_k^3 + M'_1 \cdot A_{1,k}, k = 1, \dots, 10. \end{aligned} \right\} \quad (3.18)$$

where M'_j are the same as in (3.6). Since x is a continuous free-sign variable, here

x^0, λ , and θ are introduced to form following inequalities based on PROPOSITION 3.4:

$$\left. \begin{aligned}
0 \leq x^0 \leq M\lambda, \\
0 \leq x^0 \leq \bar{x}(1-\theta) - \underline{x}\theta, \\
x^0 - M\theta \leq x \leq x^0 + M\theta, \\
-x^0 - M(1-\theta) \leq x \leq -x^0 + M(1-\theta),
\end{aligned} \right\} \quad (3.19)$$

where $\theta, \lambda \in \{0,1\}$.

(iii) Other nonconvex terms are treated similarly as in (3.18) and (3.19). These terms are denoted as $z_2 = x_1y_1y_2^2$, $z_3 = x_1y_1^2$, $z_4 = y_1y_2$, $z_5 = x_1y_1$, and $z_6 = y_1^2y_2$.

The original problem is then converted into a mixed 0-1 linear program below, which contains 162 linear constraints and 8 binary variables.

PROGRAM 2

Min $z_1 + z_2$

s.t. (C1) $z_3 + z_4 \leq -500$,

(C2) $-z_5 + z_6 \leq 500$,

(C3) $-5 \leq x \leq 5$,

(C4) (3.10), (3.13)–(3.19),

where z_2, z_3, z_4, z_5, z_6 are expressed similar to (3.18) and (3.19),

$u_1, u_2, u_3, u_4, v_1, v_2, v_3, v_4, \lambda, \theta \in \{0,1\}$, and M is a big value.

Solving Program 2 to obtain the globally optimal solution as $(x, y_1, y_2) = (-4.9, 10, -1)$ with the objective value 4851.

EXAMPLE 3.2 GGP with Signomial Continuous Function

Here we use a signomial continuous function $x_1^3 x_2$ to replace x in Program 1. The example is formulated below:

PROGRAM 3

Min (The same objective function in Program 1)

s.t. (C1) The same constraints in Program 1 but erase the constraint: $-5 \leq x \leq 5$,

(C2) $x = x_1^3 x_2, -5 \leq x_1 \leq 5, -5 \leq x_2 \leq 5$.

Compared with the solution process of Program 2, the extra work is to treat the equality constraint $x = x_1^3 x_2$ as follows:

(i) Expressing free-sign variables x_1 and x_2 by PROPOSITION 3.4 as inequalities below:

$$\left. \begin{aligned}
 &0 \leq x_i^0 \leq M\lambda_i, \\
 &0 \leq x_i^0 \leq \bar{x}_i(1-\theta_i) - x_i\theta_i, \\
 &x_i^0 - 10\theta_i \leq x_i \leq x_i^0 + 10\theta_i, \\
 &-x_i^0 - 10(1-\theta_i) \leq x_i \leq -x_i^0 + 10(1-\theta_i), \\
 &\lambda \leq \lambda_i,
 \end{aligned} \right\} i=1, 2 \tag{3.20}$$

$$\left. \begin{aligned}
 &\theta = \theta_1 + \theta_2 - 2t, \\
 &0 \leq t \leq 0.5(\theta_1 + \theta_2), \\
 &\lambda_1 + \lambda_2 - 1 - \lambda \leq 0, \\
 &-M'\lambda \leq x \leq M'\lambda,
 \end{aligned} \right\} \tag{3.21}$$

where $\lambda_1, \lambda_2, \lambda, \theta_1, \theta_2, \theta \in \{0,1\}$, and t is a integer variable.

(ii) Let $x^0 = (x_1^0)^3 (x_2^0)$, expressing the relationship between x and x^0 as the following inequalities based on Proposition 5:

$$x^0 - M'\theta \leq x \leq x^0 + M'\theta,$$

$$-x^0 - M'(1-\theta) \leq x \leq -x^0 + M'(1-\theta).$$

(iii) Using Convexification and Concavification Strategies to convert the inequalities in (ii)

as follows:

$$\left. \begin{aligned} w_1^{-3} w_2^{-1} - M'\theta \leq x \leq w_3^{0.5} w_4^{0.5} + M'\theta, \\ -w_3^5 w_4^5 - M'(1-\theta) \leq x \leq -w_1^{-3} w_2^{-1} + M'(1-\theta), \end{aligned} \right\} \quad (3.22)$$

$$\left. \begin{aligned} (x_1^0)^{-1} \leq w_1 \leq L((x_1^0)^{-1}), \\ (x_2^0)^{-1} \leq w_2 \leq L((x_2^0)^{-1}), \\ (x_1^0)^6 \leq w_3 \leq L((x_1^0)^6), \\ (x_2^0)^2 \leq w_4 \leq L((x_2^0)^2), \end{aligned} \right\} \quad (3.23)$$

where

(a) $w_1^{-3} w_2^{-1}$ is convex form of x^0 and $w_3^5 w_4^5$ is a concave form of x^0 ,

(b) $L((x_1^0)^{-1}), L((x_2^0)^{-1}), L((x_1^0)^6)$, and $L((x_2^0)^2)$ are piecewise linear functions of $(x_1^0)^{-1}, (x_2^0)^{-1}, (x_1^0)^6$, and $(x_2^0)^2$, respectively.

(c) $M = 2\overline{f(\mathbf{x})}$, $\overline{f(\mathbf{x})}$ is specified in (3.5).

The whole program is reformulated as the following mixed 0-1 convex program:

PROGRAM 4

$$\text{Min } z_1 + z_2$$

s.t. (C1) The same constraints in Program 2,

(C2) Constraints in (3.20)–(3.23).

This is a mixed binary convex program which can be solved to reach finite ε -convergence to the global optimal solution as

$$(x_1, x_2, y_1, y_2) = (-4.9051, 4.75, 1, -27) \text{ with the objective value } -369954.$$

EXAMPLE 3.3 Comparisons of Computational Efficiency for Signomial Function

This example is used to compare the computational efficiency between the proposed method and Floudas's methods. Consider following GGP problem:

PROGRAM 5

$$\begin{aligned} \text{Min } & \prod_{i=1}^n x_i^{\alpha_i} - \sum_{j=1}^n \prod_{\substack{i=1..n \\ i \neq j}} x_i^{\alpha_i} \\ \text{s.t. } & b \leq \sum_{j=1}^n \prod_{\substack{i=1..n \\ i \neq j}} x_i^{\alpha_i} \leq 2b, \end{aligned}$$

where x_i are positive integer variables, $1 \leq x_i \leq m \forall i$,

α_i and b are constants.

For $n=2$, the objective function of the above problem becomes $x_1^{\alpha_1} x_2^{\alpha_2} - x_1^{\alpha_1} - x_2^{\alpha_2}$.

For $n=3$, the objective function of above problem becomes $x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} - x_1^{\alpha_1} x_2^{\alpha_2} - x_2^{\alpha_2} x_3^{\alpha_3} - x_1^{\alpha_1} x_3^{\alpha_3}$. Reforming this problem with $n=3$ by Floudas's method to have the following convex integer program:

PROGRAM 6

$$\begin{aligned} \text{Min } & e^{\sum_{i=1}^3 \alpha_i y_i} - \sum_{j=1}^3 e^{\sum_{\substack{i=1,3 \\ i \neq j}} \alpha_i y_i} \\ \text{s.t. } & \text{(C1) } b \leq \sum_{j=1}^3 e^{\sum_{\substack{i=1,3 \\ i \neq j}} \alpha_i y_i} \leq 2b, \\ & \text{(C2) } y_i = \sum_{j=1}^m (u_{i,j} \cdot \ln d_{i,j}) \forall i, \\ & \text{(C3) } \sum_{j=1}^m (u_{i,j}) = 1 \forall i, u_{i,j} \in \{0,1\} \forall i, j. \end{aligned}$$

Resolving this problem with $n=3$ by the proposed method to have the following linear mixed binary program:

PROGRAM 7

$$\text{Min } z_{123} - z_{12} - z_{23} - z_{13}$$

$$s.t. \quad (C1) \quad b \leq z_{12} + z_{23} + z_{13} \leq 2b,$$

$$(C2) \quad z_i = \sum_{j=1}^m d_{i,j}^{\alpha_i} p_{i,j}, \quad \sum_{j=1}^m p_{i,j} A_{i,j} = 0, \quad \sum_{j=1}^m p_{i,j} = 1, \quad p_{i,j} \geq 0, \quad i=1,2,3,$$

$$(C3) \quad z_1 \cdot d_{2,j}^{\alpha_2} - M \cdot A_{2,j} \leq z_{12} \leq z_1 \cdot d_{2,j}^{\alpha_2} + M \cdot A_{2,j} \quad \forall j,$$

$$(C4) \quad z_2 \cdot d_{3,j}^{\alpha_3} - M \cdot A_{3,j} \leq z_{23} \leq z_2 \cdot d_{3,j}^{\alpha_3} + M \cdot A_{3,j} \quad \forall j,$$

$$(C5) \quad z_1 \cdot d_{3,j}^{\alpha_3} - M \cdot A_{3,j} \leq z_{13} \leq z_1 \cdot d_{3,j}^{\alpha_3} + M \cdot A_{3,j} \quad \forall j,$$

$$(C6) \quad z_{12} \cdot d_{3,j}^{\alpha_3} - M \cdot A_{3,j} \leq z_{123} \leq z_{12} \cdot d_{3,j}^{\alpha_3} + M \cdot A_{3,j} \quad \forall j,$$

$$(C7) \quad \sum_{k=1}^{\lceil \log_2 m \rceil} (2^{k-1} \cdot u_{i,k}) \leq m \forall i, \quad u_{i,k} \in \{0,1\} \forall i,k.$$

All M'_x come from (3.7). By specifying various values for m and b , we solve both programs using LINGO (2004). The related CPU time, number of binary variables, and number of constraints are reported in Table 3.2.

Table 3.2 indicates that both methods reach the same ε – global optimum for $\varepsilon = 10^{-8}$. The number of binary variables used in Floudas's method is larger than the proposed method, while the proposed method uses more constraints than Floudas's method. The CPU time demonstrates that the proposed method is much more computationally efficient than Floudas's methods.

Table 3.2 Experiment result of Example 3.3

Experiment	Conditions	Proposed method	Floudas's method
1	$m = 4$	CPU Time	1sec
	$b = 3$	Number of binary variables	6
		Number of constraints	55
		Objective Value	-5.389143
2	$m = 8$	CPU Time	3sec
	$b = 4$	Number of binary variables	9
		Number of constraints	0
		Objective Value	-7.452201
3	$m = 16$	CPU Time	11sec
	$b = 10$	Number of binary variables	12
		Number of constraints	223
		Objective Value	-19.73737
4	$m = 32$	CPU Time	2min 12sec
	$b = 31$	Number of binary variables	15
		Number of constraints	447
		Objective Value	-61.78579

* $(\alpha_1, \alpha_2, \alpha_3) = (-2, 0.5, 1.2)$

EXAMPLE 3.4 Comparison of Computational Efficiency for Discrete Variables with Large Sizes

Consider following program with three discrete variables y_1, y_2 and y_3 , where y_1 contains 8 possible values and y_2, y_3 both contain 128 possible values.

PROGRAM 8

$$\begin{aligned} \text{Min } & y_1^2 y_2^{0.816} - y_2^{0.5} - y_3^{1.2} \\ \text{s.t. (C1) } & y_1^{0.8} + y_2^{0.9} + y_3^{0.5} \geq 16, \\ & \text{(C2) } y_1^{-1.5} + y_2^{1.7} + y_3^{1.2} \leq 31, \end{aligned}$$

where $y_1 \in \{1.1, 3.2, 5.3, 7.4, 9.5, 12.6, 14.7, 16.8\}$,

$$\begin{aligned} y_2, y_3 \in \{ & 1.1, 1.2, 1.3, 1.4, 1.5, 2.0, 2.6, 2.7, 2.8, 2.9, \\ & 3.1, 3.2, 3.3, 3.4, 3.5, 4.0, 4.6, 4.7, 4.8, 4.9, \\ & \vdots \\ & 23.1, 23.2, 23.3, 23.4, 23.5, 24.0, 24.6, 24.7, 24.8, 24.9, \\ & 25.1, 25.2, 25.3, 25.4, 25.5, 26.0, 26.6, 26.7, 26.8, 26.9\}. \end{aligned}$$

Floudas's method (or Li and Tsai's method) converts this program into following form:

PROGRAM 9

$$\text{Min } e^{2\ln y_1 + 0.816 \ln y_2} - y_2^{0.5} - y_3^{1.2}$$

$$s.t. \quad (C1), (C2),$$

$$\text{where } \ln y_1 = u_1 \ln 1.1 + u_2 \ln 3.2 + \dots + u_8 \ln 16.8,$$

$$\ln y_2 = v_1 \ln 1.1 + v_2 \ln 1.2 + \dots + v_{128} \ln 26.7,$$

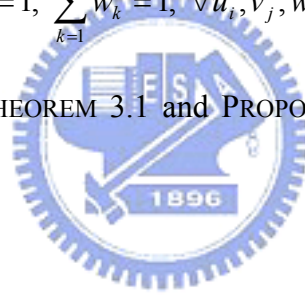
$$y_1^\alpha = 1.1^\alpha u_1 + 3.2^\alpha u_2 + \dots + 16.8^\alpha u_8, \quad \alpha = 0.8, -1.5,$$

$$y_2^\beta = 1.1^\beta v_1 + 1.2^\beta v_2 + \dots + 26.7^\beta v_{128}, \quad \beta = 0.5, 0.9, 1.7,$$

$$y_3^\gamma = 1.1^\gamma w_1 + 1.2^\gamma w_2 + \dots + 26.7^\gamma w_{128}, \quad \gamma = 1.2, 0.5,$$

$$\sum_{i=1}^8 u_i = 1, \quad \sum_{j=1}^{128} v_j = 1, \quad \sum_{k=1}^{128} w_k = 1, \quad \forall u_i, v_j, w_k \in \{0,1\}.$$

Our proposed method (i.e., THEOREM 3.1 and PROPOSITION 3.3) transforms this program into following form:



PROGRAM 10

$$\text{Min } z - y_2^{0.5} - y_3^{1.2}$$

$$s.t. \quad (C1), (C2),$$

$$1.1^2 z_1 - M'(p_1 - 1) \leq z \leq 1.1^2 z_1 + M'(p_1 - 1),$$

$$3.2^2 z_1 - M'(p_2 - 1) \leq z \leq 3.2^2 z_1 + M'(p_2 - 1),$$

⋮

$$16.8^2 z_1 - M'(p_8 - 1) \leq z \leq 16.8^2 z_1 + M'(p_8 - 1),$$

$$\text{where } z_1 = 1.1^{0.816} q_1 + 1.2^{0.816} q_2 + \dots + 26.7^{0.816} q_{128},$$

$$y_1^\alpha = 1.1^\alpha p_1 + 3.2^\alpha p_2 + \dots + 16.8^\alpha p_{128}, \quad \alpha = 0.8, -1.5,$$

$$y_2^\beta = 1.1^\beta q_1 + 1.2^\beta q_2 + \dots + 26.7^\beta q_{128}, \quad \beta = 0.5, 0.9, 1.7,$$

$$y_3^\gamma = 1.1^\gamma s_1 + 1.2^\gamma s_2 + \dots + 26.7^\gamma s_{128}, \quad \gamma = 1.2, 0.5,$$

$$\sum_{i=1}^8 p_i = 1, \quad \sum_{i=1}^8 p_i A_i = 0, \quad \forall p_i \geq 0, \quad A_i \text{ composed of 3 binary variables},$$

$$\sum_{j=1}^{128} q_j = 1, \quad \sum_{j=1}^{128} q_j A'_j = 0, \quad \forall q_j \geq 0, \quad A'_j \text{ composed of 7 binary variables},$$

$$\sum_{k=1}^{128} s_k = 1, \quad \sum_{k=1}^{128} s_k A''_k = 0, \quad \forall s_k \geq 0, \quad A''_k \text{ composed of 7 binary variables},$$

$$M' = 16.8^2 \times 26.7^{0.816}.$$

The number of binary variables, constraints, continuous variables, as well as the number of iterations and CPU time are listed in Table 3.3. Table 3.3 illustrates that both programs obtain the same global optimum with $y_1 = 16.8, y_2 = 3.1, y_3 = 14$, and objective value is 685.0155 as solved by LINGO (2004). However, the proposed method is more computational efficient than Floudas's method and Li&Tsai's method.

Table 3.3 Comparisons of existing methods with proposed method

	No. of binary variables	No. of constraints	No. of continuous variables	No. of iterations	CPU time (second)
Floudas's method and Li&Tsai's method	264	13	3	53588	72
Proposed method	17	30	269	11139	3

Chapter 4 Extension 2—Solving Task Allocation Problem

The task allocation problem (TAP) is one where a number of tasks or modules need to be assigned to a set of processors or machines at minimum overall cost. The overall cost includes the communication cost between tasks that are assigned to different processors and other costs such as the assignment cost and the fixed cost of using processors. Processors may have limited or unlimited capacities to perform tasks. Task allocation has been applied to the design of distributed computing systems and also in auto-manufacturing contexts. We try to find an efficient method to conquer the classic problems based on SRB. We also report some computational experience for the TAP.

4.1 Introduction to Task Allocation Problem

The task allocation problem (TAP) is to assign a set of tasks or modules to a set of processors or machines so that the overall cost is minimized. The overall cost may include the task assignment cost and the interprocessor communication cost, among others. And each processor has a limited capacity or memory. The TAP is an important problem that arises in distributed computing systems (see Stone 1977) and has also been applied in the automobile manufacturing industry in Rao (1992). Many heuristic and exact algorithms have been designed for solving various versions of the TAP. Heuristic algorithms are developed in Dutta et al. (1982), Kopidakis et al. (1997), Lo (1988), Ma et al. (1982), Milis (1995), and Sarje and Sagar (1991); metaheuristic algorithms such as simulated annealing, tabu search, and genetic algorithms are proposed in Chen and Lin (2000), Hadj-Alouane et al. (1999), and Hamam and Hindi (2000); and exact mathematical programming and/or branch-and-bound approaches are used in Billionnet et al. (1992), Magirou and Milis (1989), Sinclair (1987), and Stone (1977).

Because of the complexity of the TAP, none of the above-mentioned algorithms, with

the exception of Billionnet et al. (1992), are capable of solving some real-world applications optimally. Most versions of the TAP can be formulated as integer quadratic programs (see Hadj-Alouane et al. 1999, Hamam and Hindi 2000). The potential of mathematical programming approaches has not been fully explored for solving the TAP. In the last decade, great progress has been made both in computational power and computational technology of mathematical programming. Column generation and branch-and-price are two most important approaches to conquer the classic large binary problems; however, we proposed a different method and try to find an efficient exact mathematical programming formula.

For example, a TAP with n agents and m tasks requires nm binary variables in deterministic approaches to obtain global solution while our method, based on SRB introduced in Chapter 2, only needs $\lceil \log_2 nm \rceil$ binary variables.

4.2 The Proposed Mathematical Programming Formula

n agents are used to perform m tasks, each of which is assigned to exactly one processor. Let $d_{i,k}$ be the execution cost of task i if it is assigned to agent k . Let b_k and s_k be the capacity and the fixed cost for agent k , respectively. Let $a_{i,k}$ be the amount of the resource required from its assigned processor for task i . If two tasks i and j are assigned to two different agents, then the cost of communication between i and j is $c_{i,j}$ (we assume that $c_{i,j} = c_{j,i}$ and $c_{i,i} = 0$). Define the binary variables. $x_{i,k} = 1$ if and only if task i is allocated to agent k . Define y_k to be a binary variable so that $y_k = 1$ if and only if agent k is assigned at least one task. The TAP can be cast as an integer quadratic program:

TAP

$$\text{Min } \sum_{i=1}^{m-1} \sum_{j=i+1}^m c_{i,j} (1 - \sum_{k=1}^n z_{i,j,k}) + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} x_{i,k} + \sum_{k=1}^n s_k y_k$$

$$\text{Min } \sum_{i=1}^{m-1} \sum_{j=i+1}^m c_{i,j} (1 - \sum_{k=1}^n x_{i,k} x_{j,k}) + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} x_{i,k} + \sum_{k=1}^n s_k y_k$$

$$\text{s.t. } \sum_{k=1}^n x_{i,k} = 1, \quad i = 1, \dots, m, \quad (4.1)$$

$$\sum_{i=1}^m a_{i,k} x_{i,k} \leq b_k y_k, \quad k = 1, \dots, n, \quad (4.2)$$

$$x_{i,k} \leq y_k \leq 1, \quad i = 1, \dots, m, \quad k = 1, \dots, n, \quad (4.3)$$

where (i) s_k be the fixed cost for agent k ,

(ii) $d_{i,k}$ be the execution cost of task i if it is assigned to agent k ,

(iii) $c_{i,j}$ be the communication cost between two task i and task j

assigned to two different processors (assume that $c_{i,j} = c_{j,i}$ and

$c_{i,i} = 0$),

(iv) b_k be the capacity for agent k ,

(v) $a_{i,k}$ be the resource required of task i if it is assigned to agent k ,

(vi) $x_{i,k} \in \{0,1\} \forall i,k$ and when $x_{i,k} = 1$ if and only if task i is allocated

to agent k ,

(vii) $y_k = 1$ if and only if agent j is assigned at least one task.

The TAP is an integer program with quadratic terms $x_{i,k} x_{j,k}$. It is generally computationally expensive to find optimal solutions of integer quadratic programs. Let $z_{i,j,k}$ be the three-index binary variable ($i = 1, \dots, m-1, j = i+1, \dots, m, k = 1, \dots, n$), and $z_{i,j,k} = 1$ if and only if both task i and task j assigned to agent k . Based on $z_{i,j,k}$, the quadratic term

in TAP has been replaced by a linear term in the TAP3 through the three-index variable

$$z_{i,j,k}.$$

TAP3

$$\text{Min } \sum_{i=1}^{m-1} \sum_{j=i+1}^m c_{i,j} (1 - \sum_{k=1}^n z_{i,j,k}) + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} x_{i,k} + \sum_{k=1}^n s_k y_k$$

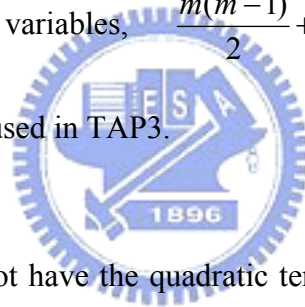
$$s.t. \quad (4.1), (4.2), (4.3),$$

$$z_{i,j,k} \leq x_{i,k}, \quad i = 1, \dots, m-1, \quad j = i+1, \dots, m, \quad k = 1, \dots, n,$$

$$z_{i,j,k} \leq x_{j,k}, \quad i = 1, \dots, m-1, \quad j = i+1, \dots, m, \quad k = 1, \dots, n,$$

where conditions (i)~(vii) in TAP are hold and $z_{i,j,k} \geq 0 \forall i, j, k$.

REMARK 4.1 nm binary variables, $\frac{m(m-1)}{2} + n$ continuous variables, and $nm^2 + 2n + m$ constraints are used in TAP3.



Even though TAP3 do not have the quadratic term present in the TAP, TAP3 is still a formidable task to solve integer linear programs with many variables when m and n are large. Therefore, it is desirable to propose formulations with two indices, Ernst et al (2006) let $u_{i,k}$ and $v_{i,k}$ be a nonnegative variable representing the total communication cost via agent k between task i and all other tasks and the total communication cost between task i and all tasks assigned to agent k when i is not assigned to agent k respectively. The following is a two-index formulation TAP2A:

TAP2A

$$\text{Min } \sum_{i=1}^m \sum_{k=1}^n \frac{1}{4} (u_{i,k} + v_{i,k}) + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} x_{i,k} + \sum_{k=1}^n s_k y_k$$

$$s.t. \quad (4.1), (4.2), (4.3),$$

$$u_{i,k} - v_{i,k} = \sum_{j=1}^m c_{i,j} (x_{i,k} - x_{j,k}), \quad u_{i,k}, v_{i,k} \geq 0, \quad i = 1, \dots, m, \quad k = 1, \dots, n,$$

where conditions (i)~(vii) in TAP are hold.

REMARK 4.2 nm binary variables, $2nm + n$ continuous variables, and $2nm + 2n + m$ constraints are used in TAP2A.

Based on TAP2A, we also proposed another two-index formulation TAP2B following:

TAP2B

$$\text{Min } C_1 - \sum_{i=1}^{m-1} \sum_{k=1}^n u_{i,k} + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} x_{i,k} + \sum_{k=1}^n s_k y_k$$

s.t. (4.1), (4.2), (4.3),

$$u_{i,k} \leq \sum_{j=i+1}^m c_{i,j} x_{j,k}, \quad i = 1, \dots, m-1, \quad k = 1, \dots, n,$$

$$u_{i,k} \leq x_{i,k} C_i, \quad i = 1, \dots, m-1, \quad k = 1, \dots, n.$$

where $C_1 = \sum_{i=1}^{m-1} \sum_{j=i+1}^m c_{i,j}$, $C_i = \sum_{j=i+1}^m c_{i,j}$, $i = 1, \dots, m-1$, $x_{i,k} \in \{0,1\}$, $u_{i,k} \geq 0 \quad \forall i, k$.

REMARK 4.3 nm binary variables, nm continuous variables, and $3nm + m$ constraints are used in TAP2B.

We apply the SRB for the above two-index formulations to reduce the number of binary variables. Use continuous nonnegative variables p_{ik} to replace the original binary variables x_{ik} and convert TAP2A and TAP2B to SRB2A and SRB2B respectively.

SRB2A

$$\begin{aligned}
\text{Min} \quad & \sum_{i=1}^m \sum_{k=1}^n \frac{1}{4} (u_{ik} + v_{ik}) + \sum_{i=1}^m \sum_{k=1}^n d_{ik} p_{ik} + \sum_{k=1}^n s_k y_k \\
\text{s.t.} \quad & \sum_{k=1}^n p_{i,k} = 1, \quad i = 1, \dots, m, \tag{4.4} \\
& \sum_{i=1}^m a_{i,k} p_{i,k} \leq b_k y_k, \quad k = 1, \dots, n, \tag{4.5} \\
& p_{i,k} \leq y_k \leq 1, \quad i = 1, \dots, m, \quad k = 1, \dots, n, \tag{4.6} \\
& u_{i,k} - v_{i,k} = \sum_{j=1}^m c_{i,j} (p_{i,k} - p_{j,k}), \quad u_{i,k}, v_{i,k}, p_{i,k}, p_{j,k} \geq 0, \quad i = 1, \dots, m, \quad k = 1, \dots, n, \\
& \sum_{\theta=1}^{\lceil \log_2 n \rceil} 2^{\theta-1} x'_{i,\theta} \leq n-1, \quad i = 1, \dots, m, \tag{4.7} \\
& \sum_{k=1}^n p_{i,k} |G(k)| + \sum_{\theta=1}^{\lceil \log_2 n \rceil} z_{i,\theta} = 0, \quad i = 1, \dots, m, \tag{4.8} \\
& -x'_{i,\theta} \leq z_{i,\theta} \leq x'_{i,\theta}, \quad \theta = 1, \dots, \lceil \log_2 n \rceil, \quad i = 1, \dots, m, \tag{4.9} \\
& \sum_{k=1}^n p_{i,k} c_{\theta,k} - (1 - x'_{i,\theta}) \leq z_{i,\theta} \leq \sum_{k=1}^n p_{i,k} c_{\theta,k} + (1 - x'_{i,\theta}), \quad \theta = 1, \dots, \lceil \log_2 n \rceil, \quad i = 1, \dots, m, \tag{4.10}
\end{aligned}$$

where $x'_{i,\theta} \in \{0,1\} \forall i, \theta$, $|G(k)|$ are same in (2.9), $c_{\theta,k}$ are same in (2.10), and conditions (i)~(vii) in TAP are hold.

REMARK 4.4 $m \lceil \log_2 n \rceil$ binary variables, $3nm + n + m \lceil \log_2 n \rceil$ continuous variables, and $2nm + 2n + 3m + 4m \lceil \log_2 n \rceil$ constraints are used in SRB2A.

SRB2B

$$\begin{aligned}
\text{Min} \quad & C_1 - \sum_{i=1}^{m-1} \sum_{k=1}^n u_{i,k} + \sum_{i=1}^m \sum_{k=1}^n d_{i,k} p_{i,k} + \sum_{k=1}^n s_k y_k \\
\text{s.t.} \quad & u_{i,k} \leq \sum_{j=i+1}^m c_{i,j} p_{j,k}, \quad i = 1, \dots, m-1, \quad k = 1, \dots, n,
\end{aligned}$$

$$u_{i,k} \leq p_{i,k} C_i, \quad i = 1, \dots, m-1, \quad k = 1, \dots, n.$$

$$(4.4) - (4.10),$$

where $C_1 = \sum_{i=1}^{m-1} \sum_{j=i+1}^m c_{i,j}$, $C_i = \sum_{j=i+1}^m c_{i,j}$, $i = 1, \dots, m-1$, and all conditions in SRB2A are hold.

REMARK 4.5 $m \lceil \log_2 n \rceil$ binary variables, $2nm + m \lceil \log_2 n \rceil$ continuous variables, and $3nm + 3m + 4m \lceil \log_2 n \rceil$ constraints are used in SRB2B.

4.3 Experiment Results

Table 4.1 is the experiment result of all method proposed in previous section in different combination of n agents and m tasks. Although SRB2A and SRB2B use less binary variables, its take much more time to obtain global solution. We will continue to find a efficient method to conquer this open question in our future research.

Table 4.1 Experiment result of TAP

n/m	Conditions	Method				
		TAP3	TAP2A	TAP2B	SRB2A	SRB2B
8/8	CPU Time	2.64 sec	0.63 sec	0.64sec	1.62sec	1.8 sec
	Iterations	8196	8300	4201	17449	16983
	brance-bound node	316	495	236	941	921
	Solution			7454		
10/10	CPU Time	7.47 sec	2.14 sec	2.5 sec	10.14 sec	11.78 sec
	Iterations	20825	23537	21704	86338	90765
	brance-bound node	589	951	1025	3998	4313
	Solution			10827		
12/12	CPU Time	268 sec	115 sec	106 sec	562 sec	590 sec
	Iterations	921084	1086282	846251	3436776	3143365
	brance-bound node	18749	31479	25799	104852	87886
	Solution			13544		
14/14	CPU Time	2054 sec	1597 sec	1724 sec	6857 sec	6082 sec
	Iterations	5132526	9447065	7447914	35883840	25079947
	brance-bound node	62476	257220	182813	988543	645217
	Solution			16588		

Chapter 5 Discussion and Concluding Remarks

5.1 Discussion

In this study, we develop the core technique of this dissertation, Superior Representation of Binary Variables (SRB), which uses only $\lceil \log_2 m \rceil$ binary variables to replace m original traditional binary variables (u_1, \dots, u_m) with $\sum_{\theta=1}^m u_{\theta} = 1$. It can effectively reduce the traditional binary variables number from $O(m)$ to $O(\log_2 m)$ and easily applies in any mixed-integer problems.

By using SRB, we propose a superior way of expressing the same piecewise linear function. A less number of binary variables and additive constraints are used. It is easy to use in various fields required piecewise, such as data fitting, network analysis, logistics, and production planning (Bazaraa et al. 1999). For expressing a function $f(x)$ with $m+1$ break points, the existing methods require m binary variables and $4m$ constraints (i.e., inequalities) while proposed methods require only $\lceil \log_2 m \rceil$ binary variables and $8 + 8\lceil \log_2 m \rceil$ constraints. The numerical examples demonstrate the computational efficiency of the proposed method over the existing methods especially when the number of break points is large.

We also extend SRB to apply Generalized Geometric Programs Problems. Based on SRB, we propose a novel method for handling a GGP problem with free-sign continuous and discrete variables. Since proposed method can treat free-sign and mixed-integer variables of GGP, it is possible to obtain the global optimization effectively in all applications with GGP, such as heat exchanger network design (Duffin and Peterson 1966), capital investment (Hellinckx and Rijckaert 1971), optimal design of cooling towers (Ecker and Wiebking 1978), batch plant modeling (Salomone and Iribarren 1992), competence sets

expansion (Li 1999), smoothing splines (Cheng et al. 2005), and digital circuit (Boyd 2005). We first use $\lceil \log_2 r \rceil$ binary variables to express a discrete function containing r discrete values, and utilize convexification and concavification strategies to treat the continuous signomial function. Comparing with of existing GGP methods, the proposed method can solve the problem to reach approximated global optimum using less number of binary variables and constraints.

The main issue on the above proposed method is that the original binary variables (u_1, \dots, u_m) must satisfy the following constraint:

$$\sum_{\theta=1}^m u_{\theta} = 1.$$

Actually, the proposed method is not appropriate for some practice applications which do not meet the above constraint.



5.2 Concluding Remarks

This study develops the Superior Representation of Binary Variables (SRB) core technique. It can effectively reduce the traditional binary variables number from $O(m)$ to $O(\log_2 m)$ and easily applies in any mixed-integer problems; moreover, it has some directions for future research are described below:

- (i) Task allocation problem (TAP): For example, a TAP with n agents and m tasks requires nm binary variables in existing deterministic approaches to obtain global solution while our method, based on SRB, only needs $\lceil \log_2 nm \rceil$ binary variables.

We try to find an efficient method to conquer the classic TAP. Furthermore, it is appropriate to solve the Multilevel Generalized Assignment Problem.

(ii) Haplotype inference: This is the optimal haplotype inference (OHI) problem as given a set of genotypes and a set of related haplotypes, find a minimum subset of haplotypes that can resolve all the genotypes in biology, which is also NP-hard and can be formulated as an integer quadratic programming (IQP) problem. We also want to find an efficient method through our SRB approach.

(iii) Integrating deterministic and heuristic approaches: It is an appropriate way to solve large scale optimal problems under the following strategy:

(a) Use the heuristic algorithms to find an initial solution to enhance the efficiency of finding the optimal solution.

(b) Then, use deterministic approaches are utilized to transcend the incumbent solution.



References

- Bazaraa, M.S., H.D. Sherali, C.M. Shetty. 1993. *Nonlinear Programming: Theory and Algorithms, 2e*. Wiley, New York.
- Beck, P.A., Ecker, J.G. 1975. A modified concave simplex algorithm for geometric programming. *Journal of Optimization Theory and Applications* **15** 189-202.
- Boyd, S.P., Kim, S.J., Patil, D.D., Horowitz, M.A. 2005. Digital Circuit Optimization via Geometric Programming. *Operations Research* **53** 899-932.
- Billionnet, A., Costa, M.C. Sutter, A. 1992. An efficient algorithm for a task allocation problem. *Journal of the Association for Computing Machinery* **39** 502–518.
- Chen, C.S., Lee, S.M., Shen, Q.S. 1995. An analytical model for the container loading problem. *European Journal of Operational Research* **80** 68 – 76.
- Chen, W., Lin, C. 2000. A hybrid heuristic to solve a task allocation problem. *European Journal of Operational Research* **27** 287–303.
- Cheng, H., Fang, S.C., Lavery, J.E. 2005. A Geometric Programming Framework for Univariate Cubic L1 Smoothing Splines. *Annals of Operations Research* **133** 229-248.
- Coloni, A., Dorigo, M., Maniezzo, V. 1992. Distributed Optimization by Ant Colonies. In F. J. Varela and P. Bourguine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA, 134-142.
- Croxton, K.L., B. Gendron, T.L. Magnanti. 2003. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science* **49** 1268–1273.
- Dantzing, G.B. 1960. On the significance of solving linear programming problems with some integer variables. *Econometrica* **28** 30–44.
- Dixon, L.C.W., G.P. Szegö. 1975. *Towards Global Optimization*. North-Holland, Amsterdam.

- Dutta, A., Koehler, G., Whinston, A. 1982. On optimal allocation in a distributed processing environment. *Management Science* **28** 839–853.
- Duffin, R.J. 1970. Linearizing geometric programming. *SIAM Review* **12** 211-227.
- Duffin, R.J., Peterson, E.L. 1966. Duality theory for geometric programming. *SIAM Journal on Applied Mathematics* **14** 1307-1349.
- Duffin, R.J., Peterson, E.L., Zener, C. 1967. *Geometric Programming: Theory and Application*. John Wiley & Sons, New York.
- Ecker, J.G., Kupferschmid, M., Lawrence, C.E., Reilly, A.A., Scott, A.C.H. 2002. An application of nonlinear optimization in molecular biology. *European Journal of Operational Research* **138** 452–458.
- Ecker, J.G., Wiebking, R.D. 1978. Optimal Design of a Dry-Type Natural-Draft Cooling Tower by Geometric Programming. *Journal of Optimization Theory and Applications* **26** 305-323.
- Faina, L. 2000. A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research* **126** 340 – 354.
- Floudas, C.A. 1999. Global optimization in design and control of chemical process systems. *Journal of Process Control* **10** 125 – 134.
- Floudas, C.A. 2000. *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academic Publishers.
- Floudas, C.A., Akrotirianakis, I.G., Caratzoulas, S., Meyer, C.A., Kallrath, J. 2005. Global optimization in the 21st century: Advances and challenges. *Computers and Chemical Engineering* **29** 1185-1202
- Floudas, C.A., Pardalos, P.M. 1996. *State of the Art in Global Optimization: Computational Methods and Applications*. Kluwer Academic Publishers.
- Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gumus, Z.H., Harding S.T., Klepeis, J.L., Meyer, C.A. and Schweiger, C.A. 1999. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Boston, 85-105.

- Fu, J.F., Fenton, R.G., Cleghorn, W.L. 1991. A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Engineering Optimization* **17** 263–280.
- Glover, F., Laguna, M. 1997. *Tabu Search*. Kluwer Academic Publishers, Boston, MA.
- Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, San Mateo, CA.
- Hadj-Alouane, A.B., Bean, J.C., Murty, K.G. 1999. A hybrid genetic optimization algorithm for a task allocation problem. *Journal of Scheduling* **2** 189–201.
- Hamam, Y., Hindi, K.S. 2000. Assignment of program modules to processors: A simulated annealing approach. *European Journal of Operational Research* **122** 509–513.
- Hellinckx, L.J., Rijckaert, M.J. 1971. Minimization of capital investment for batch processes. *Industrial & Engineering Chemistry Process Design and Development* **10** 422-423.
- Horst, R., Tuy, H. 1996. *Global Optimization: Deterministic Approaches, 3rd ed.* Springer, Berlin, Germany.
- Kochenberger, A., Woolsey, R.E.D., McCarl, B.A. 1973. On the solution of geometric programs via separable programming. *Operations Research* **24** 285-294.
- Kontogiorgis, S. 2000. Practical Piecewise-Linear Approximation for Monotropic Optimization. *INFORMS Journal on Computing* **12** 324-340.
- Kopidakis, Y., Laman, M., Zissimopoulos, V. 1997. On the task assignment problem: Two new efficient heuristic algorithms. *Journal of Parallel and Distributed Computing* **42** 21–29.
- Li, H.L. 1996. Notes: an efficient method for solving linear goal programming problems. *Journal of Optimization Theory and Applications* **90** 465–469.
- Li, H.L. 1999. Incorporating Competence Sets of Decision Makers by Deduction Graphs. *Operations Research* **47** 209-220.

- Li, H.L., Chang, C.T., Tsai, J.F. 2002. An approximately global optimization for assortment problems using piecewise linearization techniques. *European Journal of Operational Research* **140** 584–589.
- Li, H.L., Chou, C.T. 1994. A Global Approach for Nonlinear Mixed Discrete Programming in Design Optimization. *Engineering Optimization* **22** 109–122.
- Li, H.L., Lu, H.C. 2008. Optimization for Generalized Geometric Programs with Mixed Free-Sign Variables. *Operations Research* (was accepted, 21/01/2008).
- Li, H.L., Tsai, J.F. 2005. Treating free-sign variables in Generalized geometric global optimization programs. *Journal of Global Optimization* **33** 1-13.
- Lin, X., Orlowska, M. 1995. An integer linear programming approach to data allocation with minimum total communication. *Information Sciences* **85** 1–10.
- LINGO. 2004. Release. 9. Lindo System Inc., Chicago.
- Lo, V.M. 1988. Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers* **37** 1384–1397.
- Ma, P.Y.R., Lee, E.Y.S. Tsuchiya, M.T. 1982. A task allocation model for distributed computing systems. *IEEE Transactions on Computers* **C-31** 41–47.
- Magirou, V.F., Milis, J.Z. 1989. An algorithm for the multiprocessor assignment problem. *Operations Research Letters* **8** 351–356.
- Maranas, C.D., Floudas, C.A. 1997. Global optimization in generalized geometric programming. *Computer and Chemical Engineering* **21** 351–370.
- Milis, I. 1995. Task assignment in distributed systems using network flow methods. *Lecture Notes in Computer Science* **1120**. Springer-Verlag, London, UK, 396–405.
- Padberg, M. 2000. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters* **27** 1–5.
- Pardalos, P.M. and Romeijn, H.E., ed. 2002. *Handbook of Global Optimization-Volumn 2: Heuristic Approaches*. Kluwer Academic Publishers, Boston, 515-569.
- Rao, K.N. 1992. Optimal synthesis of microcomputers for GM vehicles. Technical report.

- Rinnooy, K., Timmer, G. 1987. Towards global optimization methods (I and II). *Mathematical Programming* **39** 27–78.
- Romeijn, H.E., Smith, R.L. 1994. Simulated annealing for constrained global optimization. *Journal of Global Optimization* **5** 101–126.
- Rotem, D., Schloss, G.A., Segev, A. 1993. Data allocation for multidisk database. *IEEE Transactions on Knowledge and Data Engineering* **5** 882–887.
- Ryoo, H.S., Sahinidis, N.V. 1995. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers and Chemical Engineering* **19** 551–566.
- Salomone, H.E., Iribarren, O.A. 1992. Posynomial modeling of batch plants: A procedure to include process decision variables. *Computers and Chemical Engineering* **16** 173–184.
- Sandgren, E. 1990. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design* **112** 223–229.
- Sarathy, R., Shetty, B., Sen, A. 1997. A constrained nonlinear 0-1 program for data allocation. *European Journal of Operational Research* **102** 626–647.
- Sarje, A.K., Sagar, G. 1991. Heuristic model for task allocation in distributed computer systems. *IEE Proceedings* **E-138** 313–318.
- Sharpe, W. 1971. A Linear Programming Approximation for the General Portfolio Analysis. *Journal of Financial and Quantitative Analysis* **6** 1263–1275.
- Sherali, H., Tuncbilek, C. 1992. A global optimization algorithm for polynomial programming problems using a reformulation linearization technique. *Journal of Global Optimization* **2** 101–112.
- Sinclair, J.B. 1987. Efficient computation of optimal assignments for distributed tasks. *Journal of Parallel and Distributed Computing* **4** 342–362.
- Stone, H.S. 1977. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering* **SE-3** 85–93.
- Tasi, J.F., Li H.L., Hu, N.Z. 2002. Global optimization for signomial discrete programming

problems in engineering design. *Engineering Optimization* **34** 613-622.

Topaloglu, H., W.B. Powell. 2003. An algorithm for approximating piecewise linear concave functions from sample gradients. *Operations Research Letters* **31** 66–76.

Tuy, H. 1998. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Vajda, S. 1964. *Mathematical Programming*. Addison-Wesley, New York.

Young, M.R. 1998. A minimax portfolio selection rule with linear programming solution. *Management Science* **44** 673–683.

