Applied Mathematics, 2012, 3, 1601-1606
http://dx.doi.org/10.4236/am.2012.311221 Published Online November 2012 (http://www.SciRP.org/journal/am)

An Algorithm for Global Optimization Using Formula Manupulation

Tsutomu Shohdohji[1], Fumihiko Yano[2]
[1]Department of Computer and Information Engineering, Nippon Institute of Technology, Saitama, Japan
[2]Division of Integrated Sciences, J. F. Oberlin University, Tokyo, Japan
Email: shodoji@nit.ac.jp, yano@obirin.ac.jp

Received August 13, 2012; revised September 11, 2012; accepted September 18, 2012

ABSTRACT

Constrained nonlinear optimization problems are well known as very difficult problems. In this paper, we present a new algorithm for solving such problems. Our proposed algorithm combines the Branch-and-Bound algorithm and *Lipschitz* constant to limit the search area effectively; this is essential for solving constrained nonlinear optimization problems. We obtain a more appropriate *Lipschitz* constant by applying the formula manipulation system of each divided area. Therefore, we obtain a better approximate solution without using a lot of searching points. The efficiency of our proposed algorithm has been shown by the results of some numerical experiments.

Keywords: Global Optimization; *Lipschitz* Constant; *Lipschitz* Condition; Branch-and-Bound Algorithm; Formula Manipulation

1. Introduction

Many real world problems can be formulated in *mathematical programming problems*, *i.e.*, problems in which an *objective function* that depends on a number of *decision variables* has to be optimized subject to a set of *constraints* [1]. Therefore, methods for solving such problems are of great importance. However, it is very generally difficult to solve a constrained nonlinear optimization problem (hereafter called NLP) and to find its feasible region.

A method that uses the *Lipschitz* condition (*Lipschitz* constant) has been proposed for solving nonlinear optimization problems [2-7]. It is assumed to be difficult to apply this method to multi-dimensional optimization problems because of the curse of the dimension. We have already proposed an effective method for solving the NLP [1,8,9]. This method uses the Branch-and-Bound algorithm and the *Lipschitz* condition, and it guarantees that the obtained solution is optimal.

This paper presents a new algorithm for solving the NLP; the efficiency of this algorithm is improved by using formula manipulation of the objective function and constrained functions.

In this paper, we report that the calculation frequency to obtain a good approximate solution to the NLP has been greatly reduced by using our proposed *Lipschitz* algorithm.

2. Outline of *Lipschitz* Algorithm

2.1. Formulation of the NLP

We consider the following *nonlinear optimization problem* (P):

$$\begin{aligned} &\text{Maximize} \quad f(x), \\ &\text{subject to} \quad x \in S, \end{aligned} \tag{1}$$

where the feasible region S is a compact body, *i.e.*, S is a nonempty, bounded subset of R^n equals to the closure of its interior, the boundary of S has an n-dimensional *Lebesgue* measure equals to zero, and the *objective function* f is a continuous real-valued function defined on S.

We assume that f is a *Lipschitz function* with *Lipschitz* constant K, *i.e.*, f satisfies the *Lipschitz* condition:

$$\left|f(x)-f(y)\right| \le K\left\|x-y\right\|, \quad \forall x, y \in S, \tag{2}$$

where $\|\cdot\|$ is the *Euclidean* norm on R^n.

Let $f^* = \max_{x \in S} f(x)$, and let x^* be an optimal solution so that $f^* = f(x^*)$.

Let us express the NLP described above in slightly greater detail. We can formulate the constrained NLP without loss of generality as follows:

$$\begin{aligned} &\text{Maximize} \quad f(x), \\ &\text{subject to} \quad g_i(x) \ge 0 \quad (1 \le i \le m), \end{aligned} \tag{3}$$

where

$x=(x_1, x_2, \cdots, x_n)$ and $x_k \in [a_k, b_k]\,(k=1,2,\cdots,n)$.

In this formulation, we assume that functions $f(x)$ and $g_i(x)$ are nonlinear real-valued function:

$$\begin{aligned} &f: R^n \to R, \\ &g_i: R^n \to R \quad (i=1,2,\cdots,m), \end{aligned} \tag{4}$$

and *Lipschitzian*. In the case of an equality condition, we reduce it into two equivalent inequality conditions as follows. The rewritten forms of $g_i(x)=0$ are $g_i(x)\ge 0$ and $-g_i(x)\ge 0$.

2.2. Some Definitions and the Basic Theorem of the *Lipschitz* Algorithm

In this section, we describe the notations, some definitions, and the basic theorem of the *Lipschitz* algorithm.

Definition 1 (The *Lipschitz* constant $\mathrm{Lip}(f)$)

If there exists some constant $K \ge 0$ for the function $f: X \to Y$ such that

$$d(f(x), f(x')) \le K d(x, x'), \quad x, x' \in X, \tag{5}$$

the function f is said to satisfy the *Lipschitz* condition.

The lower bound of *K* satisfying the above condition for some function *f* is called the *Lipschitz* constant Lip(*f*) of the function *f*.

We treated solutions of the given constrained NLP as an optimal searching using a type of Branch-and-Bound algorithm [10-13]. Thus, we seek some points that maximize an objective function $f(x)$ in the feasible region Ω, where $\Omega = \{x \mid x \in S, x \in I\}$. We define the constraint set *S* and the *n*-dimensional interval *I* as follows:

$$\begin{aligned} S &= \{x \mid x \in R^n, g_i(x) \ge 0\}, \quad (i=1,2,\cdots,n), \\ I &= [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n], \\ &= \Pi_{i=1}^{n} [a_i, b_i]. \end{aligned} \tag{6}$$

In order to reduce the searching area and improve the efficiency of searching optimal points in the feasible set Ω, the searching algorithm utilizes *Lipschitz* constants of the objective function $f(x)$ and constraint functions $g_i(x)$. In this algorithm, an *n*-dimensional interval *I* is subdivided in each iteration by dividing each $[a_i, b_i]$ by two; therefore, at the *k*-th iteration, each subdivision is also a sub *n*-dimensional interval that has an edge length of $(b_i - a_i)/2^k$ on each *i*-th dimension.

Definition 2 (The maximum radius of the divided area M_k).

The maximum radius (see **Figure 1**) of the divided area at the *k*-th iteration is defined as the radius of the circumscribed sphere of the divided area, *i.e.*,

$$\mathrm{M}_k = \left\{ \sum_{i=1}^{n} \left(\frac{l_i}{2^{k+1}} \right)^2 \right\}^{1/2}, \tag{7}$$

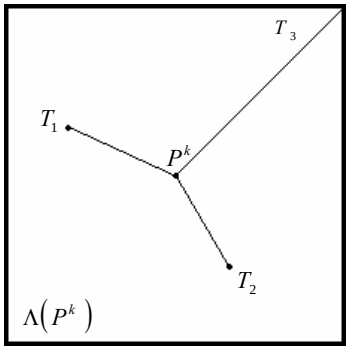

Figure 1. An example of 2-D maximum radius.

where $l_i = b_i - a_i$.

Theorem 1.

Let P_s^k be the central point of the *s*-th divided area at the *k*-th iteration, where $s = 1, 2, \cdots, m_k$, and m_k is the count of subdivisions at the *k*-th iteration. Let $f_s^{(k)}$ be the value of $f(x)$ at P_s^k; $f_{\max}^{(k)}$, the maximum value of $f_s^{(k)}$; and $\mathrm{Lip}(f)$, the *Lipschitz* constant of the function *f*(*x*).

For every divided area $\Lambda(P_s^k)$ at the *k*-th iteration, if there exists $f_s^{(k)}$ such that

$$f_{\max}^{(k)} > f_s^{(k)} + \mathrm{Lip}(f) M_k, \ (s = 1, 2, \cdots, m_k), \tag{8}$$

then the value of *f*(*x*) in $\Lambda(P_s^k)$ does not exceed $f_{\max}^{(k)}$.

Proof. By the definition of M_k and the *Lipschitzian* assumption of $f(x)$,

$$f(R) - f(P_s^k) \le \mathrm{Lip}((f) M_k, \forall R \in \Lambda(P_s^k) \tag{9}$$

holds and then,

$$f(R) \le f(P_s^k) + \mathrm{Lip}((f) M_k. \tag{10}$$

For $f_s^{(k)}$ satisfying Equation (8),

$$f(P_s^k) + \mathrm{Lip}((f) M_k < f_{\max}^{(k)}. \tag{11}$$

Combining Equations (10) and (11), we then obtain the following inequality:

$$f(R) < f_{\max}^{(k)}. \quad \text{(Q. E. D.)}$$

When solving a constrained nonlinear optimization problem, it is practically difficult to effectively handle the feasible region:

$$\Omega = \{x \mid x \in \mathrm{S}, \ x \in I\}, \tag{12}$$

in which the search should be performed.

In order to overcome this difficulty, our approach makes use of a region comprising *n*-dimensional rectangles, the union of which covers Ω, as shown in **Figure 1** for the case of *n* = 2. Instead of the actual feasible set Ω, indicated by the hatched area in **Figure 2**, we treat $\Gamma^{(k)}$, the shaded area shown in **Figure 2**, as an effective constraint set. Therefore, for every *k*-th iteration,

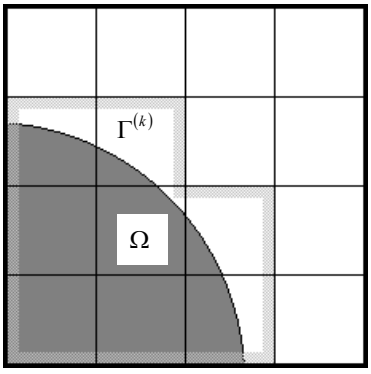

Figure 2. Feasible region Ω and $\Gamma^{(k)}$.

$$\Gamma^{(1)} \supset \Gamma^{(2)} \supset \cdots \supset \Gamma^{(k)} \supset \cdots \supset \Omega \qquad (13)$$

and $\lim_{k\to\infty} \Gamma^{(k)} = \Omega$ hold.

In this context, at the *k*-th iteration, the sub area:

$$S_i' = \left\{ x \in \Lambda\left(P_i^k\right) \;\middle|\; \Lambda\left(P_i^k\right) \cap \Omega \neq \phi,\ P_i^k \notin \Omega \right\}, \quad i = 1, 2, \cdots, m_k \qquad (14)$$

may be excluded from the search space. In order to prevent this discrepancy, irrespective of whether or not the point $P_i^k \in \Lambda\left(P_i^k\right) \subset I$ is satisfied, following the inequality:

$$g\left(P_i^k\right) + \mathrm{Lip}(g) M_k < 0, \qquad (15)$$

the criterion is used for determining if sub area $\Lambda\left(P_i^k\right)$ should be excluded from further searching.

Satisfying Equation (15) implies that sub area $\Lambda\left(P_i^k\right)$ does not satisfy $\mathrm{g}(x) \geq 0$ and therefore it cannot belong to the feasible set. Equation (15) is a sufficient condition for $\mathrm{g}(x) < 0$.

For any U_k, the sub area that does not satisfy both Equations, (8) and (15), at the *k*-th iteration,

$$U_1 \supset U_2 \supset \cdots \supset U_k \supset \cdots \supset \Phi \qquad (16)$$

holds and $\{U_k\}_{k=1}^{\infty}$ is the outer covering for Φ, where Φ is the possible set of solutions for our nonlinear optimization problem (P).

For $|\Phi|$, the cardinal number of Φ, $|\Phi| \geq 1$, and $|\Phi| = \aleph$ when Φ is a continuum. Define $\{u_i\}_{i=1}^{\infty}$ as a sequence comprising u_i, which is some representative point of U_i; then, any subsequence $\{u_i'\}$ of $\{u_i\}_{i=1}^{\infty}$ must have at least one accumulating point and this accumulating point belongs to Φ.

3. Our Proposed Algorithm

Our new algorithm that updates the *Lipschitz* constant in each step is given as follows:

Step 1. Initialization phase.

Step 2. Bisect *n*-dimensional interval $\boldsymbol{I}$.

Step 3. Check if $\Lambda\left(P_i^k\right) \cap S = \phi$ for every $\Lambda\left(P_i^k\right)$ generated in *Step* 2, where $S = \left\{ x \,\middle|\, x \in R^n,\ \mathrm{g}_i(x) \geq 0 \right\}$; then, discard every non-satisfying $\Lambda\left(P_i^k\right)$ and put the remaining $\Lambda\left(P_i^k\right)$ into U_k.

Step 4. Compute $f_i^{(k)}$: value of the objective function at each searching point $P_i^k \in S$.

Step 5. Calculate the tentative solutions such that

$$f_{\max}^{(k)} = \max f_i^{(k)} \quad (i = 1, 2, \cdots, m_k).$$

Step 6. Obtain M_k: maximum radius of divided area for this iteration.

Step 7. Check convergence criterion:

$$\left\{ \mathrm{Lip}(f)\ M_k \Big/ \left| f_{\max}^{(k)} \right| \right\} \leq \varepsilon;$$

if it is satisfied, terminate the iterations and use the tentative solution as the optimum solution.

Step 8. Refine the search space by eliminating every $\Lambda\left(P_i^k\right)$ containing x_i^k such that

$$f_{\max}^{(k)} \leq f\left(x_i^k\right) + \mathrm{Lip}(f) M_k.$$

Step 9. Calculate the *Lipschitz* constant by using formula manipulation considering the result of area judgment in *Step* 8. Then, return to *Step* 2.

4. Experimental Results

In this chapter, the results of two numerical experiments of our proposed algorithm are shown.

The following is a well-known test function called the six-hump camel back function [14].

[Test Function #1]

Objective function:

$$\begin{aligned} \text{Minimize} \quad & f(x,y) = \left(4 - 2.1x^2 + x^4/3\right)x^2 \\ & + xy + \left(-4 + 4y^2\right)y^2, \qquad (17) \\ \text{subject to} \quad & -3 \leq x \leq 3,\ -2 \leq y \leq 2. \end{aligned}$$

The followings are the global minimum value and the optimal solution for this test function respectively.

$$f(x,y) = -1.0316,$$

$$(x,y) = (0.0898, -0.7126),\ (-0.0898, 0.7126).$$

In our computer experiments, we transformed equation (17) and solved it as a maximization problem. We set the following four stopping criteria on our proposed algorithm (using variable *Lipschitz* constant). Therefore, this algorithm stops when it fills one of these four rules.

1) Difference between the upper bound and the lower bound becomes below the ε ($=10^{-6}$),

2) Number of iterations exceeds 200,

3) Execution time exceeds 10 min., and

4) Total number of search points exceeds 10 million.

Figure 3 shows a contour line of the six-hump camel back function. **Table 1** shows the comparison of the calculation efficiency by the difference between the fixed *Lipschitz* constant and the changeable *Lipschitz* constant. This algorithm was coded with Visual C++ (Version 6)®, and the computer experiments were executed on a personal computer equipped with an Intel Pentium IV® 2 GHz processor.

[Test Function #2]

Objective function:

$$\begin{aligned} &\text{Maximize} \quad \exp\left[\cos\left\{2(x+y)\right\}\right]-y, \\ &\text{subject to} \quad x^2+y^2\le 1, \quad (x-1)^2+y^2\le 1, \\ &\qquad x,y\in[-1,1]. \end{aligned} \tag{18}$$

Our proposed algorithm for test function #2 was tested under two conditions:

1) *Lipschitz* constant is fixed at the initial value, and

2) *Lipschitz* constant is recalculated according to the different regions.

This experiment examines if Equation (2) described above remains effective despite repeated calculations of the *Lipschitz* constant.

This numerical experiment was coded in Visual C++ (Version 6)® and executed on a personal computer equipped with an Intel Pentium III® 450 MHz processor.

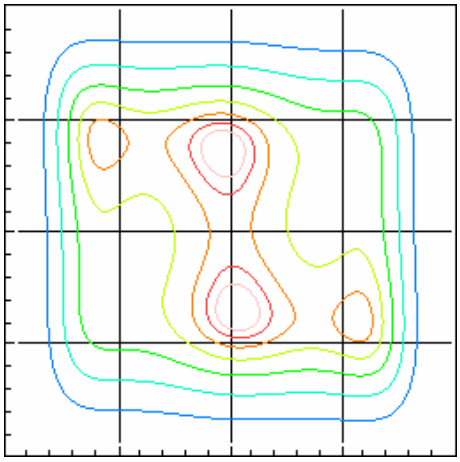

Figure 3. Contour line of objective function.

Table 1. Comparison of two numerical results.

	In case of fixed *Lipschitz* constant	In case of variable *Lipschitz* constant
Computation time in seconds	833.218	0.515
No. of iteration	15	25
Total searching points	15,404,252	3580
Upper bound	1.031629	1.032844
Lower bound	1.031628	1.031628
Maximum value	1.031628	1.031628

In this case, ε in *Step* 7 of the algorithm was set at 10^{-6}. This is done in order to determine if the optimal value of the test function #2 exists in the intersection of the lighter colored part in **Figure 4** and shaded area in **Figure 5**. **Figures 6-8** show the branch-and-bound process of the algorithm without changing the *Lipschitz* constant. The shaded area expresses the divided area left by the *k*-th iteration. These figures show that the divided area converges to an optimal solution in the feasible region. The solution obtained by this algorithm is (*x*,*y*) = (0.68438, –0.72912).

The comparison between the results with and without changing the *Lipschitz* constant is shown in **Table 2**. It shows that the method, in which the *Lipschitz* constant is changed in each iteration, reduces the search area and calculation time. Even if the *Lipschitz* constant is fixed, the same result is obtained. However, the closer the *Lipschitz* constant is to the optimal value, the more advantageous the execution of the branch-and-bound operation is.

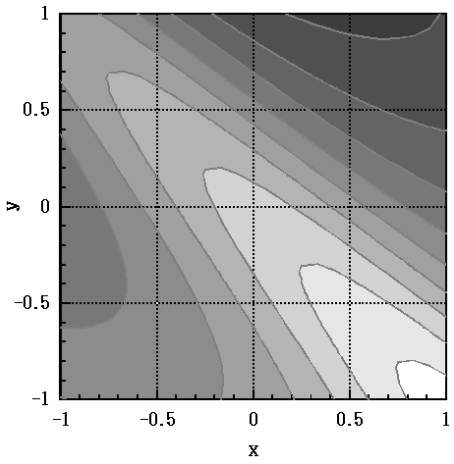

Figure 4. Contour line of test function #2.

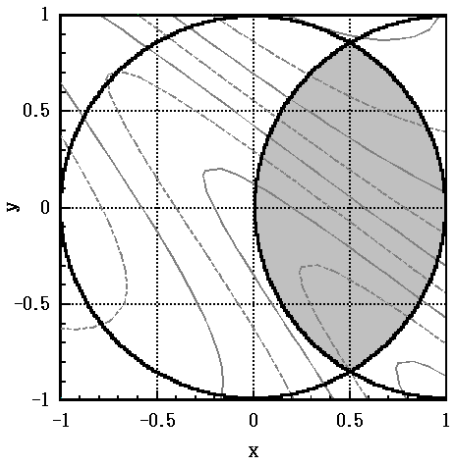

Figure 5. Feasible region.

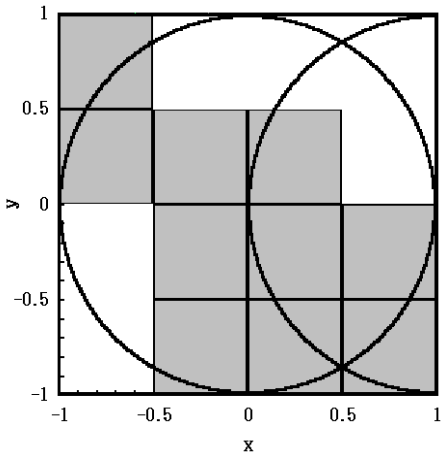

Figure 6. Feasible region at the 3rd iteration.

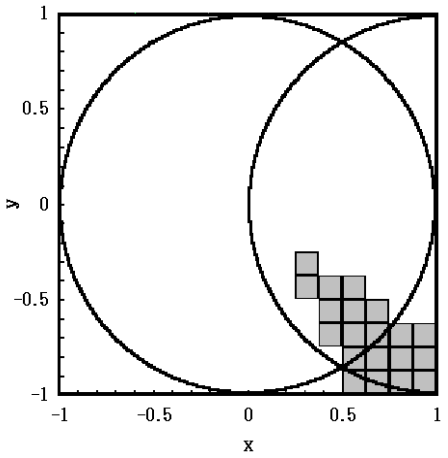

Figure 7. Feasible region at the 5th iteration.

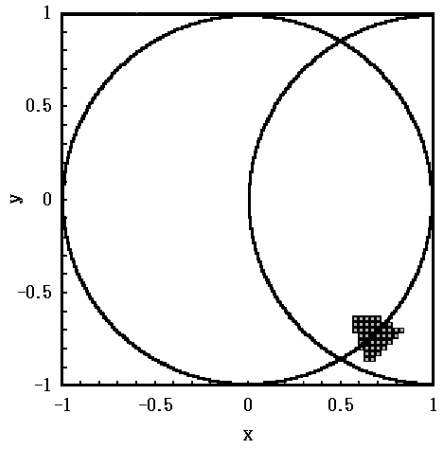

Figure 8. Feasible region at the 7th iteration.

5. Conclusions

Through several computer experiments, we have determined that our proposed algorithm efficiently obtains approximate solutions to the global solution. The inequality in *Step* 8 of the algorithm clearly shows that the branch-and-bound operation is difficult to use when the product of the *Lipschitz* constant and the maximum radius is large. However, the solution always converges to the optimal by *Step* 2.

Table 2. Comparison of the numerical results for test function #2.

	Lipschitz constant is fixed	*Lipschitz* constant is changed
Computation time in seconds	1.430	0.06
The number of iterations	21	19
The total number of points searched	74,104	3408
Optimal solution	(0.684381, –0.729125)	(0.684381, –0.729124)
Maximum value of the objective function	3.43655	

The cost to calculate the *Lipschitz* constant to obtain a better approximate solution at a little computing time is indispensable. It is the most important to decide the best timing that should calculate the *Lipschitz* constant for us to reduce all computing time necessary to obtain the final solution. At present, it is impossible to know the best timing for recalculation of the *Lipschitz* constant. The problem that has been left for the future is how to collect information about the objective function.

In order to improve the efficiency of our proposed algorithm, we will apply *Dynamic Programming* [15] to it in the near future.

6. Acknowledgements

This work was supported by JSPS (Japan Society for the Promotion of Science) Grants-in-Aid for Scientific Research (18510132) and the first author's research was supported in part by the research grants council of NIT (Nippon Institute of Technology). The authors would like to thank Mr. Masao Shinohara, a researcher of NIT, for helpful discussions.

REFERENCES

[1] T. Shohdohji, "An Algorithm for Obtaining a Global Optimum for One Variable Multi-Modal Functions (In Japanese)," *Journal of the Operations Research Society of Japan*, Vol. 19, No. 4, 1976, pp. 295-307.

[2] J. Pinter, "Globally Convergent Methods for *n*-Dimensional Multi-Extremal Optimization," *Optimization*, Vol. 17, No. 2, 1986, pp. 187-202. doi:10.1080/02331938608843118

[3] J. Pinter, "Extended Univariate Algorithms for *n*-Dimensional Global Optimization," *Computing*, Vol. 36, No. 1-2, 1986, pp. 91-103. doi:10.1007/BF02238195

[4] J. Pinter, "Branch-and-Bound Algorithms for Solving

Global Optimization Problems with Lipschitzian Structure," *Optimization*, Vol. 19, No. 1, 1988, pp. 101-110. doi:10.1080/02331938808843322

[5] A. H. G. Rinnooy Kan and G. T. Timmer, "Global Optimization," In: G. L. Nemhauser, A. H. G. Rinnooy Kan and M. J. Todd, Eds., *Handbooks in Operations Research and Management Science*, *Volume* 1: *Optimization*, Elsevier Science Publishers B. V., Amsterdam, 1989, pp. 631-662.

[6] T. Shohdohji, "Global Optimization Algorithm Using Branch-and-Bound Method," *Electrical Proceedings of the* 16*th International Conference on Production Research* (*ICPR*-16), Prague, 9 July-3 August 2001.

[7] B. O. Shubert, "A Sequential Method Seeking the Global Maximum of a Function," *SIAM Journal on Numerical Analysis*, Vol. 9, No. 3, 1972, pp. 379-388. doi:10.1137/0709036

[8] T. Shohdohji, "An Algorithm for Obtaining Global Optima for Multi-Variable Multi-Modal Functions (In Japanese)," *Journal of the Operations Research Society of Japan*, Vol. 20, No. 4, 1977, pp. 311-320.

[9] T. Shohdohji and Y. Yazu, "A New Algorithm for Nonlinear Programming Problem," *Proceedings of International Workshop on Intelligent Systems Resolutions—The* 8*th Bellman Continuum*, National Tsing-Hua University, Hsinchu, Taiwan, 11-12 December 2000, pp. 229-233.

[10] T. Ibaraki, "On the Computational Efficiency of Branch-and-Bound Algorithms," *Journal of Operations Research Society of Japan*, Vol. 20, No. 1, 1977, pp. 16-35.

[11] T. Ibaraki, "Branch-and-Bound Procedure and State—Space Representation of Combinatorial Optimization Problems," *Information and Control*, Vol. 36, No. 1, 1978, pp. 1-27. doi:10.1016/S0019-9958(78)90197-3

[12] E. L. Lawler and D. E. Wood, "Branch-and-Bound Methods: A Survey," *Operations Research*, Vol. 14, No. 4, 1966, pp. 699-719. doi:10.1287/opre.14.4.699

[13] T. L. Morin and R. E. Marsten, "Branch-and-Bound Strategies for Dynamic Programming," *Operations Research*, Vol. 24, No. 4, 1976, pp. 611-627. doi:10.1287/opre.24.4.611

[14] L. C. W. Dixon and G. P. Szego, Eds., "Towards Global Optimization 2," North-Holland Publishing Company, Amsterdam, 1978, p. 97.

[15] A R. E. Bellman, "Dynamic Programming," Princeton University Press, Princeton, New Jersey, 1957.

