An Improved Surrogate Constraints Method for Separable Nonlinear Integer Programming

著者	仲川 勇二
journal or publication title	Journal of the Operations Research Society of Japan
volume	46
number	2
page range	145-163
year	2003-06
権利	(C) The Operations Research Society of Japan 2003, All rights reserved, Original Text is available at http://ci.nii.ac.jp/els/110001183565.pdf?id=ART0001514088&type=pdf&lang=jp&host=cinii&order_no=&ppv_type=0&lang_sw=&no=1352269014&cp=
URL	http://hdl.handle.net/10112/7432

Journal of the Operations Research
Society of Japan
2003, Vol. 46, No. 2, 145-163

AN IMPROVED SURROGATE CONSTRAINTS METHOD FOR SEPARABLE NONLINEAR INTEGER PROGRAMMING

Yuji Nakagawa
Kansai University

(Received Received February 6, 2002; Revised December 2, 2002)

Abstract An improved surrogate constraints method for solving separable nonlinear integer programming problems with multiple constraints is presented. The surrogate constraints method is very effective in solving problems with multiple constraints. The method solves a succession of surrogate constraints problems having a single constraint instead of the original multiple constraint problem. A surrogate problem with an optimal multiplier vector solves the original problem exactly if there is no duality gap. However, the surrogate constraints method often has a duality gap, that is it fails to find an exact solution to the original problem. The modification proposed closes the surrogate duality gap. The modification solves a succession of target problems that enumerates all solutions hitting a particular target. The target problems are produced by using an optimal surrogate multiplier vector. The computational results show that the modification is very effective at closing the surrogate gap of multiple constraint problems.

Keywords: Combinatorial optimization, discrete optimization, separable nonlinear integer programming, multidimensional knapsack, nonlinear knapsack, surrogate constraints

1. Introduction

This paper presents a new surrogate constraints method [25] for solving a separable nonlinear integer problem with multiple constraints. The existing surrogate constraints method solves a succession of surrogate problems, which have a single constraint, instead of the original multiple constraint problem. However, the surrogate constraints method often has a duality gap, that is it fails to produce an exact optimal solution of the original problem. The new surrogate constraints method can close the surrogate duality gap.

Consider the separable nonlinear integer programming problem:

$$\begin{aligned} [\mathrm{P}]: \quad \text{Maximize} \quad & f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & g_j(\mathbf{x}) = \sum_{i=1}^{n} g_{ji}(x_i) \le b_j \quad \text{for } j = 1, 2, \ldots, m, \\ & x_i \in K_i \quad \text{for } i = 1, 2, \ldots, n, \end{aligned}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and K_i is a set of items, $K_i = \{1, 2, \ldots, k_i\}$, for $i = 1, 2, \ldots, n$. We assume all the problem data is non-negative. If a problem includes negative data, then the problem can be equivalently transformed into a non-negative data problem by resetting

$$f_i(k) \leftarrow f_i(k) - \min_{t=1,2,\ldots,k_i} \{f_i(t)\} \quad (k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n),$$

$$g_{ji}(k) \leftarrow g_{ji}(k) - \min_{t=1,2,\ldots,k_i} \{g_{ji}(t)\} \quad (k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m),$$

and

$$b_j \leftarrow b_j - \sum_{i=1}^{n} \min_{t=1,2,\ldots,k_i} \{g_{ji}(t)\} \quad (j = 1, 2, \ldots, m).$$

Note that Problem [P] is equivalent to the following 0-1 integer programming problem

$$\begin{aligned}
[\mathrm{P}^{\mathrm{IP}}]: \quad \text{Maximize} \quad & \sum_{i=1}^{n} \sum_{k=1}^{k_i} c_{ik} y_{ik} \\
\text{subject to} \quad & \sum_{i=1}^{n} \sum_{k=1}^{k_i} a_{jik} y_{ik} \le b_j \quad \text{for } j = 1, 2, \ldots, m, \\
& \sum_{k=1}^{k_i} y_{ik} = 1 \quad \text{for } i = 1, 2, \ldots, n, \\
& y_{ik} = 0 \text{ or } 1 \quad \text{for } k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n,
\end{aligned}$$

where $c_{ik} = f_i(k),\ a_{jik} = g_{ji}(k)$.

Morin and Marsten [17] called [P] the Multi-dimensional Nonlinear Knapsack (MNK) problem. Several special cases of the MNK problem include 1)Nonlinear Resource Allocation Problem (Bretthauer and Shetty [2]) which has a differentiable convex objective and constraint functions, 2)some resource allocation problems, e.g. Ibaraki and Katoh [9], which have a convex objective function and the single constraint of the sum of variables, and 3)Multiple-Choice Knapsack Problem (Nauss [26]) which is a linear problem of the singly constrained MNK problem. The surrogate constraints were first used in mathematical programming by Glover [7]. Luenberger[13] showed that any quasi-convex programming problems could be solved exactly if the surrogate multipliers are correctly chosen. Karwan and Rardin [11] provided some empirical evidence on the effectiveness of surrogate constraints in integer linear programming.

There are several algorithms for yielding an optimal surrogate multiplier vector to a MNK problem. Dyer [5] proposed two algorithms: one analogous to generalized linear programming and the other to the subgradient method. Nakagawa et al. [20, 21] proposed a Cutting-Off Polyhedron (COP) algorithm that generates a succession of multiplier vectors by cutting off polyhedrons and then using the center of the polyhedron as the next multiplier vector. Karwan, Rardin and Sarin [12] presented a measure (percent gap closure) of the quality of the computed surrogate dual problem with respect to the linear programming relaxation.

Unfortunately, most of MNK problems include surrogate duality gaps. In other words, even if a surrogate constraints method solves a surrogate problem with an optimal surrogate multiplier vector, the method fails to produce a feasible optimal solution of the original multi-constrained problem. A new method, which we call the Slicing Algorithm (SA), is proposed to search for exact optimal solutions to the original multi-constrained problem from the feasible region of the optimal surrogate problem. The SA solves a succession of target problems that enumerate all solutions in a slice of the feasible region. The target problems can be solved by using a MA (Modular Approach) proposed by Nakagawa [22]. The MA executes the following steps (1) and (2), repeatedly.

(1) Apply the fathoming test to the current problem to reduce the decision space of the variables.

(2) Combine two variables into one new variable to reduce the number of variables in the current problem.

To solve a target problem exactly, we can use the feasibility test and the bounding test as the fathoming test, but not the dominance test.

Sometimes we have difficulty in enumerating all the solutions to the target problem since the target solution space is too large. The SA uses two techniques to reduce the solution space size. One technique is to thin out the target solution space. Another is to narrow the feasible region of target problem by reducing the value of the right-hand-side. Computational results with the SA are presented in Section 5.

The rest of the paper is organized as follows. Section 2 describes the surrogate dual problem and its properties. Section 3 presents an approach for closing the surrogate gap. Section 4 illustrates the SA by using an example from the literature. Section 5 presents some computational experiments using with test problems from the literature and randomly generated large-scale test problems. Section 6 makes some conclusions from these experiments.

2. Surrogate Dual

A surrogate problem $[\mathrm{P}^{\mathrm{S}}(\mathbf{u})]$ corresponding to the original problem [P] is written as follows:

$$\begin{aligned} [\mathrm{P}^{\mathrm{S}}(\mathbf{u})]: \quad & \text{Maximize} \quad f(\mathbf{x}) \\ & \text{subject to} \quad \mathbf{ug}(\mathbf{x}) \leq \mathbf{ub}, \\ & \qquad\qquad\quad \mathbf{x} \in \mathbf{K}, \end{aligned}$$

where

$$\begin{aligned} \mathbf{u} &= (u_1, u_2, \ldots, u_m), \\ \mathbf{g}(\mathbf{x}) &= (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})), \\ \mathbf{b} &= (b_1, b_2, \ldots, b_m), \\ \mathbf{K} &= \{\mathbf{x} : x_i \in K_i \text{ for } i = 1, 2, \ldots, n\}. \end{aligned}$$

The surrogate dual problem is defined by

$$[\mathrm{P}^{\mathrm{SD}}]: \ \min\left\{v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u})] \ : \mathbf{u} \in \mathbf{U}\right\}$$

where $v^{\mathrm{OPT}}[\,\bullet\,]$ means the optimal objective function value of problem $[\,\bullet\,]$ and

$$\mathbf{U} = \left\{\mathbf{u} \in R^m : \sum_{j=1}^{m} u_j = 1, \mathbf{u} \geq \mathbf{0}\right\}.$$

The surrogate problem $[\mathrm{P}^{\mathrm{S}}(\mathbf{u})]$ has the following property.

Property 1 Let $\mathbf{x}^q$ be an optimal solution of a problem $[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^q)]$ for a surrogate multiplier vector $\mathbf{u}^q \in \mathbf{U}$. For any $\mathbf{u} \in \mathbf{U}$ such that $\mathbf{ug}(\mathbf{x}^q) \leq \mathbf{ub}$, it holds that

$$v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u})] \geq f(\mathbf{x}^q).$$

Proof It is clear, since $[\mathrm{P}^{\mathrm{S}}(\mathbf{u})]$ includes $\mathbf{x}^q$ as a feasible solution.

This property means that the region $\{\mathbf{u} \in \mathbf{U} : \mathbf{ug}(\mathbf{x}^q) \leq \mathbf{ub}\}$ can be removed from U. An algorithm [21] using this property can solve the surrogate dual $[\mathrm{P}^{\mathrm{SD}}]$ exactly. The algorithm

starts with the polyhedron $\mathbf{U}^1 = \mathbf{U}$. Using the centroid of vertices the $q-$th polyhedron $\mathbf{U}^q$ as a surrogate multiplier $\mathbf{u}^q$, we get an optimal solution $\mathbf{x}^q$ of the surrogate problem $[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^q)]$ and then have a cutting plane $\mathbf{ug}(\mathbf{x}^q) \geq \mathbf{ub}$. The hyperplane cuts the polyhedron $\mathbf{U}^q$ to generate the next polyhedron $\mathbf{U}^{q+1}$. Finally we have a finite set of multiplier vectors $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^q$ that covers the whole of U from Property 1. The multiplier vector $\mathbf{u}^*$ is defined such that

$$v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^*)] = \min\left\{ v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^1)],\ v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^2)],\ \ldots,\ v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^q)]\right\}$$

is optimal to the surrogate dual $[\mathrm{P}^{\mathrm{SD}}]$.

Property 2 If an optimal solution $\mathbf{x}^{\mathrm{SD}}$ of the problem $[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^*)]$ is feasible to the original multi-constrained problem [P], then $\mathbf{x}^{\mathrm{SD}}$ is an exact optimal solution of [P].

Proof Since the feasible region of $[\mathrm{P}^{\mathrm{S}}(\mathbf{u}^*)]$ includes that of [P], $\mathbf{x}^{\mathrm{SD}}$ is an exact optimal to [P].

When the succession of surrogate problems provides no feasible solutions of [P], it is said that there exists a surrogate duality gap. In this case, the value $f(\mathbf{x}^{\mathrm{SD}})$ is an upper bound on the optimal objective function value of [P].

3. Resolution of the Surrogate Duality GAP

To close the surrogate duality gap, consider the target problem (Nakagawa et al. [19] and Miyaji et al. [16]):

$$\begin{aligned}[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}}, b^{\mathrm{T}})]: \quad & \text{Enumerate all solutions } \mathbf{x} \text{ hitting} \\ & \text{a target} \quad f(\mathbf{x}) \geq f^{\mathrm{T}} \\ & \text{subject to} \quad \mathbf{u}^*\mathbf{g}(\mathbf{x}) \leq b^{\mathrm{T}}, \\ & \qquad\qquad \mathbf{x} \in \mathbf{K},\end{aligned}$$

where $\mathbf{u}^*$ is an optimal multiplier vector of a surrogate dual problem $[\mathrm{P}^{\mathrm{SD}}]$ corresponding to the original multi-constrained problem [P]. The feasible solutions hitting the target are called target solutions.

Property 3 If $f^T \leq v^{\mathrm{OPT}}[\mathrm{P}]$, then all exact optimal solutions of [P] are target solutions to the target problem $[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}}, \mathbf{u}^*\mathbf{b})]$.

Proof All of the exact optimal solutions of [P] hit the target $f(\mathbf{x}) \geq f^T$ and are feasible to the surrogate constraint $\mathbf{u}^*\mathbf{g}(\mathbf{x}) \leq \mathbf{u}^*\mathbf{b}$, since $f^T \leq v^{\mathrm{OPT}}[\mathrm{P}]$ and any feasible solution of [P] satisfies $\mathbf{u}^*\mathbf{g}(\mathbf{x}) \leq \mathbf{u}^*\mathbf{b}$, respectively.

Property 3 means that if $f^T \leq v^{\mathrm{OPT}}[\mathrm{P}]$ and we can enumerate every target solution of problem $[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}}, \mathbf{u}^*\mathbf{b})]$, then we have all the exact optimal solutions of [P]. If we have a heuristic method that produces near-optimal solutions of good quality, then we can use the near-optimal value as the target value f^T of $[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}}, \mathbf{u}^*\mathbf{b})]$. However, if we use a poor near-optimal solution value as the target value f^T, then it may waste a large amount of computational time and memory.

When we have poor quality estimate of the optimal value, the target values f^T will be chosen from an interval such as $f(\mathbf{x}^{\mathrm{Near}}) \leq f^T \leq f(\mathbf{x}^{\mathrm{SD}})$, where $\mathbf{x}^{\mathrm{Near}}$ is an estimate of the optimal solution of [P]. A near optimal solution can be obtained by using an existing heuristic algorithm, e.g., Ohtagaki et al. [27]. The solutions hitting a target are called target solutions. Implementing the MA without dominance testing can solve the target problem

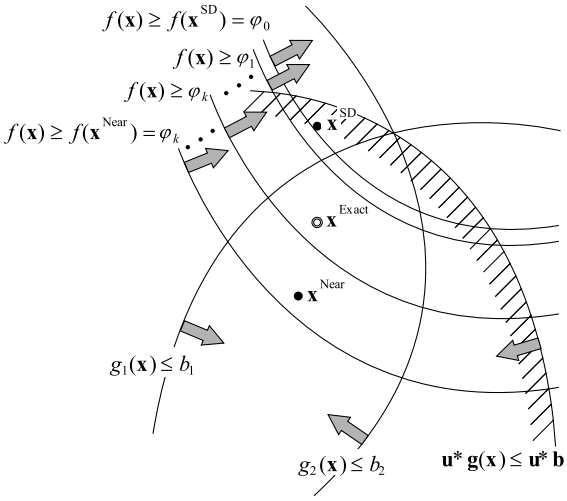

Figure 1 Exact case of the SA

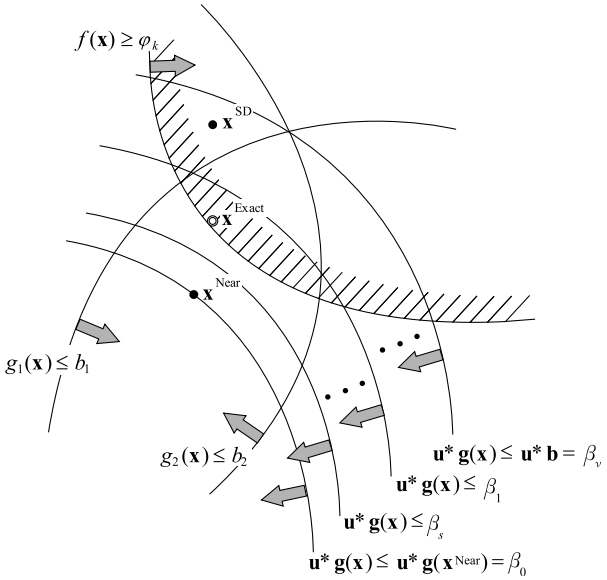

Figure 2 Heuristic case of the SA

$[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}},\ \mathbf{u}^*\mathbf{b})]$ exactly using the SA. The problem $[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}},\ \mathbf{u}^*\mathbf{b})]$ becomes harder to solve with a decreasing value of f^{T}, because of the increasing number of target solutions.

Consider a succession of $\kappa + 1$ problems $[\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}},\ \mathbf{u}^*\mathbf{b})]$ with $f^{\mathrm{T}} = \varphi_0,\ \varphi_1, \ldots,\ \varphi_\kappa$, where

$$\varphi_t = \frac{(\kappa - t)f(\mathbf{x}^{\mathrm{SD}}) + tf(\mathbf{x}^{\mathrm{Near}})}{\kappa} \quad (t = 0, 1, 2, \ldots, \kappa).$$

These target problems are solved in the order $[\mathrm{P}^{\mathrm{T}}(\varphi_0, \mathbf{u}^*\mathbf{b})],\ldots,\ [\mathrm{P}^{\mathrm{T}}(\varphi_\kappa, \mathbf{u}^*\mathbf{b})]$ using the SA, as shown in Figure 1. When the SA finds an exact optimal solution to [P] out of the target solutions, the algorithm stops. However, the problems become harder to solve as the problem size increases. There are two main reasons for this. The first reason is that there exist too many target solutions with the same optimal objective function value. The second reason is that the target region is too wide. In the SA, two techniques are used to decrease these difficulties. One technique is to thin out the target solutions. When the number of items for a variable exceeds a certain threshold value in the MA code, a thinning-out technique is carried out. Another technique is to slice off piece-by-piece parts of the feasible region by changing the value of right-hand-side b^{T}. It should be noted that the solutions obtained by using these techniques cannot be guaranteed to be exact optimal to the original multi-constrained problem [P].

Consider a succession of $\nu + 1$ problems $[\mathrm{P}^{\mathrm{T}}(\varphi_k, b^{\mathrm{T}})]$ with $b^{\mathrm{T}} = \beta_0, \beta_1, \ldots, \beta_\nu$, where

$$\beta_s = \frac{(\nu - s)\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Near}}) + s\mathbf{u}^*\mathbf{b}}{\nu} \quad (s = 0, 1, 2, \ldots, \nu).$$

The method SA tries to solve the problems $[\mathrm{P}^{\mathrm{T}}(\varphi_k, \beta_s)]$ in the order of $s = 0, 1, 2, \ldots, \nu$. Figure 2 illustrates this technique. An algorithm for producing an exact optimal solution of the original multi-constrained problem [P] is as follows:

Algorithm SA

1. Solve the surrogate dual problem [P^{SD}] and obtain $\mathbf{u}^*$ and $\mathbf{x}^{\mathrm{SD}}$;
2. **If** $\mathbf{x}^{\mathrm{SD}}$ is feasible to the constraints of [P] **then**
3. Set $\mathbf{x}^{\mathrm{Exact}} \leftarrow \mathbf{x}^{\mathrm{SD}}$;
4. Terminate this algorithm;
5. **Else**
6. Obtain a near-optimal solution $\mathbf{x}^{\mathrm{Near}}$ of [P] by using a heuristic algorithm;
7. **EndIf**
8. Set $b^{\mathrm{T}} \leftarrow \mathbf{u}^*\mathbf{b}$ and read a division number κ;
9. **For** $t = 0, 1, \ldots, \kappa$
10. Set $\varphi_t \leftarrow \frac{(\kappa - t)f(\mathbf{x}^{\mathrm{SD}}) + tf(\mathbf{x}^{\mathrm{Near}})}{\kappa}$;
11. Enumerate the target solutions of [$\mathrm{P}^{\mathrm{T}}(\varphi_t,\ \mathbf{u}^*\mathbf{b})$] by using MA;
12. **If** there exists a target solution $\mathbf{x}^{\mathrm{T}}$ feasible to the constraints of [P] **then**
13. Set $\mathbf{x}^{\mathrm{Slc}} \leftarrow \mathbf{x}^{\mathrm{T}}$ and $f^{\mathrm{Slc}} \leftarrow f(\mathbf{x}^{\mathrm{T}})$; Exit this **For** loop;
14. **EndIf**
15. **EndFor**
16. Write the obtained solution $\mathbf{x}^{\mathrm{Slc}}$ and f^{Slc}, which is an exact optimal solution;

4. Example

The SA will be demonstrated with a simple example, which is a variation of the Rosen-Suzuki problem [29]. The example problem is:

$$
\begin{aligned}
[\mathrm{P}^{\mathrm{Ex}}] : \max \quad & f(\mathbf{x}) = -x_1^2 - x_2^2 - 2x_3^2 - x_4^2 + 5x_1 + 5x_2 + 21x_3 - 7x_4 \\
\text{s. t.} \quad & \mathrm{g}_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 \le 5, \\
& \mathrm{g}_2(\mathbf{x}) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 \le 5, \\
& \mathrm{g}_3(\mathbf{x}) = 2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 \le 5, \\
& -9 \le x_i \le 10 \ \text{ for } \ i = 1, 2, 3, 4, \\
& x_i \ \text{ integer for } \ i = 1, 2, 3, 4.
\end{aligned}
$$

A COP (Cutting-Off Polyhedron) algorithm proposed by Nakagawa et al. [21] is started with a surrogate multiplier vector $\mathbf{u}^1 = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, as shown in Figure 3. The surrogate subproblem is [$\mathrm{P}^{\mathrm{S}}(\mathbf{u}^1)$] : max $f(\mathbf{x})$ subject to $\mathbf{u}^1\mathbf{g}(\mathbf{x}) \le \mathbf{u}^1\mathbf{b}$ and $-9 \le x_i \le 10$, x_i integer for i= 1, 2, 3, 4. Using the MA produces an optimal solution $\mathbf{x}^1$ =(0, 0, 2, 0), $f(\mathbf{x}^1)$ =34, and $\mathbf{g}(\mathbf{x}^1)$ =(6, 4, 4), to the surrogate problem [$\mathrm{P}^{\mathrm{S}}(\mathbf{u}^1)$]. We have the first cutting plane $2u_1 + 0u_2 \ge 1$ from $\mathbf{ug}(\mathbf{x}^1) \ge \mathbf{ub}$ and $u_3 = 1 - u_1 - u_2$. The COP algorithm generates $\mathbf{u}^2$ =(0.666667, 0.166667, 0.166667) which is the center of balance of a material-points-system that has an unit weight at every vertex. Similarly, the surrogate constraints method generates $\mathbf{x}^2$ =(0, 1, 1, -1), $f(\mathbf{x}^2) = 29$, $\mathbf{g}(\mathbf{x}^2)$ =(4, 6, 2), and the last cutting plane $2u_1 + 4u_2 \ge 3$. This plane cuts off the remaining region of U as shown in Figure 3. Therefore the optimal surrogate multiplier is $\mathbf{u}^* = \mathbf{u}^2$, an optimal solution of the surrogate dual problem is $\mathbf{x}^{\mathrm{SD}} = \mathbf{x}^2$, $f(\mathbf{x}^{\mathrm{SD}}) = 29$, from $f(\mathbf{x}^{\mathrm{SD}}) = \min\{f(\mathbf{x}^1),\ f(\mathbf{x}^2)\}$. However, $\mathbf{x}^{\mathrm{SD}}$ is infeasible to the second constraint $g_2(\mathbf{x}) \le b_2$ of the original problem [P^{Ex}]. There exists a surrogate duality gap.

In order to close this surrogate gap, consider a succession of target problems [$\mathrm{P}^{\mathrm{T}}(f^{\mathrm{T}}, \beta^{\mathrm{T}})$] with $\beta^{\mathrm{T}} = \mathbf{u}^*\mathbf{b} = 5$ and $f^{\mathrm{T}} = 29, 28, 27$, where a near optimal solution $\mathbf{x}^{\mathrm{Near}} = (1, 1, 1, 0)$,

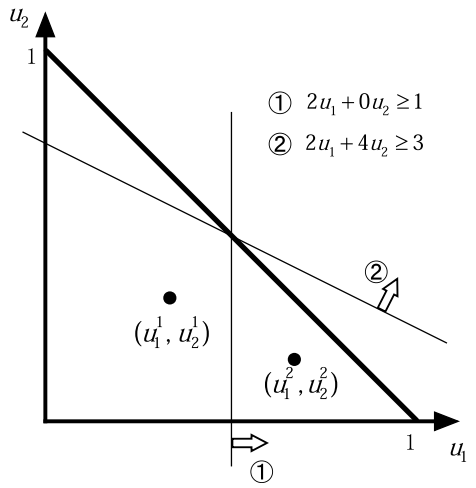

Figure 3 Surrogate multipliers and cutting planes

$f(\mathbf{x}^{\text{Near}}) = 27$ by a heuristic method. The succession of target problems are solved by the SA. The problems $[\text{P}^T(29, 5)]$, $[\text{P}^T(28, 5)]$ have no target solutions satisfying the constraints of $[\text{P}^{\text{Ex}}]$. The problem $[\text{P}^T(27, 5)]$ provides a target solution $[\text{P}^{\text{Slc}}]$ =(1, 1, 1, 0) satisfying all constraints of $[\text{P}^{\text{Ex}}]$. This target solution is guaranteed to be the optimal solution of the original multi-constrained problem $[\text{P}^{\text{Ex}}]$.

5. Computational Experience

The performance of the SA has been evaluated on the following four groups of test problems. Group 1 are test problems from the literature and variations of them. Group 2, 3 and 4 sets are randomly generated using a machine-independent random number generator (Marse and Roberts [14]).

Group 1:

Problem 1: The following capital budgeting problem with 39 zero/one variables (Problem 6 in Petersen [28]):

$$\text{Maximize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_{ji} x_i \leq b_j \quad \text{for } j = 1, 2, \ldots, 5,$$

$$x_i = 0 \text{ or } 1 \quad \text{for } i = 1, 2, \ldots, n,$$

where the data c_i and a_{ji} are given in Table 5 of Appendix.

Problem 2: The capital budgeting test problem with 50 zero/one variables (Problem 7 in Petersen [28]).

Problem 3: The same problem as Problem 1 except for the variable restriction:

$$x_i = 0,\ 1,\ \ldots,\ \text{or } 5 \quad \text{for } i = 1, 2, \ldots, 39,$$

instead of $x_i = 0$ or 1 for $i = 1, 2, \ldots, 39$.

Problem 4: The same problem as Problem 2 except for the variable restriction:

$$x_i = 0,\ 1,\ \ldots,\ \text{or } 5 \quad \text{for } i = 1, 2, \ldots, 50,$$

instead of $x_i = 0$ or 1 for $i = 1, 2, \ldots, 50$.

Problem 5: Maximize a nonlinear objective function $\sum_{i=1}^{50} (c_i x_i + a_{1i} x_i^2 + a_{2i} x_i^3)$, subject to the same constraints as Problem 4, where coefficients c_i, a_{1i}, a_{2i} are the same values as Problem 2. This objective function is from Cooper [4].

Problem 6: Maximize the same objective function as Problem 5, subject to the nonlinear constraints $\sum_{i=1}^{50} (a_{ji} x_i^2 - a_{j+1,i} x_i) \le b_j$ for j = 1,2,3,4, and $x_i = 0, 1, 2, \ldots,$ or 10 for $i = 1, 2, \ldots, 50$, where coefficients a_{ji} are the same values as Problem 2. The constraint function is from Bretthouer and Shetty [2].

Group 2:

Problem 1-30: All of the objective and constraint functions of [P] are nonlinear and monotone increasing:

$$0 \le f_i(k) < f_i(k+1) \le 256 k_i \quad (k = 1, 2, \ldots, k_i - 1,\ i = 1, 2, \ldots, n),$$

$$0 \le g_{ji}(k) < g_{ji}(k+1) \le 256 k_i \quad (k = 1, 2, \ldots, k_i - 1,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m),$$

$f_i(k)$, $g_{ji}(k)$ are all integers, and

$$b_j = \left\lfloor \frac{1}{2} \sum_{i=1}^{n} (g_{ji}(1) + g_{ji}(k_i)) \right\rfloor \quad (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m).$$

Group 3:

Problem 1-5: All of the objective and constraint functions of [P] are nonlinear and monotone increasing:

$$\begin{aligned} 0 &\le f_i(k) < f_i(k+1) \le 8 k_i \quad (k = 1, 2, \ldots, k_i - 1,\ i = 1, 2, \ldots, n), \\ 0 &\le g_{ji}(k) < g_{ji}(k+1) \le 8 k_i \quad (k = 1, 2, \ldots, k_i - 1,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m), \end{aligned}$$

$f_i(k)$, $g_{ji}(k)$ are all integers, and

$$b_j = \left\lfloor \frac{1}{2} \sum_{i=1}^{n} (g_{ji}(1) + g_{ji}(k_i)) \right\rfloor \quad (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m).$$

Group 4:

Problem 1-30: All of the objective and constraint functions of [P] are nonlinear and non-monotone:

$$\begin{aligned} 0 &\le f_i(k) \le 256(k_i - 1) \quad (k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n), \\ 0 &\le g_{ji}(k) \le 256(k_i - 1) \quad (k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m), \end{aligned}$$

$f_i(k), g_{ji}(k)$ are all integers, and

$$b_j = \left\lfloor \frac{1}{2} \sum_{i=1}^{n} (g_{ji}(1) + g_{ji}(k_i)) \right\rfloor (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m).$$

and reset

$$\begin{aligned} f_i(k) &\leftarrow f_i(k) - \min_{t=1,2,\ldots,k_i}\{f_i(t)\} \quad (i = 1, 2, \ldots, n), \\ g_{ji}(k) &\leftarrow \gamma_{ji} g_{ji}(k) \quad (k = 1, 2, \ldots, k_i,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m), \end{aligned}$$

where γ_{ji} are zero/one variables that set so that 25% are zero elements. Problems 1-30 in Group 2 set and Problems 1-5 in Group 3 set are generated referring to test problems presented in Sinha and Zoltners [30] Problems 1-30 in Group 4 set are generated with 75% nonzero elements as used by Karwan, Rardin and Sarin [12].

All algorithms are coded in a Modula 2- like C language developed by Nakagawa [23]. This language is oriented to scientific calculations, implemented by using the macro expansion of the C preprocessor, and compiled by a C compiler. The algorithms were tested on a Windows computer (Pentium 4/1.7GHz) with the Code Warrior C compiler. The algorithms implement an upper-bound calculation technique presented by Sinha and Zoltners [30] is used. When the algorithm solves a succession of target problems, the bounding test in the code utilizes not only the constraint of the optimal surrogate problem, but also the constraints of the original multi-constrained problem [P]. The algorithm is a modification of a MA algorithm used in Nakagawa and Iwasaki [24]. The calculations of all the real-valued numbers in this code have been executed in double precision (64 bits).

Karwan, Rardin and Sarin [12] have provided a measure of Percent Gap Closure (PGC). The PGC value is defined as:

$$\frac{f^{\mathrm{Real}} - v^{\mathrm{OPT}}[\mathrm{P}^{\mathrm{SD}}]}{f^{\mathrm{Real}} - v^{\mathrm{OPT}}[\mathrm{P}]} \times 100$$

where f^{Real} is the linear programming relaxation of the linear integer programming problem corresponding to the original problem [P]. This paper adopts the PGC value to measure the difficulty of the original problem [P]. A small PGC value means that the surrogate duality gap is wide. The value 100% of PGC suggests that the optimal function value of [P^{SD}] is equivalent to the optimal value of [P].

The near-optimal solutions of test problems were generated by a heuristic method (a smart greedy algorithm). The greedy algorithm was proposed by Ohtagaki et al. [27] and is an improvement on a heuristic algorithm by Nakagawa et al. [18]. The greedy algorithm generates several greedy solutions by using multiple greedy criteria designated by a balance coefficient between the objective function and constraints. It then chooses the best solution among greedy solutions as a smart greedy solution.

The computational times for $\mathbf{x}^{\mathrm{Slc}}$ indicated in tables 1, 2, 3 and 4 are the CPU seconds taken by the SA to find the solution $\mathbf{x}^{\mathrm{Slc}}$. The target values used by the SA are the objective function values of the best known near-optimal solutions, most of which are exact optimal values. It should be noted that these computational times are actual times if we assume to use a heuristic algorithm for producing very good near-optimal solutions. The target value f^{NS} is the minimum value whose target problem can be solved exactly by SA and have been constructed such that there are no feasible solutions to the original multi-constrained problem.

These test problems are available publicly from the Web site:

`http://www.res.kutc.kansai-u.ac.jp/~nakagawa/orlib/jorsj/.`

Tables 1a and 1b report the computational results for the six test problems of Group 1. Table 1a gives the optimal solutions to the test problems. The obtained solution $\mathbf{x}^{\mathrm{Slc}}$ for

Problems 1 and 2 are the same as the optimal solutions in Petersen [28]. Computational results for the performance of the SA are shown in table 1b. Since Problem 3 has a wide gap in the surrogate duality (the PGC value of Problem 3 is 2.5-4.6%). The SA fails to yield a solution guaranteed to be exactly optimal to the original problem [P]. However, a near-optimal solution, $f(\mathbf{x}^{\mathrm{Slc}}) = 14{,}296$, has been obtained, which is better than the near-optimal solution, $f(\mathbf{x}^{\mathrm{Near}}) = 13{,}981$, yielded by the smart greedy algorithm. Except for Problem 3, the SA has succeeded in finding the exact optimal solutions for all problems in Group 1.

Table 1a Computational results for test problems in Group 1

Problem No.	Computational Results (solutions by the Slicing Algorithm)
1	b= (600, 500, 500, 500, 600), $\mathbf{u}^*$=(0.413040, 0.150670, 0.090110, 0.000333, 0.345848), $\mathbf{u}^*\mathbf{b}$= 575.89, $\mathbf{x}^{\mathrm{Slc}}$=(1,1,0,1,0,1,0,1,1,0,1,0,1,0,1,1,1,1,1,1,0,0,1,0,1,0,1,1,1,0,1,1,0,1,1,1,1,1,1), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 573.39, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (597, 496, 493, 427, 600).
2	b= (800, 650, 550, 550, 650), $\mathbf{u}^*$=(0.444075, 0.113238, 0.136335, 0.000004, 0.306349), $\mathbf{u}^*\mathbf{b}$= 702.98, $\mathbf{x}^{\mathrm{Slc}}$=(0,0,0,1,0,1,0,1,1,0,1,1,1,0,1,1,1,0,1,1,0,0,1, 0,1,1,1,1,1,1,0,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 701.60, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (800, 639, 549, 472, 650).
3	b= (600, 500, 500, 500, 600), $\mathbf{u}^*$= (0.282473, 0.379201, 0.128373, 0.113609, 0.096344), $\mathbf{u}^*\mathbf{b}$= 537.88, $\mathbf{x}^{\mathrm{Slc}}$= (0,0,0,0,0,3,0,0,0,0, 0,0,5,5,5,1,5,0,5,4, 0,0,0,0,4,0,1,0,5,5, 2,5,0,0,0,4,0,3,0), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 533.44, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (600, 500, 474, 492, 598)
4	b= (800, 650, 550, 550, 650), $\mathbf{u}^*$= (0.701090, 0.000041, 0.295762, 0.003077, 0.000031), $\mathbf{u}^*\mathbf{b}$= 725.28, $\mathbf{x}^{\mathrm{Slc}}$= (0,0,0,0,0,0,0,0,0,0, 0,0,5,0,4,0,5,0,5,4, 0,0,0,0,0,0,0,0,0,5, 0,5,0,0,0,0,0,0,5,0, 0,5,4,0,0,0,1,5,4,0), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 725.21, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (800, 497, 550, 530, 532)
5	b= (800, 650, 550, 550, 650), $\mathbf{u}^*$= (0.485373, 0.406892, 0.088352, 0.019063, 0.000321), $\mathbf{u}^*\mathbf{b}$= 712.06, $\mathbf{x}^{\mathrm{Slc}}$= (0,0,0,0,0,0,0,0,0,0, 0,0,5,5,3,0,5,0,5,0, 0,0,0,0,0,0,0,0,0,5, 0,5,0,0,0,0,0,0,5,0, 0,5,3,0,5,0,5,5,5,0), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 711.062, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (800, 650, 539, 549, 615)
6	b= (4000, 3000, 2500, 5000), $\mathbf{u}^*$= (0.178335, 0.699742, 0.121727, 0.000196), $\mathbf{u}^*\mathbf{b}$= 3117.86, $\mathbf{x}^{\mathrm{Slc}}$= (2,1,1,3,1,1,1,2,1,2, 2,1,1,2,6,1,10,2,2,1, 1,1,1,1,3,1,1,1,2,4, 1,2,1,1,2,2,1,5,10,1, 1,3,4,1,4,1,1,2,1,2), $\mathbf{u}^*\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= 3116.25, $\mathbf{g}(\mathbf{x}^{\mathrm{Slc}})$= (3996, 3000, 2493, 4766)

$\mathbf{b}$: rigtht-hand-side vector of constraints.

$\mathbf{u}^*$: an optimal surrogate multiplier vector obtained by a surrogate constraints algorithm.

$\mathbf{x}^{\mathrm{Slc}}$: a solution obtained by the SA.

Table 2a and 2b illustrate the performance of the SA for solving two or three constrained nonlinear knapsack test problems in Group 2 with 250, 500, or 1000 variables. Each variable is allowed to take 20 different integer values (items). Five test problems have been solved for each problem size. Tables 2a and 2b show that the PGC value is a good measure of the difficulty of the problems. The SA produces exact optimal solutions for all problem sizes with two constraints. For problems with three constraints the SA succeeds in finding exact optimal solutions for all problems with the exception of one problem. The SA failed to produce a solution guaranteed to be exactly optimal to Problem 30 of the size $m = 3$, $n = 1000$, $k_i = 20$ in Table 2b. However, the solution by the SA is a near-optimal solution of fairly high quality. Problem 30 has a small PGC value of 8.8-16.2%. Another hard problem is Problem 25, which has a small PGC value of 17.5%. The SA succeeded in solving exactly the target problem with $f^{\mathrm{T}} = 1{,}592{,}158$, but failed to solve exactly a target problem with $f^{\mathrm{T}} = 1{,}592{,}157$. However the approximate solution produced by the SA is exactly optimal to Problem 25. The SA has spent most of CPU time guaranteeing the exact optimality of the solution. For example, the result for Problem 25 in Table 2b reports that SA took 16

Table 1b Computational results for Group 1

Problem Number	1	2	3	4	5	6
f^{Real}	10,672.3	16,612.8	14,458.1	24,591.7	44,744.4	79,851.1
$f(\mathbf{x}^{\mathrm{SD}})$	10,659.0	16,599.0	14,454.0	24,582.0	44,625.0	79,798.0
$f(\mathbf{x}^{\mathrm{Near}})$	10,479.0	16,499.0	13,981.0	24,306.0	36,041.0	79,537.0
f^{NS}	–	–	14,370.0	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	10,618.0	16,537.0	14,296.0	24,451.0	41,850.0	79,716.0
CPU time for f^{NS}	–	–	6sec	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	<1sec	<1sec	3sec	<1sec	<1sec	1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	24.6%	18.2%	2.5%-4.6%	6.9%	4.1%	39.3%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	near optimal	exact	exact	exact

f^{Real} : optimal value of a linear relaxation on problem of an optimal surrogate problem [$\mathbf{P}^{\mathbf{S}}(\mathbf{u}^*)$].

$\mathbf{x}^{\mathrm{SD}}$: a solution of surrogate dual problem.

$\mathbf{x}^{\mathrm{Near}}$: a near optimal solution obtained by the smart gready algorithm.

$\mathbf{x}^{\mathrm{Slc}}$: a solution obtained by the SA.

f^{NS} : minimum target value whose target problem can be solved exactly by the slicing algorithm and have been constructed with no feasible solution to the original multi-constrained problem.

seconds to produce a better solution $\mathbf{x}^{\mathrm{Slc}}$ than the near-optimal solution $\mathbf{x}^{\mathrm{Near}}$ and took 111 seconds to guarantee the exact optimality of $\mathbf{x}^{\mathrm{Slc}}$.

Table 3 is the computational results for problems with three constraints in Group 3. Problems 1, 2, ..., 5 of Group 3 are easy to solve except that they have too many exact optimal solutions. This is overcome by a thinning-out technique. It should be noted that all problems have the relationships $f(\mathbf{x}^{\mathrm{SD}}) = f(\mathbf{x}^{\mathrm{Slc}})$. A comparison between tables 2 and 3 shows that the test problems in Group 2 are rather harder to solve than problems of the corresponding size in Group 3 and Group 4.

To investigate the effect of allowing the objective and constraint functions to be non-monotone and to have many constraints, the SA has been tested on problems in Group 4. The problems 1-30 in the Group 4 have eight constraints, which were generated as in Karwan, Rardin and Sarin [12]. Problems 1-15 are nonlinear knapsack problems with 2 items for each variable. Problems 16-30 are nonlinear knapsack problems with 50 items for each variable. The constraint functions $g_{ji}(k)$ have 25% zero elements for every $j = 1, 2, \cdots, m$ and $i = 1, 2, \cdots, n$. Computational results for Problems 1-30 of Group 4 are reported in Tables 4a and 4b. These tables show that Group 4 test problems are quite difficult for the smart greedy algorithm to give good solutions. For most problems, the heuristic algorithm fails to produce feasible solutions. Even if the algorithm succeeds to produce a feasible solution, the obtained solution is very poor. On the other hand, the SA succeeds in finding the exact optimal solutions for all problems in Group 4. The computational results suggest that the SA is still powerful even for problems with many constraints and problems with non-monotone functions. There is no clear relationship between the PCG values and the CPU times for Group 4 problems.

Further computational experiments are found in James and Nakagawa [10]. An enumeration method, in conjunction with the SA, is used to find the optimal solutions to a number of 500-variables, 5-constraint multidimensional knapsack test problems presented in Chu and Beasley [3]. The exact solutions to these problems were previously unknown.

Table 2a Computational results for Group 2 ($m \times n \times k_i$)

(a) $2 \times 250 \times 20$

Problem Number	1	2	3	4	5
f^{Real}	796,256.5	803,346.3	796,605.4	805,474.4	812,961.8
$f(\mathbf{x}^{\mathrm{SD}})$	796,238.0	803,340.0	796,596.0	805,460.0	812,958.0
$f(\mathbf{x}^{\mathrm{Near}})$	796,195.0	803,157.0	796,542.0	805,381.0	812,908.0
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	796,220.0	803,317.0	796,575.0	805,432.0	812,943.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	1sec	<1sec	<1sec	3sec	1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	50.7%	21.5%	31.0%	34.0%	20.4%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(b) $2 \times 500 \times 20$

Problem Number	6	7	8	9	10
f^{Real}	1,590,219.7	1,594,289.4	1,595,970.8	1,606,619.8	1,608,771,3
$f(\mathbf{x}^{\mathrm{SD}})$	1,590,211.0	1,594,282.0	1,595,957.0	1,606,616.0	1,608,762.0
$f(\mathbf{x}^{\mathrm{Near}})$	1,590,187.0	1,594,246.0	1,595,934.0	1,606,598.0	1,608,683.0
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	1,590,205.0	1,594,267.0	1,595,948.0	1,606,607.0	1,608,747.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	2sec	4sec	3sec	3sec	3sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	59.0%	33.0%	60.5%	29.8%	38.2%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(c) $2 \times 1000 \times 20$

Problem Number	11	12	13	14	15
f^{Real}	3,185,590.9	3,202,766.9	3,193,243.7	3,210,199.7	3,208,589.7
$f(\mathbf{x}^{\mathrm{SD}})$	3,185,586.0	3,202,761.0	3,193,240.0	3,210,194.0	3,208,585.0
$f(\mathbf{x}^{\mathrm{Near}})$	3,185,563.0	3,202,732.0	3,193,205.0	3,210,167.0	3,208,504.0
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	3,185,579.0	3,202,755.0	3,193,234.0	3,210,189.0	3,208,580.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	15sec	14sec	13sec	14sec	14sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	41.1%	49.6%	38.1%	53.2%	48.6%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

Table 2b Computational results for Group 2 ($m \times n \times k_i$)

(a) $3 \times 250 \times 20$

Problem Number	16	17	18	19	20
f^{Real}	786,005.6	792,355.3	780,755.0	795,285.5	804,599.8
$f(\mathbf{x}^{\mathrm{SD}})$	785,993.0	792,339.0	780,745.0	795,274.0	804,585.0
$f(\mathbf{x}^{\mathrm{Near}})$	785,827.0	792,130.0	780,594.0	795,229.0	804,508.0
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	785,964.0	792,310.0	780,707.0	795,242.0	804,565.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	3sec	3sec	10sec	21sec	1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	30.3%	36.0%	20.9%	26.4%	42.5%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(b) $3 \times 500 \times 20$

Problem Number	21	22	23	24	25
f^{Real}	1,571,928.4	1,575,461.8	1,576,813.6	1,588,045.7	1,592,189.7
$f(\mathbf{x}^{\mathrm{SD}})$	1,571,920.0	1,575,453.0	1,576,807.0	1,588,040.0	1,592,184.0
$f(\mathbf{x}^{\mathrm{Near}})$	1,571,878.0	1,575,405.0	1,576,731.0	1,587,929.0	1,592,123.0
f^{NS}	–	–	–	–	1,592,158.0
$f(\mathbf{x}^{\mathrm{Slc}})$	1,571,899.0	1,575,441.0	1,576,784.0	1,588,019.0	1,592,157.0
CPU time for f^{NS}	–	–	–	–	111sec
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	145sec	11sec	29sec	27sec	16sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	28.5%	42.4%	22.4%	21.4%	17.5%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(c) $3 \times 1000 \times 20$

Problem Number	26	27	28	29	30
f^{Real}	3,154,705.4	3,169,554.7	3,161,112.3	3,172,623.2	3,176,050.7
$f(\mathbf{x}^{\mathrm{SD}})$	3,154,703.0	3,169,550.0	3,161,109.0	3,172,620.0	3,176,049.0
$f(\mathbf{x}^{\mathrm{Near}})$	3,154,568.0	3,169,429.0	3,161,035.0	3,172,564.0	3,175,910.0
f^{NS}	–	–	–	–	3,176,040.0
$f(\mathbf{x}^{\mathrm{Slc}})$	3,154,694.0	3,169,535.0	3,161,104.0	3,172,612.0	3,176,031.0
CPU time for f^{NS}	–	–	–	–	119
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	28sec	161sec	19sec	179sec	18sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	21.4%	23.9%	39.6%	28.3%	8.8%-16.2%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	near optimal

Table 3 Computational results for Group 3 ($m \times n \times k_i$)

$3 \times 1000 \times 20$

Problem Number	1	2	3	4	5
f^{Real}	98,627.1	98,837.3	98,780.4	98,970.4	99,181.0
$f(\mathbf{x}^{\mathrm{SD}})$	98,627.0	98,837.0	98,780.0	98,970.0	99,180.0
$f(\mathbf{x}^{\mathrm{Near}})$	98,613.0	98,820.0	98,761.0	98,969.0	99,175.0
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	98,627.0	98,837.0	98,780.0	98,970.0	99,180.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	18sec	18sec	30sec	23sec	61sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	100.0%	100.0%	100.0%	100.0%	100.0%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

6. Conclusions

We have presented an exact method for solving the multi-dimensional nonlinear knapsack problem. The method is an attempt to close the duality gaps produced by the surrogate constraints methods. The computational experiments show that this attempt succeeds well. The previously unsolved three-constraints nonlinear knapsack problems with 20^{1000} possible combinations of solutions are solved exactly using this technique. Sometimes existing algorithms cannot find an exact optimal solution, even in cases of problems without a duality gap, because of there being too many exact optimal solutions. This difficulty has been overcome by a thinning-out technique. The computational results also show that the percent gap closure (PGC) proposed by Karwan, Rardin and Sarin [12] may be used as a standard by which the difficulty of the problem [P] is measured. A hard problem for the present method SA is not a problem with many variables and is not a problem with many constraints, but is a problem with a wide surrogate gap.

Further research for solving the multi-dimensional nonlinear knapsack problem should be devoted to the development of an effective method for hard problems with a large duality gap. It is also important to develop heuristic algorithms for solving the hard problems.

Acknowledgement

The author would like to thank referees and Prof. Shuzo Yajima and Prof. Norman D. Cook, Kansai University, and Dr. Ross James, University of Canterbury, for their valuable comments and suggestions.

Table 4a Computational results for Group 4 ($m \times n \times k_i$)

(a) $8 \times 20 \times 2$

Problem Number	1	2	3	4	5
f^{Real}	2,974.0	3,551.0	3,282.0	2,007.1	3,123.0
$f(\mathbf{x}^{\mathrm{SD}})$	2,934.0	3,269.0	3,104.0	1,946.0	3,062.0
$f(\mathbf{x}^{\mathrm{Near}})$	infeasible	infeasible	infeasible	infeasible	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	2,668.0	2,932.0	3,104.0	1,654.0	3,052.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	<1sec	<1sec	<1sec	<1sec	<1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	13.1%	45.6%	100.0%	11.4%	85.9%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(b) $8 \times 100 \times 2$

Problem Number	6	7	8	9	10
f^{Real}	17,097.0	15,394.0	18,098.8	16,306.0	14,869.9
$f(\mathbf{x}^{\mathrm{SD}})$	17,082.0	14,955.0	18,083.0	15,883.0	14,857.0
$f(\mathbf{x}^{\mathrm{Near}})$	infeasible	infeasible	infeasible	infeasible	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	17,080.0	14,935.0	17,999.0	15,873.0	14,836.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	<1sec	<1sec	<1sec	<1sec	<1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	88.2%	95.6%	15.8%	97.7%	38.1%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(c) $8 \times 500 \times 2$

Problem Number	11	12	13	14	15
f^{Real}	83,331.0	81,535.3	81,287.7	83,415.6	79,409.4
$f(\mathbf{x}^{\mathrm{SD}})$	83,064.0	81,534.0	81,285.0	83,411.0	79,406.0
$f(\mathbf{x}^{\mathrm{Near}})$	infeasible	infeasible	infeasible	infeasible	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	83,056.0	81,511.0	81,265.0	83,381.0	79,388.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	2sec	3sec	3sec	9sec	3sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	97.1%	5.3%	11.9%	13.3%	15.9%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

Table 4b Computational results for Group 4 ($m \times n \times k_i$)

(a) $8 \times 20 \times 50$

Problem Number	16	17	18	19	20
f^{Real}	246,824.8	246,096.4	245,424.0	246,520.9	247,807.4
$f(\mathbf{x}^{\mathrm{SD}})$	246,606.0	246,057.0	244,973.0	246,476.0	247,747.0
$f(\mathbf{x}^{\mathrm{Near}})$	infeasible	infeasible	infeasible	infeasible	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	246,543.0	246,057.0	244,898.0	246,291.0	247,577.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	<1sec	<1sec	<1sec	<1sec	<1sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	77.6%	100.0%	85.7%	19.5%	26.2%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(b) $8 \times 100 \times 50$

Problem Number	21	22	23	24	25
f^{Real}	1,234,050.8	1,236,572.8	1,229,004.2	1,233,810.1	1,237,837.4
$f(\mathbf{x}^{\mathrm{SD}})$	1,234,040.0	1,236,563.0	1,228,801.0	1,233,799.0	1,237,807.0
$f(\mathbf{x}^{\mathrm{Near}})$	infeasible	infeasible	infeasible	1,232,763.0	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	1,234,017.0	1,236,520.0	1,228,718.0	1,233,761.0	1,237,676.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	1sec	1sec	4sec	1sec	10sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	32.0%	18.6%	71.0%	22.6%	18.8%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

(c) $8 \times 500 \times 50$

Problem Number	26	27	28	29	30
f^{Real}	6,149,388.0	6,154,322.1	6,145,452.7	6,149,597.9	6,153,333.8
$f(\mathbf{x}^{\mathrm{SD}})$	6,148,683.0	6,154,319.0	6,145,448.0	6,149,595.0	6,153,327.0
$f(\mathbf{x}^{\mathrm{Near}})$	6,134,320.0	infeasible	6,140,273.0	6,141,509.0	infeasible
f^{NS}	–	–	–	–	–
$f(\mathbf{x}^{\mathrm{Slc}})$	6,148,675.0	6,154,310.0	6,145,417.0	6,149,578.0	6,153,286.0
CPU time for f^{NS}	–	–	–	–	–
CPU time for $\mathbf{x}^{\mathrm{Slc}}$	25sec	25sec	26sec	25sec	28sec
PGC value of $\mathbf{x}^{\mathrm{SD}}$	98.9%	25.6%	13.2%	14.6%	14.2%
quality of $\mathbf{x}^{\mathrm{Slc}}$	exact	exact	exact	exact	exact

References

[1] R. D. Armstrong, D. S. Kung, P. Sinha and A. A. Zoltners: A computational study of a multiple-choice knapsack algorithm. *ACM Transactions on Mathematical Software*, **9** (1983) 184-198.

[2] K. M. Bretthauer and B. Shetty: The nonlinear resource allocation problem. *Operations Research*, **43** (1995) 670-683.

[3] P. C. Chu and J. E. Beasley: A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, **6** (1998) 63-86.

[4] M. W. Cooper: A survey of methods for pure nonlinear integer programming. *Management Science*, **27** (1981) 353-361.

[5] M. E. Dyer: Calculating surrogate constraints. *Mathematical Programming*, **19** (1980) 255-278.

[6] M. E. Dyer, N. Kayal and J. Walker: A branch and bound algorithm for solving the multiple-choice knapsack problem. *Journal of Computational and Applied Mathematics*, **11** (1984) 231-249.

[7] F. Glover: Surrogate constraints. *Operations Research*, **16** (1968) 741-749.

[8] Ibaraki, T: Enumerative approaches to combinatorial optimization. *Annals of Operations Research*, **10** (1987) 1-602.

[9] T. Ibaraki and N. Katoh: Resource allocation problems (MIT Press, Cambridge, Mass, 1988).

[10] R. J.W. James and Y. Nakagawa: Enumeration methods for repeatedly solving multidimensional knapsack sub-problems. *submitted for publication in ACM Transactions on Mathematical Software.*

[11] M. H. Karwan and R. L. Radin: Surrogate dual multiplier search procedures in integer programming. *Operations Research*, **32** (1984) 52-69.

[12] M. H. Karwan, R. L. Rardin, and S. Sarin: A new surrogate dual multiplier search procedure. *Naval Research Logistics*, **34** (1987) 431-450.

[13] D. G. Luenberger: Quasi-convex programming. *SIAM Journal Applied Mathematics*, **16** (1968) 1090-1095.

[14] K. Marse and S. D. Roberts: Implementing a portable FORTRAN uniform (0,1) generator. *Simulation*, **41** (1983) 135-139.

[15] R. E. Marsten and T. L. Morin: A hybrid approach to discrete mathematical programming. *Mathematical Programming*, **14** (1977) 21-40.

[16] I. Miyaji, Y. Nakagawa and K. Ohno: Decision support system for the composition of the examination problem. *European Journal of Operational Research*, **80** (1995) 139-138.

[17] T. L. Morin and R. E. Marsten: Branch-and-bound strategies for dynamic programming. *Operations Research*, **24** (1976) 611-627.

[18] Y. Nakagawa and K. Nakashima: A heuristic method for determining optimal reliability allocation. *IEEE Transactions on Reliability*, **R-26** (1977) 156-161.

[19] Y. Nakagawa and Y. Hattori: Reliability optimization with multiple properties and integer variables. *IEEE Transactions on Reliability*, **R-28** (1979) 73-78.

[20] Y. Nakagawa and S. Miyazaki: Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Transactions on Reliability*, **R-30** (1981) 175-180.

[21] Y. Nakagawa, M. Hikita and H. Kamada: Surrogate constraints algorithm for reliability optimization problems with multiple constraints. *IEEE Transactions on Reliability*, **R-33** (1984) 301-305.

[22] Y. Nakagawa: A new method for discrete optimization problems. *Electronics and Communications in Japan*, Part 3, 73 (1990) 99-106. *Translated from Transactions of the Institute of Electronics, Information and Communication Engineers*, **73-A** (1990) 550-556.

[23] Y. Nakagawa: Easy C programming (*Asakura Shoten, Tokyo*, 1996) (in Japanese).

[24] Y. Nakagawa and A. Iwasaki: Modular approach for solving nonlinear knapsack problems. *IEICE Transactions on Fundamentals*, **E82-A** (1999) 1860-1864.

[25] Y. Nakagawa: A reinforced surrogate constraints method for separable nonlinear integer programming. *RIMS* **1068**, Kyoto University(1998) 194-202.

[26] R. M Nauss: The 0-1 knapsack problem with multiple choice constraints. *European Journal of Operational Research*, **2** (1978) 125-131.

[27] H. Ohtagaki, Y. Nakagawa, A. Iwasaki, and H. Narihisa: Smart greedy procedure for solving a nonlinear knapsack class of reliability optimization problems. *Mathematical and Computer Modeling*, **22** (1995) 261-272.

[28] C. C. Petersen: Computational experience with variants of the Balas algorithm applied to the selection of R&D projects. *Management Science*, **13** (1967) 736-750.

[29] J. B. Rosen and S. Suzuki: Construction of nonlinear programming test problems. *Communications of ACM*, **27** (1965) 113.

[30] P. Sinha and A. Zoltners: The multiple-choice knapsack problem. *Operations Research*, **27** (1978) 125-131.

Appendix

Table 5 The Petersen problem

i	c_i	a_{1i}	a_{2i}	a_{3i}	a_{4i}	a_{5i}	i	c_i	a_{1i}	a_{2i}	a_{3i}	a_{4i}	a_{5i}
1	560	40	16	38	8	38	26	103	5	5	4	7	8
2	1125	91	92	39	71	52	27	215	10	10	22	8	6
3	300	10	41	32	30	30	28	81	8	6	4	2	0
4	620	30	16	71	60	42	29	91	2	4	6	8	0
5	2100	160	150	80	200	170	30	26	1	0	1	0	3
6	431	20	23	26	18	9	31	49	0	4	5	2	0
7	68	3	4	5	6	7	32	420	10	12	14	8	10
8	328	12	18	40	30	20	33	316	42	8	8	6	6
9	47	3	6	8	4	0	34	72	6	4	2	7	1
10	122	18	0	12	8	3	35	71	4	3	8	1	3
11	322	9	12	30	31	21	36	49	8	0	0	0	0
12	196	25	8	15	6	4	37	108	0	10	20	0	3
13	41	1	2	0	3	1	38	116	10	0	0	20	5
14	25	1	1	1	0	2	39	90	1	6	0	8	4
15	425	10	0	23	18	14	40	738	40	28	6	14	0
16	4260	280	200	100	60	310	41	1811	86	93	12	20	30
17	416	10	20	0	21	8	42	430	11	9	6	2	12
18	115	8	6	20	4	4	43	3060	120	30	80	40	16
19	82	1	2	3	0	6	44	215	8	22	13	6	18
20	22	1	1	0	2	1	45	58	3	0	6	1	3
21	631	49	70	40	32	18	46	296	32	36	22	14	16
22	132	8	9	6	15	15	47	620	28	45	14	20	22
23	420	21	22	8	31	38	48	418	13	13	0	12	30
24	86	6	4	0	2	10	49	47	2	2	1	0	4
25	42	1	1	6	2	4	50	81	4	2	2	1	0

Yuji Nakagawa
Faculty of Informatics
Kansai University
2-1-1 Ryouzenji-Cho, Takatsuki-City
Osaka 569-1095 Japan
E-mail: nakagawa@res.kutc.kansai-u.ac.jp