数理解析研究所講究録
第 798 巻 1992 年 108-118

A New Approach for Solving Large-Scale Discrete Optimization Problems

Yuji NAKAGAWA[†]
仲川 勇二[†]

[†]Faculty of Engineering,
Okayama University of Science

ABSTRACT

A generalized optimization system with a discrete decision space is defined, and an optimization problem associated with the system is described. A new solution method, which is called Modular Approach (MA), is presented to solve the optimization problem. This method extends the Morin-Marsten hybrid idea to solve troublesome problems for Dynamic Programming. The present method is also an extension of branch-and-bound using breadth-first search. Computational experiences for large-scale test problems of the multiple-choice knapsack show the great power of MA.

1. INTRODUCTION

An optimization problem that has a discrete space to be determined, is called a discrete optimization problem. The typical solution methods for the problem are Dynamic Programming (DP) and Branch-and-Bound (B&B). However, basic ideas of the two methods are quite different. The B&B, which solves an original problem by repeatedly dividing it into smaller problems, is the top-down scheme. On the other hand, DP is the bottom-up scheme, which decomposes stages by turns, enumerates partial solutions, and finally solves the primal problem.

The B&B method consists of the branching and the fathoming. The bounding divides problems and the fathoming judges whether or not the generated subproblems should be added into a candidate list. The branching includes the searching that selects a candidate from the candidate list. The early B&B has only two fathoming tests of feasibility and bounding [Geoffrion, Marsten; 72].

Kohler and Steiglitz [74] introduced the dominance, which is the core of DP, into B&B in a generalized form. Ibaraki [77] discussed the importance of dominance in B&B. On the other hand, Morin and Marsten [76, 77] introduced the bounding test, which is the core of B&B, into DP. They call their method hybrid DP/B&B. The hybrid DP/B&B is practically the same as B&B using the breadth-first-search (FIFO policy). The relationship of DP and B&B was discussed by Ibaraki [87] in detail. Ibaraki [87] is devoted to generalizing B&B and DP.

Nakagawa[90] proposed a new solution method called modular approach (MA) for discrete optimization problems. This paper is a development of the paper [Nakagawa; 90]. MA is a bottom-up scheme as well as DP. First, MA considers an optimization system corresponding to a given discrete optimization problem. Next, MA executes the following items 1) and 2) recursively and solves the primal problem:

1) Apply fathoming tests to the current system and reduce the decision space, and
2) Integrate several modules into one module and reduce the number of modules in the current system.

MA is an extension of hybrid DP/B&B and is easily applicable to problems with a hierarchical structure that DP does not deal with very easily.

A number of multiple-choice knapsack test problems with maximum 250,000 variables, which have no integer-dominated variables, were solved on a work-station (CPU R3000). The computational result shows that a policy [CIM] of choosing modules to be integrated plays an important role. The policy [CIM] is an original one of MA, and does not be used in B&B or DP.

2. $|\Lambda|$– tuple

A cartesian product of a family $\{A_i\}_{i\in\Lambda}$ is written as

$$A_{\times\Lambda} \equiv \underset{i\in\Lambda}{\times} A_i, \tag{1}$$

where

$$\Lambda = \{\lambda_1, \lambda_2, \cdots\} \in \mathbf{N} \ \textit{(the set of natural numbers)}. \tag{2}$$

Let a direct sum of a family of mutually exclusive sets, $\{U_i\}_{i\in\Lambda}$, be

$$U_{+\Lambda} \equiv \underset{i\in\Lambda}{+} U_i. \tag{3}$$

A $|\Lambda|$-tuple, which is an element of cartesian product $A_{\times\Lambda}$, is defined as

$$a_\Lambda \equiv (a_{\lambda_1}, a_{\lambda_2}, \ldots, a_{\lambda_{|\Lambda|}}) \tag{4}$$

For example, when $\Lambda = \{2,5,8\}$, $a_\Lambda = (a_2, a_5, a_8)$.

For two ordered tuples a^1_Λ and a^2_Λ, it holds that
(1) if $a^1_i = a^2_i$ for any $i\in\Lambda$, then $a^1_\Lambda = a^2_\Lambda$,
(2) if $a^1_i \geq a^2_i$ for any $i\in\Lambda$, then $a^1_\Lambda \geq a^2_\Lambda$,
(3) if $a^1_i > a^2_i$ for any $i\in\Lambda$, then $a^1_\Lambda > a^2_\Lambda$.

3. Discrete Optimization System and Problem

Consider a system $[\Sigma]$ that consists of $|I|$ units (initial modules), where $I = \{1, 2, \cdots\} \in \mathbf{N}$. Each module has been designated a number $i \in I$. Each module $i\in I$ has $|K_i|$ alternatives, where $K_i = \{1, 2, \cdots\} \in \mathbf{N}$. The set K_i is a set of module alternatives. Its Cartesian product $\mathbf{X} = K_{\times I}$ is the initial decision space. The solutions $\mathbf{x}\in\mathbf{X}$ are alternatives for the system $[\Sigma]$.

The system $[\Sigma]$ is assumed to be under a constraint space $\mathrm{S}\subset\mathbf{N}^{|I|}$. If a solution $\mathbf{x}\in\mathbf{X}$ is $\mathbf{x}\in\mathrm{S}$, then $\mathbf{x}$ is a feasible solution. We consider a real value function f: $\mathbf{X}\to R$ as a performance index for the system. Thus the discrete optimization system is written as the following four tuple.

$$[\Sigma] = [I, \mathbf{X}, \mathrm{S}, f]. \tag{5}$$

The purpose of this paper is to solve the following discrete optimization problem corresponding to $[\Sigma]$.

$$[\Pi]: \max_{\mathbf{x}\in\mathbf{X}} \left\{ f(\mathbf{x}):\ \mathbf{x}\in\mathrm{S}\subset N^{|I|} \right\} \tag{6}$$

4. Modular System and Modular Problem

A modular system at a level $l \in \{0\}+L$ is defined as

$$[\Sigma^{(l)}] = [(M^{(l)}, U^{(l)}), (\mathbf{A}^{(l)}, \mathbf{t}^{(l)}), \mathrm{S}, f] \tag{7}$$

where

L	a set of system numbers; $L = \{1, 2, \cdots\} \in \mathbf{N}$,
$M^{(l)}$	a set of module numbers composing the system $[\Sigma^{(l)}]$,
$U^{(l)}$	a family of mutually excusive sets of unit numbers; $U^{(l)} = \{U_m\}_{m\in M^{(l)}}$,
U_m	a set of unit numbers composing a module $m \in M^{(l)}$,
$\mathbf{A}^{(l)}$	an alternative space of $[\Sigma^{(l)}]$; $\mathbf{A}^{(l)} = \underset{m\in M^{(l)}}{\times} A^{(l)}_m$,

$A_m^{(l)}$	a set of alternatives for a module $m \in M^{(l)}$ at a system level l,
$\mathbf{t}^{(l)}$	mapping from the decision space $\mathbf{A}^{(l)}$ to a partial decision space $\mathbf{X}^{(l)} \subseteq \mathbf{X}$,

A modular system $[\Sigma^{(0)}]$ is the system $[\Sigma]$ itself. That is, $M^{(0)}= I$, $U_m= \{m\}$ $(m \in I)$, $A_m^{(0)}= K_m$ $(m \in I)$, and $\mathbf{t}^{(0)} : \mathbf{A}^{(0)} \rightarrow \mathbf{X}$.

Let us now discuss a process that generates $[\Sigma^{(l)}]$ from a system $[\Sigma^{(l-1)}]$ at the previous level $(l-1)$. We begin by applying fathoming tests to the system $[\Sigma^{(l-1)}]$. Then suppose that, without loss of optimality, we can reduce a module alternative set $A_m^{(l-1)}$ $(m \in M^{(l-1)})$ into a module alternative set $A_m^{(l)}$. That is,

$$A_m^{(l)} \subseteq A_m^{(l-1)} \quad (m \in M^{(l-1)}). \tag{8}$$

Next, using a policy $[CIM]$ (this policy will be discussed later), MA chooses a subset $C^{(l)}$ $(|C^{(l)}| \geq 2)$ from a module set $M^{(l-1)}$. The $|C^{(l)}|$ modules chosen are integrated into a new module with a module number $l+|I|$. The integration of modules means to consider all combinations of alternatives of modules $m \in C^{(l)}$ as an alternative set $A_{l+|I|}^{(l)}$ for the module $l+|I|$. That is,

$$A_{l+|I|}^{(l)}= \{1, 2, \ldots, |A_{\times C^{(l)}}^{(l)}|\}. \tag{9}$$

Then the decision space corresponding to module alternative set $A_{l+|I|}$ is

$$\mathbf{X}_{l+|I|}^{(l)}= \underset{m \in C^{(l)}}{\times} \mathbf{X}_m^{(l)}. \tag{10}$$

The decision variables for the alternative set $A_{l+|I|}^{(l)}$ and the decision space $\mathbf{X}_{l+|I|}^{(l)}$ are $a_{l+|I|}$ and $\mathbf{x}_{l+|I|}$, respectively. Let a bijection mapping from $A_{l+|I|}^{(l)}$ to $\mathbf{X}_{l+|I|}^{(l)}$ be defined as

$$\mathbf{t}_{l+|I|}: A_{l+|I|}^{(l)} \rightarrow \mathbf{X}_{l+|I|}^{(l)}. \tag{11}$$

A set of units composing a module $l+|I|$ is

$$U_{l+|I|}= \underset{m \in C^{(l)}}{+} U_m. \tag{12}$$

A module set $M^{(l)}$ composing a system $[\Sigma^{(l)}]$ is obtained by removing modules $m \in C^{(l)}$ from $M^{(l-1)}$ and adding the new module $l+|I|$; that is,

$$M^{(l)} = M^{(l-1)} - C^{(l)} + \{l+|I|\}. \tag{13}$$

Let

$$\mathbf{t}^{(l)}= \underset{m \in M^{(l)}}{\times} \mathbf{t}_m\,, \tag{14}$$

then $\mathbf{t}^{(l)}$ is the mapping from $\mathbf{A}^{(l)}= A_{\times M^{(l)}}^{(l)}$ to $\mathbf{X}^{(l)}$. We have the relations

$$|I|=|M^{(0)}|>|M^{(1)}|> \cdots >|M^{(|L|)}|=1. \tag{15}$$

$$\sum_{m \in M^{(l)}} U_m= I \quad (l \in L). \tag{16}$$

Then we have a modular system $[\Sigma^{(l)}]$ at a level l. A modular system $[\Sigma^{(|L|)}]$ is assumed to be the last system having only one module $|L|+|I|$; i.e., $M^{(|L|)}= \{|L|+|I|\}$, $U_{|L|+|I|}= I$.

Each of modular systems $[\Sigma^{(l)}]$ $(l \in L)$ is corresponding to the following modular problem:

$$[\Pi^{(l)}]: \max\left\{f(\mathbf{x}): a_{M^{(l)}} \in \mathbf{A}^{(l)},\ \mathbf{x}=\mathbf{t}^{(l)}(a_{M^{(l)}}),\ \mathbf{x} \in \mathbf{S}\right\}. \tag{17}$$

Obviously, this problem is equivalent to $[\Pi]$. As a special case, if we never use the fathoming; that is, if Eq. (8) is

$$A_m^{(l)}= A_m^{(l-1)} \quad (m \in M^{(l)}) \tag{18}$$

at any level $l \in L$, then an alternative set $A_{|L|+|I|}^{(|L|)}$ obtained from Eq. (9) at the last level $l=|L|$ has the relation

$$|A_{|L|+|I|}^{(|L|)}|= |\mathbf{X}|. \tag{19}$$

That is, we have all possible solutions of the system $[\Sigma]$.

5. Fathoming

This section discusses three types of fathoming. The fathoming is used for reducing a decision space $\mathbf{A}^{(l)}$ of a system $[\Sigma^{(l)}]$ $(l \in L)$. As a result of a fathoming test, if we can judge that the optimal value (the objective function value of an optimal solution) of $[\Sigma^{(l)}]$ does not change even if a subset A_m^1 is removed from an alternative set A_m of a module $m \in M^{(l)}$, then the subset A_m^1 has fathomed and can be removed from further consideration. We say that the set A_m^1 is inactive and the remaining set $A_m - A_m^1$ is active.

Let a problem $[\Pi^{(l)}\text{: } a_m \in A_m^1]$ be the problem $[\Pi^{(l)}]$ added a constraint $a_m \in A_m^1 \subseteq A_m$. Since a problem $[\Pi^{(l)}\text{: } a_m \in A_m^1]$ is a subproblem of $[\Pi^{(l)}]$, it holds that

$$v^{OPT}[\Pi^{(l)}\text{: } a_m \in A_m^1] \le v^{OPT}[\Pi^{(l)}], \tag{20}$$

where $v^{OPT}[\cdot]$ is the optimal value of a problem $[\cdot]$. Three tests of fathoming is considered for the subproblem $[\Pi^{(l)}\text{: } a_m \in A_m^1]$.

1) Feasibility test: If the result of this test shows that the subproblem has no feasible solutions, then the alternative set A_m^1 is inactive.
2) Dominance test: If there exists a set $A_m^2 \subseteq A_m$ such that

$$v^{OPT}[\Pi^{(l)}\text{: } a_m \in A_m^1] \le v^{OPT}[\Pi^{(l)}\text{: } a_m \in A_m^2] , \tag{21}$$

then subproblem $[\Pi^{(l)}\text{: } a_m \in A_m^1]$ is said to be dominated. The alternative set A_m^1 is judged to be inactive.
3) Bounding test: Suppose that a nonexact method has already found a feasible near-optimal solution $\mathbf{x}^{NEAR}$ and its objective function value f^{NEAR}. When we get an upper-bound of the subproblem $[\Pi^{(l)}\text{: } a_m \in A_m^1]$ by some technique (in most cases, relaxation techniques are used), if

$$v^{UB}[\Pi^{(l)}\text{: } a_m \in A_m^1] \le f^{NEAR}, \tag{22}$$

where $v^{UB}[\cdot]$ is upper-bound of a problem $[\cdot]$, the subproblem has no feasible solutions better than $\mathbf{x}^{NEAR}$. Hence the alternative set A_m^1 is inactive.

After the fathoming test, we can remove an inactive alternative set A_m^1 from an alternative set A_m of a module $m \in M^{(l)}$ of a system $[\Sigma^{(l)}]$. Then alternative space $\mathbf{A}^{(l)}$ of system $[\Sigma^{(l)}]$ is reduced.

The fathoming mentioned above is only for the case of finding only one optimal solution. When we want to find all of optimal solutions, at the dominance test, replace Eq. (21) with

$$v^{OPT}[\Pi^{(l)}\text{: } a_m \in A_m^1] < v^{OPT}[\Pi^{(l)}\text{: } a_m \in A_m^2] \tag{23}$$

and at the bounding test replace Eq. (22) with

$$v^{UB}[\Pi^{(l)}\text{: } a_m \in A_m^1] < f^{NEAR}. \tag{24}$$

6. Procedure for Modular Approach

MA uses a policy

$$[Py] = [\ [Fm], [CIM]\ ], \tag{25}$$

where

$[Fm]$	policy for determining what kinds of fathoming are applied for which modules.
$[CIM]$	policy for choosing modules to be integrated.

We can subdivide the policy [Fm] as follows:

$$[Fm] = [\ [CFM], [NEAR], [FT]\ ] \tag{26}$$

where

$[CFM]$	policy for determining which modules will be put under the fathoming tests.
$[NEAR]$	near-optimal solution $\mathbf{x}^{NEAR}$ and its objective function value (near-optimal value) f^{NEAR}.
$[FT]$	policy that is how to use what kind of fathoming tests.

Suppose that we have prepared two functions Fathom() and ChoiceIM() to execute the policies $[Fm]$ and $[CIM]$, respectively. Fig. 1 shows a procedure of MA by a pseudo code considering abstraction of data structure. An arrow $\Leftarrow$ means multiple outputs of function. For example, $\{ A, B \} \Leftarrow func(C, D)$ means that the function func() has two input data C and D, and returns two data A and B. The functions in Fig. 1 are as follows:

Fathom() executes fathoming tests to System $[\Sigma^{(l-1)}]$ by using policy $[Fm]$. Then, if possible, find a better near-optimal solution. The function returns a reduced decision space $A^{(l)}_{\times M^{(l-1)}}$, and a near-optimal solution.

ChoiceIM() chooses modules to be integrated from a module set $M^{(l)}$ by using a policy $[CIM]$. The return of this function is a set $C^{(l)}$ of selected modules.

Integrate() integrates the modules $m \in C^{(l)}$ into one module $l+|I|$ so as to obtain a new modular system $[\Sigma^{(l)}]$. This function returns the new system.

FindOptimalSolution() finds an optimal solution of a system $[\Sigma^{(|L|)}]$ having only one module. The return is an optimal solution $\mathbf{x}^{OPT}$.

7. Policies of MA

When MA is applied to practical problems, the performance of MA greatly depends on the policy $[Py]$. Here we discuss some examples of the policy $[Py] = [[Fm], [CIM]]$.

Policy $[Fm]$ can be divided into three policies $[CFM]$, $[NEAR]$, and $[FT]$. As for Policy $[CFM]$, it should be adopted to apply fathoming tests to all modules at the first level $l = 1$. At $l > 1$, the following policies are considered: Choose a module generated most recently by the integration or having the greatest number of alternatives. Furthermore, we can consider the policy that all modules or the modules having alternatives more than a certain number are fathomed at every certain number of levels. The policy [CFM] used by B&B and DP
is the rigid one that applies fathoming tests only to the most recently generated module.

If there exists some efficient bounding technique which quickly calculates good upper bounds, a good near-optimal value f^{NEAR} is very useful in reducing alternative spaces. However, if it does not exist, then a good f^{NEAR} is of no practical use. A good f^{NEAR} and a good bounding work as one body. The general bounding technique uses the relaxation of integer restriction of problems. However, some problems are solved effectively by using approximate equations [Nakagawa et al. 78].

The number of modules in a system can be reduced by integration. Generally the proper policy $[CIM]$ for choosing integrated modules is to choose two modules having the fewest number $|A_m|$ $(m \in M^{(l)})$ of alternatives [Nakagawa et al. 78]. Since it is desirable that $\sum_{m \in M^{(l)}} |A_m|$ at every level is as small as possible. However, this is not always true. In section 9, we will see an evidance against this.

Now let us consider relations between MA, B&B, and DP. We classify B&B into

1) earlyB&B: The early B&B which does not use the dominance in the fathoming,
2) B&B: B&B using the dominance.

DP is classified as follows:

1) pureDP: DP that does not use the bounding,
2) hybridDP/B&B: DP using the bounding.

Then Fig. 2 illustrates the relationship with MA.

MA includes hybridDP/B&B as its special case. That is, if Policy $[CFM]$ is a policy to choose a module that is generated most recently (i.e., has the largest module number)and Policy $[CIM]$ is to choose two modules having the largest module number, then MA is essentially the same as hybridDP/B&B.

8. Example

In this section, MA is illustrated by a simple example. Consider a series-parallel reliability system having five units as shown Fig. 3, and each unit has four alternatives. Table 1 shows the reliability and cost of alternatives. The allowable maximum cost is $b = 60$.

A discrete optimization system can be written as

$$[\Sigma] = [\, I, \mathbf{X}, \mathbf{S}, f \,], \tag{27}$$

where

$$I = \{ 1, 2, 3, 4, 5 \}$$

$$\mathbf{X} = \underset{i \in I}{\times} K_i, \; K_i = \{1,2,3,4\} \;\; (i \in I)$$

$$\mathbf{S} = \left\{ \mathbf{x} \in \mathbf{X}: g(\mathbf{x}) = \sum_{i \in I} g_i(x_i) \leq b \right\}$$

$$f(\mathbf{x}) = \{1-(1-r_1(x_1)r_2(x_2))(1-r_3(x_3))\}r_4(x_4)r_5(x_5)$$

The problem to be solved is to find a combination of alternatives that achieves the maximum system reliability within the allowable cost b; i.e., to find an exact solution of optimization problem $[\Pi]$ corresponding to system $[\Sigma]$.

Table 2 shows the input $[\Sigma^{(0)}]$ for the function ModularApproach(). Policy $[Py]$ is

$[CFM]$ = to apply the fathoming tests to all modules at every level,

$[NEAR] = [f^{NEAR}, \mathbf{x}^{NEAR}] = [0.9349, (1,3,1,3,4)]$,

$[FT]$ = (this policy will be explained later by an example),

$[CIM]$ = to integrate modules having the fewest alternatives by turns by taking the function form of $f(\mathbf{x})$ into consideration.

At level l = 1, the function Fathom() reduces the alternative space by using the following fathoming tests:

1) Dominance: An alternative $a_1=2$ $(g_1(2)=5.8, r_1(2)=0.91)$ is dominated by an alternative $a_1=1$ $(g_1(1)=5.7, r_1(1)=0.95)$. The alternative $a_1=2$ is inactive.
2) Feasibility: Even if all modules besides a module under the fathoming test use alternatives of the minimum cost, we have

$$g(1,1,1,4,1) = 73 > b=60.$$

The alternative $a_4=4$ is judged to be inactive.

3) Bounding: Even if all modules besides a module under the fathoming test use alternatives of the maximum reliability,

$$f(4,4,4,1,4) = 0.6923 < f^{NEAR} = 0.9349$$
$$f(4,4,4,2,4) = 0.8407 < f^{NEAR}$$
$$f(4,4,4,3,1) = 0.8536 < f^{NEAR}$$
$$f(4,4,4,3,2) = 0.9303 < f^{NEAR}.$$

The alternatives x_3= 1, 2 and x_4= 1, 2 are inactive.

First ChoiceIM() chooses modules 4 and 5 as modules to be integrated by taking the form of reliability function $f(\mathbf{x})$ into consideration. Function Integrate() integrates modules 4 and 5 into module 6 and returns a modular system $[\Sigma^{(1)}]$. These results is shown in Table 3(a).

After level l=2, the procedures mentioned above are repeated and lastly an optimal solution $a_9=2$ $(\mathbf{x}^{OPT}=(1,2,2,3,4)$, $f^{OPT}=0.9390)$ is yielded. The progress are shown in Table 3(b), (c), (d). The integrating process of modules is illustrated in Fig. 4.

9. Computational Results

The purpose of this computational experience is to show the power of policy CIM, which is a distinctive characteristic of MA. The MA algorithm for the multiple choice knapsack problem was implemented through a code written in C language and tested on a UNIX workstation (SONY NEWS-3260; CPU R3000). The computer code treats all floating-point calculations in double precision.

The multiple-choice knapsack problem with $|I|$ classes and $|K_i|$ 0-1 variables for each class is written

$$[MCK]: \;\; \max \sum_{i \in I} \sum_{k \in K_i} c_{ik} \xi_{ik}$$

$$s.t. \sum_{i\in I}\sum_{k\in K_i} a_{ik}\xi_{ik} \le b$$

$$\sum_{k=1}^{K_i} \xi_{ik} = 1 \quad (i \in I)$$

$$\xi_{ik} \ge 0, \; integer \quad (k \in K_i,\ i \in I).$$

where $I=\{1, 2, \cdots\} \in \mathbf{N}$, $K_i = \{1, 2, \cdots\} \in \mathbf{N}$.

The MA algorithm treats a multiple-choice knapsack problem as an equivalent nonlinear knapsack problem. The nonlinear knapsack problem with $|I|$ variables and $|K_i|$ alternatives for each variables:

$$[NK]: \quad \max \sum_{i\in I} f_i(x_i)$$

$$s.t. \sum_{i\in I} g_i(x_i) \le b$$

$$x_i \in K_i \quad (i \in I).$$

where $f_i(k) = c_{ik}$, $g_i(k) = a_{ik}$ $(k \in K_i,\ i \in I)$. The problem [NK] has much smaller solution space than [MCK], since the solution space of [MCK] and [NK] are $2^{\sum_{i\in I}|K_i|}$ and $\prod_{i\in I} |K_i|$ $(= 2^{\sum_{i\in I}\log_2 |K_i|})$, respectively.

The discrete optimization system corresponding to [NK] can be written as

$$[\Sigma] = [\, I, \mathbf{X}, \mathbf{S}, f\,], \tag{$*$}$$

where

$$\mathbf{X} = \prod_{i\in I} K_i,$$

$$\mathbf{S} = \left\{ \mathbf{x} \in \mathbf{X}: g(\mathbf{x}) = \sum_{i\in I} g_i(x_i) \le b \right\}$$

$$f(\mathbf{x}) = \sum_{i\in I} f_i(x_i).$$

Two types of Multiple-Choice Knapsack test problems were randomly generated by an uniform random number generater [Marse and Roberts 83]. Type 1 problems has integer coefficients a_{ik}, c_{ik} $(i \in I,\ k \in K_i)$ and

$$1 \le c_{i,k+1} - c_{ik} \le 8$$

$$1 \le a_{i,k+1} - a_{ik} \le 8$$

for any k, $k+1 \in K_i$ $(i \in I)$, and

$$b = \sum_{i\in I} \frac{(a_{i1} + a_{i|K_i|})}{2}$$

This test problems are the most difficult problems in the paper [Sinha and Zoltners 79]. Type 2 test problems has real (double precision) coefficients

$$0.0 \le c_{i,k+1} - c_{ik} \le 1.0$$

$$0.0 < a_{i,k+1} - a_{ik} \le 1.0$$

for any k, $k+1 \in K_i$ $(i \in I)$, and

$$b = \sum_{i\in I} \frac{(a_{i1} + a_{i|K_i|})}{2}$$

The coefficients of test problems of both of types 1 and 2 are treated as double precision real numbers in the present computer code.

The computer code of MA used the following policies:

[*CFM*] At the beginning and when a near-optimal (the current best) solution is renewed, all modules are put under fathoming tests. Otherwise, a generated new module is fathomed.

[*NEAR*] Initial near-optimal solutions are generated by a modification of the N-N method [Nakagawa, Nakashima 77]. A near-optimal solution is renewed by a better solution found during bounding tests.

[*FT*] Upper bounds of subproblems are obtained by using the Sinha-Zoltners bound [Sinha, Zoltners 79] (This bound should be investigated in detail, since the bound hardly seems to be much better than the LP bound)

[*CIM*] A: Choose two modules having the fewest alternatives,
B: Choose a module having the fewest alterntives and a module having the most alternatives,
C: Choose two modules having the most alternatives,
D: Choose two modules having the maximum mudule number
(Note that a module generated the most recently by an integration has the maximum module number).

The computational code with the policy D as CIM is essentially the same as the hybrid DP/B&B or the branch-and-bound using the breadth-first-search.

Table 4 shows the computational results for type 1 test problems, whose coefficients are all integer. The times reported are in seconds and does not include the times for obtaining initial near-optimal solutions. When we use the policy C as CIM, the code can not solve some of 25 test problems whose size is $|I|= 100$ and $|K_i|= 50$. The code with th policy D solved all of 25 test problems whose size is $|I|= 2000$ and $|K_i|= 50$. However, the code can not solve some of 3000×50 test problems. The code with the policy A or B are clearly superior to the one with policy C or D. Between A and B, there is no big difference in computational times. The code with A or B can solve all of 25 randomly generated test problems whose size is $|I|= 5000$ and $|K_i|= 50$. The memory space needed for keeping the largest size of intermidiate modular problems depends on the maximum number of alternatives per a modular problem. As for the memory space, policy B is clearly superior to policy A. The computation times by policy B are scattered. For example, in the case of $|I|= 5000$ and $|K_i|= 50$, times for 20 out of 25 problems are less than 1.0 seconds, where the additional times of 68.2 ± 1.6 seconds are used for getting initial near-optimal solutions for MA (see Table 5).

Table 6 shows the computational result for type 2 test problems, whose coefficients are all real numbers. The comparison of Tables 4 and 6 gives that problems with real coefficients is much more difficult than ones with integer coefficients. Table 6 shows the big power of the policy CIM. The policy B is the best one for CIM and is very stable in both of the computation time and the required memory space.

10. Concluding Remarks

This paper presents Modular Approach in a generalized form as a new solution method for discrete optimization problems. The present method extends the hybridDP/B&B and is easily applicable to the problems having hierarchical structure that are difficult for Dynamic Programming and Branch-and-Bound to apply without any special idea. Since the present approach is flexible, it will develop more efficient solution algorithms even for the problems that can be solved by Dynamic Programming or Branch-and-Bound. Computational experiments of the existing papers are done for problems with only integer coefficients and for less than 20 test problems per one size. Our computational experience suggests that more than 20 test problems per one size should be solved since different problems of the same size has the quite different computational times. Test problems with real coefficients should be solved, since problems with integer coefficients are much easier than the ones with real coefficients.

Acknowledgments

The author is grateful to Dr. Ronald L. Rardin, Professor of Purdue University, for his valuable advices and discussions.

References

Armstrong R. D., D. S. Kung, P. Sinha and A. A. Zoltners : "A computational Study of a multiple-choice knapsack algorithm", ACM Trans. on Math. Software, Vol. 9, pp. 184-198 (1983).

Dyer M. E., N. Kayal and J. Walker : "A branch and bound algorithm for solving the multiple-choice knapsack problem", J. Computational and Applied Math., Vol. 11, pp. 231-249 (1984).

Geoffrion A. M. and R. E. Marsten : "Integer programming algorithms: A framework and state of the art survey", Manag. Sci., Vol. 18, No. 9, pp. 465-491 (May 1972).

Ibaraki T.: "The power of dominance relations in branch-and-bound algorithms", J. Assoc. Comput. Mach., Vol. 24, No. 2, pp. 264-279 (Apr. 1977).

Ibaraki T.: "Enumerative approachs to combinatorial optimization", Annals of Operations Research,Vol. 10, No. 1-4, pp. (1987).

Kohler W.H. and K. Steiglitz : "Characterization and theoretical comparison of branch-and-bound algorithm for permutation problems", J. Assoc. Comput. Mach., Vol. 21, No. 1, pp. 140-156 (Jan. 1974).

Marse K. and S. D. Roberts : "Implementing a portable FORTRAN Uniform (0,1) generator", SIMULATION, Vol. 41, pp. 135-139 (Oct. 1983). pp 135-139 (1983).

Marsten R. E. and T. L. Morin : "A hybrid approach to discrete mathematical programming", Math. program., Vol. 14, pp. 21-40 (1977).

Morin T. L. and R. E. Marsten : "Branch-and-bound strategies for dynamic programming", Oper. Res., Vol. 24, No. 4, pp. 611-627 (July-Aug. 1976).

Nakagawa Y. and K. Nakashima : "A heuristic method for determining optimal reliability allocation", IEEE Trans. Reliab., Vol. R-26, No. 3, pp. 156-161 (Aug. 1977).

Nakagawa Y., K. Nakashima, and Y. Hattori : "Optimal reliability allocation by branch-and-bound technique", IEEE Trans. Reliab., Vol. R-27, No. 1, pp. 31-38 (Apr. 1978).

Nakagawa Y. : "A new method for discrete optimization problems", Electronics and Comunications in Japan, Part 3, Vol. 73, No. 11, pp. 99-106 (1990), translated from Trans. of the Institute of Electronics, Information and Communication Engineers, Vol. 73-A, No. 3, pp. 550-556 (March 1990) (In Japanese).

Parker R. G. and R. L. Rardin : "Discrete Optimization", Academic Press (1988).

Sinha P. and A. Zoltners : "The multiple-choice knapsack Problem", Operations Res., Vol. 27, No. 3, pp. 125-131 (1978).

```
FUNCTION ModularApproach()
INPUT  System $[\Sigma^{(0)}]$, Policy $[Py] = [\,[Fm], [CIM]]$;
BEGIN
        $l \leftarrow 0$;
        WHILE $|M^{(l)}| > 1$ DO
                $l \leftarrow l + 1$;
                $\{ A^{(l)}_{\times M^{(l-1)}}, [NEAR] \} \Leftarrow$ Fathom( $l$, $[\Sigma^{(l-1)}]$, $[Fm]$);
                $C^{(l)} \Leftarrow$ ChoiceIM( $M^{(l-1)}$, $A^{(l)}_{\times M^{(l-1)}}$, $[CIM]$);
                $[\Sigma^{(l)}] \Leftarrow$ Integrate( $C^{(l)}$, $[\Sigma^{(l-1)}]$, $A^{(l)}_{\times M^{(l-1)}}$);
        ENDWHILE

        IF $|A^{(l)}_{l+|l|}| = 0$ THEN
                $x^{OPT} \leftarrow x^{NEAR}$; $f^{OPT} \leftarrow f^{NEAR}$;
        ELSE
                $x^{OPT} \Leftarrow$ FindOptimalSolution($[\Sigma^{(l)}]$);
                $f^{OPT} \leftarrow f(x^{OPT})$;
        ENDIF

RETURN Optimal Solution $x^{OPT}$, Optimal value $f^{OPT}$;
END
```

Fig. 1 A Procedure of Modular Approach.

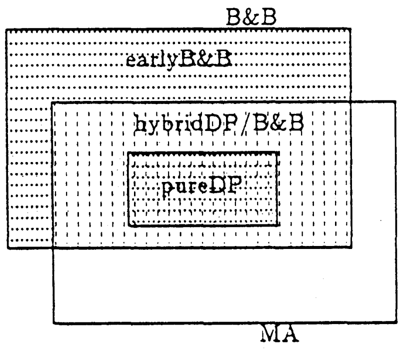

Fig.2 Relations between three methods (MA, B&B, and DP).

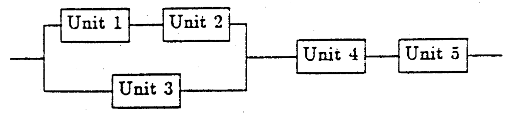

Fig.3 A series-parallel reliability system.

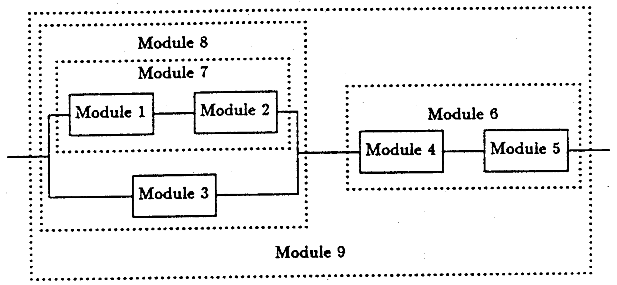

Fig.4 An integrating process of modules.

Table 1 Reliability and cost of alternatives

Alternative	Reliability					Cost				
k	$r_1(k)$	$r_2(k)$	$r_3(k)$	$r_4(k)$	$r_5(k)$	$g_1(k)$	$g_2(k)$	$g_3(k)$	$g_4(k)$	$g_5(k)$
1	0.95	0.76	0.86	0.70	0.89	5.7	3.5	8.5	28.1	5.3
2	0.91	0.80	0.95	0.85	0.97	5.8	3.6	10.3	30.0	7.5
3	0.96	0.93	0.97	0.96	0.98	9.6	4.5	14.4	32.0	7.9
4	0.98	0.97	0.98	0.99	0.99	10.5	7.3	16.6	50.0	8.3

Table 2 Data of System $\Sigma^{(0)}$

$M^{(0)}$			$\{1, 2, 3, 4, 5\}$
$U^{(0)}$			$\left\{\{1\},\{2\},\{3\},\{4\},\{5\}\right\}$
$t^{(0)}$	t_1	$A_1^{(0)}$	$\{1, 2, 3, 4\}$
		$X_1^{(0)}$	$\{(1),(2),(3),(4)\}$
	t_2	$A_2^{(0)}$	$\{1, 2, 3, 4\}$
		$X_2^{(0)}$	$\{(1),(2),(3),(4)\}$
	t_3	$A_3^{(0)}$	$\{1, 2, 3, 4\}$
		$X_3^{(0)}$	$\{(1),(2),(3),(4)\}$
	t_4	$A_4^{(0)}$	$\{1, 2, 3, 4\}$
		$X_4^{(0)}$	$\{(1),(2),(3),(4)\}$
	t_5	$A_5^{(0)}$	$\{1, 2, 3, 4\}$
		$X_5^{(0)}$	$\{(1),(2),(3),(4)\}$

Table 3(a) State of variables at level 1.

$l = 1$		
$A_1^{(1)}$		$\{1, 3, 4\}$
$A_2^{(1)}$		$\{1, 2, 3, 4\}$
$A_3^{(1)}$		$\{1, 2, 3, 4\}$
$A_4^{(1)}$		$\{3\}$
$A_5^{(1)}$		$\{3, 4\}$
$C^{(1)}$		$\{4, 5\}$
$M^{(1)}$		$\{1, 2, 3, 6\}$
$U^{(1)}$		$\left\{\{1\},\{2\},\{3\},\{4,5\}\right\}$
t_6	$A_6^{(1)}$	$\{1, 2\}$
	$X_6^{(1)}$	$\{(3,3),(3,4)\}$

Table 3(b) State of variables at level 2.

$l = 2$		
$A_1^{(2)}$		$\{1\}$
$A_2^{(2)}$		$\{1, 2, 3\}$
$A_3^{(2)}$		$\{1, 2\}$
$A_6^{(2)}$		$\{1, 2\}$
$C^{(2)}$		$\{1, 2\}$
$M^{(2)}$		$\{3, 6, 7\}$
$U^{(2)}$		$\left\{\{3\},\{4,5\},\{1,2\}\right\}$
t_7	$A_7^{(2)}$	$\{1, 2, 3\}$
	$X_7^{(2)}$	$\{(1,1),(1,2),(1,3)\}$

Table 3(c) State of variables at level 3.

$l = 3$		
$A_3^{(3)}$		$\{1, 2\}$
$A_6^{(3)}$		$\{1, 2\}$
$A_7^{(3)}$		$\{1, 2, 3\}$
$C^{(3)}$		$\{3, 7\}$
$M^{(3)}$		$\{6, 8\}$
$U^{(3)}$		$\left\{\{4,5\},\{1,2,3\}\right\}$
t_8	$A_8^{(3)}$	$\{1, 2, 3, 4, 5, 6\}$
	$X_8^{(3)}$	$\{(1,1,1),(1,1,2),(1,2,1),(1,2,2),(1,3,1),(1,3,2)\}$

Table 3(d) State of variables at level 4.

$l = 4$		
$A_6^{(4)}$		$\{2\}$
$A_8^{(4)}$		$\{2, 4\}$
$C^{(4)}$		$\{6, 8\}$
$M^{(4)}$		$\{9\}$
$U^{(4)}$		$\left\{\{1,2,3,4,5\}\right\}$
t_9	$A_9^{(4)}$	$\{1, 2\}$
	$X_9^{(4)}$	$\{(1,1,2,3,4),(1,2,2,3,4)\}$
$x^{OPT} = (1,2,2,3,4)$		

Table 4. Computational Results for Type 1 Problems with Integer Coefficients

$\|I\|\times\|K_i\|$		Policies			
		A	B	C	D
30×50	average time[a]	0.4	0.3	219.0	3.0
	maximum time[b]	1.1	0.6	2600.1	51.9
	max num. of alt.[c]	1767	1168	103569	14756
100×50	average time	2.5	2.3		41.9
	maximum time	9.6	4.4	---[d]	290.8
	max num. of alt.	5378	4858		15019
500×50	average time	47.1	48.4		463.8
	maximum time	143.9	121.2	---	6537.5
	max num. of alt.	23514	23563		32816
1000×50	average time	186.9	166.5		1004.0
	maximum time	649.8	558.5	---	7244.9
	max num. of alt.	51272	42973		53790
2000×50	average time	840.7	792.4		12774.2
	maximum time	3284.7	2412.4	---	219526.0
	max num. of alt.	151553	3292		109926
3000×50	average time	1639.6	1620.3		
	maximum time	7024.9	6633.0	---	---
	max num. of alt.	85410	3930		
4000×50	average time	4508.3	4220.6		
	maximum time	17141.0	15424.5	---	---
	max num. of alt.	163417	5312		
5000×50	average time	2536.6	2974.9		
	maximum time	26722.0	30562.5	---	---
	max num. of alt.	83262	6473		

$|I|$: number of classes.
$|K_i|$: number of variables per class.
[a,b]average (or maximum) time of 25 test problems in seconds.
[c]maximum number of alternatives per one modular problem; i.e., maximum of $\max_{l \in L}\left\{\sum_{m \in M^{(l)}} |A_m|\right\}$ for 25 test problems. (This number indicates the minimum storage space necessary to solve the 25 test problems.)
[d]not available due to the storage limitation or the computational time.

Table 5 Computational Times for Near-Optimal solutions of Type 1 Problems

$\|I\|\times\|K_i\|$		1000×50	2000×50	3000×50	4000×50	5000×50
Bound[a]	average	0.9	2.6	5.1	8.5	12.6
	maximum	0.9	2.6	5.2	8.6	12.8
Near[b]	average	5.7	15.2	28.8	46.5	68.3
	maximum	6.0	15.8	29.6	47.8	69.9
Exact (policy B)	average	166.5	792.4	1620.3	4220.6	2974.9
	maximum	558.5	2412.4	6633.0	15424.5	30562.5

[a]Times for obtaining upper bounds of test problems by the Sinha-Zoltners bound [Sinha, Zoltners 79].
[b]Times for obtaining near-optimal solutions of test problems by a modification of N-N method [Nakagawa, Nakashima 77].

Table 6. Computational Results of Test Problems with Real Coefficient

$\|I\|\times\|K_i\|$[a,b]		Policies			
		A	B	C	D
30×50	average time	1.6	0.8		33.0
	maximum time	14.4	4.4	---	428.6
	max num. of alt.	16630	2186		29021
100×50	average time	19.5	15.4		3401.5
	maximum time	105.6	71.5	---	28024.4
	max num. of alt.	63758	7067		195055
500×50	average time		352.7		
	maximum time	---	2410.0	---	---
	max num. of alt.		36010		
1000×50	average time		1424.1		
	maximum time	---	6101.6	---	---
	max num. of alt.		59423		