Journal of the Operations Research Society of Japan
Vol. 41, No. 3, September 1998

AN INTEGER RESOURCE ALLOCATION PROBLEM WITH COST CONSTRAINT

Ryusuke Hohzaki Koji Iida
National Defense Academy

(Received August 1, 1997; Final April 20, 1998)

Abstract This paper investigates a non-linear integer programming problem with an exponential objective function and cost constraints, which is a general model and could be applied to the assignment problem or the search problem. As a representative example, we consider a following problem of assigning m types of missiles to n targets. The value of target i is V_i and the single shot kill probability (SSKP) of a j-type missile to target i is $\beta_{ij} = 1 - \exp(-\alpha_{ij})$ where $\alpha_{ij} \geq 0$. Denoting the number of j-type missiles assigned to target i by n_{ij}, the expected destroyed value is given by $\sum_{i=1}^{n} V_i \left\{1 - \exp(-\sum_j \alpha_{ij} n_{ij})\right\}$. If a unit price of j-type missile is c_j and the total cost of missiles is limited by C, we have cost constraints $\sum_{j=1}^{m} c_j \sum_{i=1}^{n} n_{ij} \leq C$. To this problem which is NP-hard, we propose two methods, the dynamic programming method and the branch and bound method. An approximate algorithm and an estimation of the upper bound of the objective function are incorporated in the branch and bound method. Each of these methods has its characteristic merits for the computational time. We clarify their characteristics through the sensitivity analysis by some examples in this paper.

1. Introduction

This paper investigates a non-linear integer programming problem with an exponential objective function and cost constraints, which is frequently encountered in a variety of application areas, and proposes new methods for solving the problem. For example, we consider a following problem of assigning m types of missiles to n targets. The value of target i is V_i and the single shot kill probability (SSKP) of a j-type missile to target i is $\beta_{ij} = 1 - \exp(-\alpha_{ij})$ where $\alpha_{ij} \geq 0$. Denoting the number of j-type missile assigned to target i by n_{ij}, the expected destroyed value is given by $\sum_{i=1}^{n} V_i \left\{1 - \exp(-\sum_j \alpha_{ij} n_{ij})\right\}$. If a unit price of j-type missile is c_j and the total cost of missiles is limited by C, we have cost constraints $\sum_{j=1}^{m} c_j \sum_{i=1}^{n} n_{ij} \leq C$. R.H. Nickel et al.[5] incorporated a similar model in his uniformly assignment model with multi-types of missiles and multi targets. Since they dealt with their constraints as the condition not on cost but on the numbers of missiles, their optimization problem was slightly different from our problem and comparatively easily solved.

We can find a similar model in the search problems. A searcher wants to maximize the probability of detecting a stationary target which hides himself in one of n cells. The location of the target is not known with a certainty and the prior probability of the target's existing in cell i is estimated by p_i. The searcher is given T search time points, $t = 1, 2, \cdots, T$, when he can look into whichever cell as many times as he likes. At time point t, he can detect the target with probability $\beta_{it} (= 1 - \exp(-\alpha_{it}))$ by looking into cell i if the target is there. On the other hand, a look into a cell costs c_t at time t and the total cost is limited by C. When the searcher has the strategy of looking into cell i n_{it} times at time t,

the detection probability and the cost constraint are given as $\sum_{i=1}^{n} p_i \{1 - \exp(-\sum_t \alpha_{it} n_{it})\}$ and $\sum_{t=1}^{T} c_t \sum_{i=1}^{n} n_{it} \leq C$, respectively. A problem of distributing discrete search resources was studied by J.B. Kadane[6]. He proposed a little different model where the searcher took only a look into a cell at every time and the look cost varied depending on the number of past looks, and the detection probability was to be maximized. His problem is different from our problem and it is able to be reduced to the problem of optimizing the number of looks at one time point and to the knapsack problem in consequence.

The above two models, the missile assignment problem and the search problem have the same objective function and the same constraints on variables. A purpose of this paper is not to solve the real-life missile assignment problem or the practical search problem but to propose some new methods of giving an exact solution to this type of problem with an exponential objective function and cost constraints, which has not been studied clearly so far.

Our problem stated above becomes the knapsack problem in the case of $n = 1$. It reminds us that our problem is NP-hard. In this paper, the dynamic programming method and the branch and bound method are proposed to solve our problem. In the next section, we formulate our problem as an integer programming problem. Two exact methods, the dynamic programming method and the branch and bound method are discussed in Sections 3 and 4, respectively. In Section 5, a numerical example is examined to investigate the alternative strategy about which of special use missiles or general-purpose missiles should be adopted. Other numerical examples are used to estimate the performance of two methods concerning with the computational time.

2. Formulation of Problem

Here, noting that the same model is possible to some problems in other fields as stated in the introduction, we take a missile assignment model as one of representative examples and formulate it as an integer programming problem.

(1) A defender has n targets with their values $\{V_i \geq 0\ ,\ i = 1, \cdots, n\}$.

(2) An attacker has m types of missiles and a j-type missile has the capability of SSKP $0 \leq \beta_{ij} < 1$ to target i.

(3) A unit cost of j-type missile is $c_j > 0$ and the total cost of the attacker is limited by $C > 0$.

(4) The attacker wants to find an optimal assignment of his missiles of maximizing the expected destroyed value of targets.

For convenience, we replace β_{ij} with $1 - \exp(-\alpha_{ij})$ where $\alpha_{ij} \geq 0$. Denoting the number of j-type missiles assigned to target i by variable n_{ij}, we obtain the kill probability of target i, $1 - \prod_{j=1}^{m}(1 - \beta_{ij})^{n_{ij}} = 1 - \exp(-\sum_{j=1}^{m} \alpha_{ij} n_{ij})$ and the total cost, $\sum_{j=1}^{m} c_j \sum_{i=1}^{n} n_{ij}$. Therefore, our problem is formulated as the following integer programming problem (P0). Z^+ is a set of non-negative integers.

$$(P0) \quad \max_{n_{ij}} \sum_{i=1}^{n} V_i \left\{ 1 - \exp(-\sum_{j=1}^{m} \alpha_{ij} n_{ij}) \right\} \tag{1}$$

$$s.t. \quad \sum_{j=1}^{m} c_j \sum_{i=1}^{n} n_{ij} \leq C \tag{2}$$

$$n_{ij} \in Z^+,\ i = 1, \cdots, n,\ j = 1, \cdots, m \tag{3}$$

The objective function has a specific structure with the weighted summation of the exponential functions over one suffix which includes the summation of n_{ij} over another

suffix. One of the earlier studies about the assignment problems of missiles is Manne's[4] where the objective function has one suffix and the total number of missiles is restricted. Another one is Lemus's[3]. He studied the objective function given by Eq.(1) with the constraint upon the number of missiles and proposed an approximate algorithm. Nickel[5] extended it and analyzed the effective use of the general-purpose weapon. Concerning with multiple resource allocation problems with the objective function of Eq.(1), some studies have been published so far but constraints are given only upon the number of resources[7]. As you can imagine, cost coefficients $\{c_j\}$ make this problem (P0) difficult to solve.

In the case of $n = 1$, the problem (P0) becomes a knapsack problem of maximizing $\sum_{j=1}^{m} \alpha_{1j} n_{1j}$. It proves that our problem is NP-hard. Assuming that parameters $\{c_j,\ j = 1, \cdots, m\}$ are rational numbers, we multiply both sides of inequality (2) by the least common multiple of their denominators and then make coefficients $\{c_j\}$ be integers. Hereafter we treat $\{c_j\}$ as positive integers.

3. Solution by Dynamic Programming Method

We define a subproblem of (P0) with the number of targets $k \leq n$ and cost limit D as follows.

$$F_k(D) := \max_{\{n_{ij}\}} \left[\sum_{i=1}^{k} V_i \left\{ 1 - \exp\left(- \sum_{j=1}^{m} \alpha_{ij} n_{ij} \right) \right\} \middle| \sum_{i=1}^{k} \sum_{j=1}^{m} c_j n_{ij} \leq D,\ n_{ij} \in Z^+ \right] . \tag{4}$$

We define the optimal value of a knapsack problem by

$$v_k(Q) := \max_{\{n_{kj}\}} \left\{ \sum_{j=1}^{m} \alpha_{kj} n_{kj} \middle| \sum_{j=1}^{m} c_j n_{kj} \leq Q,\ n_{kj} \in Z^+ \right\} . \tag{5}$$

Then we have the following recursion expression.

$$\begin{aligned} F_k(D) &= \max_{Q_k = 0, \cdots, D} \left[\max_{\{n_{kj}\}} \left\{ V_k \left(1 - \exp(- \sum_{j=1}^{m} \alpha_{kj} n_{kj}) \right) \middle| \sum_{j=1}^{m} c_j n_{kj} \leq Q_k,\ n_{kj} \in Z^+ \right\} \right. \\ &\quad \left. + F_{k-1}(D - Q_k) \right] \\ &= \max_{Q_k = 0, \cdots, D} \left\{ V_k \left(1 - e^{-v_k(Q_k)} \right) + F_{k-1}(D - Q_k) \right\} . \end{aligned} \tag{6}$$

Many studies about the integer knapsack problem have been cumulated so far. Gilmore[1] proposed the dynamic programming algorithm by using the periodicity of the knapsack function. By his study, $v_k(Q),\ Q = 0, 1, \cdots, C$ are given as by-products in the process of computing $v_k(C)$ [2]. We use recursively Eq.(6) varying $k = 1, \cdots, n,\ D = 1, \cdots, C$ to obtain $F_n(C)$ in consequence. Denoting the complexity of computing $v_k(C)$ by $knap(C)$, the computation of $F_n(C)$ takes the complexity of order $O(n \cdot (knap(C) + C^2))$. We call the dynamic programming method the DP method for short.

4. Solution by Branch and Bound Method

As known from recursion (6), cost Q_n permitted to missiles which are designated to target n ought to be determined at the first step for evaluating $F_n(C)$. If Q_n is determined, then the decision making of Q_{n-1} within residual cost $C - Q_n$ is the second step. These decision making points look like nodes on an enumeration tree used in the branch and bound method. We propose the branch and bound method by this idea. It is essential to find an effective bound estimation and an approximate solution as exactly as possible for the efficient execution of this method. The approximate algorithm is discussed afterward.

Our branch and bound method proceeds as follows. A root node branches to $C+1$ nodes, $I(1) = \{0, 1, \cdots, C\}$, each of which indicates the cost constraint Q_1 to be designated to target 1. From node Q_1, nodes $I(2) = \{0, \cdots, C - Q_1\}$ are branched. Each of them indicates the

cost constraint Q_2 for target 2. In the result, a $n-$ply layered tree is generated as an enumeration tree. On the sth layer, Q_1, $Q_2, \cdots$, Q_s are already given but $Q_{s+1}, \cdots$, Q_n are not determined yet. By given Q_1, $Q_2, \cdots$, Q_s, the maximum of the expected destroyed value of targets 1, $2, \cdots$, s is calculated, which is saved in $\overline{v}(s)$ in the algorithm. In $Q(s)$, $\sum_{i=1}^{s} Q_i$ is saved. Because the distribution of the residual cost $C-Q(s)$ to targets $s+1, \cdots, n$ is not determined yet, the problem of giving the maximum of the expected destroyed value of targets $s+1, \cdots$, n,

$$G_s(C-Q(s)) := \max_{\{n_{kj}\}} \left\{ \sum_{k=s+1}^{n} V_k \, f\left(\sum_{j=1}^{m} \alpha_{kj} n_{kj}\right) \middle| \sum_{k=s+1}^{n} \sum_{j=1}^{m} c_j n_{kj} \leq C-Q(s), n_{kj} \in Z^+ \right\},$$

is difficult to be solved, where we use notation $f(x) := 1-\exp(-x)$ for convenience. Instead of rigidly solving the problem, we can estimate an upper bound $\widehat{G}_s(C-Q(s))$ of $G_s(C-Q(s))$ by solving a problem of relaxing integer variables $\{n_{kj}\}$ to real numbers and furthermore estimate an upper bound of the optimal value of the original problem (P0) by

$$\widetilde{G} = \overline{v}(s) + \widehat{G}_s(C-Q(s)) \, . \tag{7}$$

(Step1) By an approximate algorithm, find a tentative value of the objective function and store it in f_c. Set $Q(0)=0$, $\overline{v}(0)=0$, $s=1$ and $I(1)=\{0,1,\cdots,C\}$ and go to (Step2).

(Step2) Select a value from $I(s)$, store it in Q_s and set $Q(s)=Q(s-1)+Q_s$. Solve the following knapsack problem

$$v_s(Q_s) = \max_{\{n_{sj}\}} \left\{ \sum_{j=1}^{m} \alpha_{sj} n_{sj} \middle| \sum_{j=1}^{m} c_j n_{sj} \leq Q_s, \; n_{sj} \in Z^+ \right\}$$

and calculate $\overline{v}(s)$ by $\overline{v}(s-1)+V_s \cdot f(v_s(Q_s))$.
If s equals to n, go to (Step7).

(Step3) Solve a relaxed problem of $G_s(C-Q(s))$ to find $\widehat{G}_s(C-Q(s))$ and estimate an upper bound by setting $\widetilde{G} = \overline{v}(s) + \widehat{G}_s(C-Q(s))$.
If $\widetilde{G} \leq f_c$, go to (Step4).
If $\widetilde{G} > f_c$ and the relaxation problem has an integer optimal solution, set $f_c = \widetilde{G}$ and go to (Step4). In the case of $\widetilde{G} > f_c$ and non-integer solution, go to (Step6).

(Step4) Set $I(s) = I(s) - \{Q_s\}$.
If $I(s)$ becomes an empty set, set $s = s-1$ and go to (Step5). Otherwise, go to (Step2).

(Step5) If s equals 0, terminate. Current f_c gives the optimal value.
Otherwise, go to (Step4).

(Step6) Increase s by one or set $s = s+1$.
If s equals n, set $I(s) = \{C-Q(s-1)\}$. Otherwise, set $I(s) = \{0,1,\cdots,C-Q(s-1)\}$. Go to (Step2).

(Step7) If $f_c < \overline{v}(n)$, set $f_c = \overline{v}(n)$. Go to (Step4).

Now we discuss the approximate algorithm and the estimation of an upper bound of the objective function. Hereafter the branch and bound method is occasionally called the BAB method for short.

4.1 Approximate algorithm

For emphasizing the missile assignment $\{n_{ij}, i=1,\cdots,n, j=1,\cdots,m\}$, we provide another form of the objective function (1).

$$H_n(\{n_{ij}\}) := \sum_{i=1}^{n} V_i \left\{ 1-\exp\left(-\sum_{j=1}^{m} \alpha_{ij} n_{ij}\right) \right\} \, . \tag{8}$$

Consider another assignment $\{n'_{ij}\}$. Only the specific variable n'_{kl} is different from n_{kl}, that is $n_{kl}+1$, and all other elements are the same as $\{n_{ij}\}$. Then the ratio of the value increment of the objective function to cost increment is given by the following equation.

$$y_{kl} = \frac{H_n(\{n'_{ij}\}) - H_n(\{n_{ij}\})}{c_l} = \frac{V_k(1-\exp(-\alpha_{kl}))}{c_l} \exp\left(-\sum_{j=1}^{m} \alpha_{kj} n_{kj}\right) . \tag{9}$$

This ratio gives us a measure of efficiency about the increment of the number of missiles. Now we propose an approximate algorithm, say a greedy algorithm, as follows. Starting from $\{n_{ij}\} = \{0\}$, n_{kl} which satisfies constraint (2) and maximizes y_{kl}, is increased by one. This process is repeated until constraint (2) is violated. As known from Eq.(9), the increment of n_{kl} makes each of ratios $\{y_{kj}, j = 1, \cdots, m\}$ multiplied by $\exp(-\alpha_{kl})$. Furthermore, the ordering among y_{kj} is never changed for $j = 1, \cdots, m$. On this process, if the increment of n_{kl} is not permitted by cost constraint, which means more c_l is beyond the cost permission, $\{n_{il}, i = 1, \cdots, n\}$ are no longer candidates for being increased. We itemize our greedy algorithm as follows.

(Step1) For $i = 1, \cdots, n$, $j = 1, \cdots, m$, calculate the measure of efficiency by

$$y_{ij} = \frac{V_i}{c_j}(1 - \exp(-\alpha_{ij})) .$$

Initialize parameters as follows.

$$\{n_{ij}\} = \{0\} , \quad B = C , \quad F = \{1, 2, \cdots, m\} .$$

(Step2) Set $F = F - \{j \in F \mid B - c_j < 0\}$.
If F is $\emptyset$, then terminate. Current $\{n_{ij}\}$ gives an approximate solution.

(Step3) Select (k, l) by

$$y_{kl} = \max_{j \in F, i} y_{ij} ,$$

and revise parameters as follows.

$$n_{kl} = n_{kl} + 1 , \quad B = B - c_l , \quad y_{kj} = y_{kj} \exp(-\alpha_{kl}) , \quad j = 1, \cdots, m .$$

Go to (Step2).

4.2 Upper bound estimation

Here we solve the relaxation of $G_s(C - Q(s))$ in (Step3) of the BAB procedure. To avoid the complicated presentation, we discuss the relaxation of (4), which is essentially the same problem as $G_s(C - Q(s))$.

$$\widehat{F}_k(D) := \max_{\{n_{ij}\}} \left[\sum_{i=1}^{k} V_i \left\{ 1 - \exp(-\sum_{j=1}^{m} \alpha_{ij} n_{ij}) \right\} \middle| \sum_{i=1}^{k} \sum_{j=1}^{m} c_j n_{ij} \le D, \ n_{ij} \ge 0 \right] . \tag{10}$$

An optimal solution is given by the following theorem.

Theorem 1. *For $i = 1, \cdots, k$, set $\sigma_i = \dfrac{\alpha_{ij^*(i)}}{c_{j^*(i)}} = \max_j \dfrac{\alpha_{ij}}{c_j}$. Then an optimal solution $\{n_{ij}, i = 1, \cdots, k, \ j = 1, \cdots, m\}$ of the relaxation (10) is presented by the following formulae.*

$$n_{ij^*(i)} = \frac{1}{c_{j^*(i)}\sigma_i} \left[\log \frac{V_i \sigma_i}{\lambda} \right]^+ , \tag{11}$$

$$n_{ij} = 0, \ j \neq j^*(i) . \tag{12}$$

λ is a Lagrangean multiplier uniquely determined by the next equation.

$$\sum_{i=1}^{k} \frac{1}{\sigma_i} \left[\log \frac{V_i c_i}{\lambda} \right]^+ = D . \tag{13}$$

$[x]^+$ denotes $\max\{0, x\}$.

Proof: By using some notations

$$m_{ij} := c_j n_{ij},\ \mu_{ij} := \frac{\alpha_{ij}}{c_j},\ \sigma_i := \mu_{ij^*(i)} = \max_{j=1,\cdots,m} \mu_{ij}\ ,$$

$\widehat{F}_k(D)$ can be transformed as follows.

$$\begin{aligned}
\widehat{F}_k(D) &= \max_{\{m_{ij}\}} \left\{ \sum_{i=1}^{k} V_i f\Big(\sum_{j=1}^{m} \frac{\alpha_{ij}}{c_j} m_{ij}\Big) \Big| \sum_{i=1}^{k}\sum_{j=1}^{m} m_{ij} \le D,\ m_{ij} \ge 0 \right\} \\
&= \max_{\{m_{ij}\}} \left\{ \sum_{i=1}^{k} V_i f\Big(\sum_{j=1}^{m} \mu_{ij} m_{ij}\Big) \Big| \sum_{i=1}^{k} q_i \le D,\ q_i = \sum_{j=1}^{m} m_{ij},\ m_{ij} \ge 0 \right\} \\
&= \max_{\{q_i, m_{ij}\}} \left\{ \sum_{i=1}^{k} V_i f(\sigma_i q_i) \Big| \sum_{i=1}^{k} q_i \le D,\ m_{ij^*(i)} = q_i \ge 0,\ m_{ij} = 0\ (for\ j \ne j^*(i)) \right\} \\
&= \max_{\{q_i\}} \left\{ \sum_{i=1}^{k} V_i f(\sigma_i q_i) \Big| \sum_{i=1}^{k} q_i = D,\ q_i \ge 0 \right\}\ . \qquad (14)
\end{aligned}$$

It is easy to solve this maximization by using the Kuhn-Tucker condition. A Lagrangean function $L(\{q_i\})$ is defined by the following.

$$L(\{q_i\}) := \sum_{i=1}^{k} V_i (1 - \exp(-\sigma_i q_i)) + \lambda \left(D - \sum_{i=1}^{k} q_i \right) + \sum_{i=1}^{k} \nu_i q_i\ . \qquad (15)$$

The following conditions are necessary and sufficient for an optimal solution.

$$\frac{\partial L}{\partial q_i} = V_i \sigma_i e^{-\sigma_i q_i} - \lambda + \nu_i = 0,\quad i = 1, \cdots, k\ , \qquad (16)$$

$$\nu_i \ge 0,\ q_i \ge 0,\quad i = 1, \cdots, k\ , \qquad (17)$$

$$\nu_i q_i = 0,\quad i = 1, \cdots, k\ , \qquad (18)$$

$$\sum_{i=1}^{k} q_i = D\ . \qquad (19)$$

Using these conditions, $q_i > 0$ results in $\nu_i = 0$ and $V_i \sigma_i e^{-\sigma_i q_i} = \lambda$. The case of $q_i = 0$ results in $\nu_i \ge 0$ and $V_i \sigma_i e^{-\sigma_i q_i} = \lambda - \nu_i \le \lambda$. Therefore, an optimal solution of $\{q_i\}$ is presented by the following equation.

$$q_i = \frac{1}{\sigma_i} \left[\log \frac{V_i \sigma_i}{\lambda} \right]^+\ . \qquad (20)$$

In the result, an optimal solution of $\{n_{ij}\}$ is given by Eqs.(11) and (12). Multiplier λ is determined uniquely by Eq.(19) or

$$g(\lambda) := \sum_{i=1}^{k} \frac{1}{\sigma_i} \left[\log \frac{V_i \sigma_i}{\lambda} \right]^+ = D\ . \qquad (21)$$

Q.E.D.

We explain about a concrete procedure of computing $\{n_{ij}\}$ and $\widehat{F}_k(D)$. Without loss of generality, assume that $V_1\sigma_1 \ge V_2\sigma_2 \ge \cdots \ge V_k\sigma_k$. After finding $\overline{\lambda} = V_1\sigma_1 = \max_i V_i\sigma_i$, $\underline{\lambda} = \max_i V_i\sigma_i e^{-\sigma_i D}$, Eq.(21) holds for a finite λ satisfying $\overline{\lambda} > \lambda > \underline{\lambda}$ because $g(\overline{\lambda}) = 0$, $g(\underline{\lambda}) \ge D$ and $g(\lambda)$ is non-increasing for variable λ. Therefore, we can determine l^* uniquely, l^* satisfying $g(V_{l^*}\sigma_{l^*}) < D \le g(V_{l^*+1}\sigma_{l^*+1})$ or $l^* = k$ satisfying $g(V_k\sigma_k) < D$. Then Eq.(21) becomes the following.

$$\sum_{i=1}^{l^*} \frac{1}{\sigma_i} \log(V_i\sigma_i) - \left(\sum_{i=1}^{l^*} \frac{1}{\sigma_i} \right) \log\lambda = D\ .$$

By using this relation, λ and $\widehat{F}_k(D)$ can be calculated.

$$\lambda = \exp\left\{ \left(\sum_{i=1}^{l^*} \frac{1}{\sigma_i} \log(V_i\sigma_i) - D \right) \Big/ \sum_{i=1}^{l^*} \frac{1}{\sigma_i} \right\}\ , \qquad (22)$$

$$\widehat{F}_k(D) = \sum_{i=1}^{l^*} \left(V_i - \frac{\lambda}{\sigma_i} \right) . \tag{23}$$

5. Numerical Examples

Here, we investigate the characteristics of optimal solutions and the computational efficiency of the proposed methods. We take missile assignment problems as comprehensive examples. By the first example, we investigate the alternative strategy among missiles for special use and for general purpose. By the second and the third examples, the sensitivity of the proposed two methods are analyzed for the change of the cost limit C and the number of targets or missiles in terms of computational time.

5.1 Alternative strategy among some types of missiles

Here we examine some characteristics of optimal solutions. Generally speaking, if the constraints of the integer problem are loose, optimal integral solutions become little different from optimal continuous solutions of the problem relaxed by changing integral variables to continuous ones. Then, in some examples, we impose tight constraints on available total cost so that small C is set comparing with the unit cost c_j and elucidate interesting properties of optimal integral solutions.

Consider 4 targets with their values $V_1 \leq V_2 \leq V_3 \leq V_4$ and 5 types of missiles, that is, $n = 4$ and $m = 5$. The fifth type of missiles is manufactured for general purpose. It is comparatively effective to each of 4 targets and cheap. Each of other types of missiles is most effective to a specific target and a little expensive. The i-type missile for special purpose has the highest SSKP to target i but lower SSKP to other targets. We analyze the optimal assignment of missiles by varying parameters.

(Case1) This is a basic case.

- Target value: $V_1 = 2,\ V_2 = 4,\ V_3 = 6,\ V_4 = 8$
- Unit prices: $c_1 = 2,\ c_2 = 3,\ c_3 = 4,\ c_4 = 5,\ c_5 = 1$
- SSKPs: $\beta_{ii} = 0.7,\ \beta_{ij} = 0.1\ (i \neq j),\ i, j = 1, \cdots, 4,\ \beta_{i5} = 0.2,\ i = 1, \cdots, 4$

Changing the total cost limit to $C = 10, 12, 14, 16, 18, 20$, optimal assignments $\{n_{ij}\}$ are presented by Table 1. As seen from this table, the optimal solution consists of assigning special-purpose missiles at first and then covering residual cost permission by cheap missile for general purpose.

(Case2) In this case, only SSKPs of general-purpose missile of Case 1 increase a little and other parameters are the same as Case 1. An optimal solution is given in Table 2.

- SSKPs: $\beta_{i5} = 0.3,\ i = 1, \cdots, 4$

The optimal assignment changes as follows. The assignment of the 1st and 2nd types of missiles is preferable, which is the same as Case 1, but the adoption of more expensive 3rd and 4th types of missiles is likely to be restrained. Instead of the 3rd and 4th types of missiles, many general-purpose missiles are concentrically assigned to targets 3 and 4 in order to assure the high expected destroyed value.

(Case3) This is the case in which unit prices of missiles of Case 2 increase by one and other parameters are the same as Case 2. Optimal solutions are shown in Table 3.

- Unit prices: $c_1 = 3,\ c_2 = 4,\ c_3 = 5,\ c_4 = 6,\ c_5 = 2$

In this example, the general-purpose missile plays both of a complementary and main role. In the case of $C = 10$, the cost constraint could permit us to buy one

Table 1. Optimal Allocation of Case 1

$C=10$: $F_n(C)=10.9$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	0	3

$C=12$: $F_n(C)=12.6$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

$C=14$: $F_n(C)=14.0$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

$C=16$: $F_n(C)=14.9$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	2

$C=18$: $F_n(C)=15.5$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	1
4	0	0	0	1	3

$C=20$: $F_n(C)=16.1$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	2	0	0
4	0	0	0	1	2

Table 2. Optimal Allocation of Case 2

$C=10$: $F_n(C)=12.8$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	3
4	0	0	0	0	4

$C=12$: $F_n(C)=14.2$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	3
4	0	0	0	0	4

$C=14$: $F_n(C)=15.4$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	4
4	0	0	0	0	5

$C=16$: $F_n(C)=16.3$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	5
4	0	0	0	0	6

$C=18$: $F_n(C)=16.9$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	1
3	0	0	0	0	6
4	0	0	0	0	6

$C=20$: $F_n(C)=17.4$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	2
3	0	0	0	0	6
4	0	0	0	0	7

Table 3. Optimal Allocation of Case 3

$C=10$: $F_n(C)=8.7$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	1
4	0	0	0	0	2

$C=12$: $F_n(C)=10.2$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	1
4	0	0	0	1	0

$C=14$: $F_n(C)=11.5$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	2
4	0	0	0	1	0

$C=16$: $F_n(C)=12.6$					
Target No.	Missile Types				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

$C=18$: $F_n(C)=14.0$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

$C=20$: $F_n(C)=14.7$					
Target No.	Missile Types				
	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	1

more 4th type of missile or two more 1st type of missiles instead of 3 general-purpose missiles. In this case, the general-purpose missile is of main use. In cases of other cost limits, it is of complementary use. The usage of the general-purpose weapon depends not only on its cost-performance but also on other missiles' cost-performance. In addition, the solution is perturbed a little by the attribute of its being integer.

5.2 Computational analysis for two methods

Two methods, the DP method and the BAB method, are proposed in the previous section. Here, we are interested in comparison between the computational time of them. A mainframe computer HITACHI S3600/120A and FORTRAN 77 are used as the computer hardware and language.

(1) Change of cost permission

With varying the cost limit C from 40 to 200 in Case 1, CPU-time(seconds) for both methods are measured and presented by Table 4. In data for the BAB method, CPU-time of using the Greedy algorithm and solving the relaxed problems are all included. For the sake of comparison, the result by the total enumeration method (TE method) on the enumeration tree and the result by the greedy algorithm are written also. Not only CPU-time but also the relative error of the value of the greedy solution to an optimal one and the number of branches on the enumeration tree are both obtained for the BAB method. From the discussion of Section 3, it is comprehensible that computational time increases as C increases for the DP method. However, for the branch and bound method, it depends on the precision of the greedy solution and the efficiency of the bounding process and so it varies case by case. From this example, the increase of CPU-time appears in the case of $C = 40 \sim 120$. On the contrary, CPU-time decreases in the case of $C = 140 \sim 180$. This is caused mainly by the fact

Table 4. CPU-Time(sec.) of Case 1 with varying C

Method	Items	Cost permission : C								
		40	60	80	100	120	140	160	180	200
DP	CPU-Time	5.0×10^{-3}	1.0×10^{-2}	1.7×10^{-2}	2.5×10^{-2}	3.6×10^{-2}	4.8×10^{-2}	6.1×10^{-2}	7.6×10^{-2}	9.2×10^{-2}
BAB	CPU-Time	1.5×10^{-2}	2.2×10^{-2}	4.6×10^{-2}	6.1×10^{-2}	9.3×10^{-2}	7.9×10^{-2}	2.7×10^{-2}	7.5×10^{-3}	8.3×10^{-3}
	♯ of branches	851	1183	2661	3497	5386	4167	983	101	101
Greedy	CPU-Time	3.8×10^{-4}	5.1×10^{-4}	6.8×10^{-4}	8.2×10^{-4}	9.7×10^{-4}	1.1×10^{-3}	1.3×10^{-3}	1.4×10^{-3}	1.6×10^{-3}
	Relative errors	1.5×10^{-2}	4.4×10^{-3}	1.1×10^{-3}	3.1×10^{-4}	7.2×10^{-5}	1.7×10^{-5}	3.1×10^{-6}	9.4×10^{-8}	2.9×10^{-8}
TE	CPU-Time	4.5×10^{-2}	1.4×10^{-1}	3.2×10^{-1}	6.1×10^{-1}	1.0×10^{0}	1.6×10^{0}	2.4×10^{0}	3.5×10^{0}	4.7×10^{0}
	♯ of branches	2.6×10^{4}	8.1×10^{4}	1.9×10^{5}	3.6×10^{5}	6.1×10^{5}	9.6×10^{5}	1.4×10^{6}	2.0×10^{6}	2.8×10^{6}
$F_n(C)$		19.33	19.88	19.98	20.00	20.00	20.00	20.00	20.00	20.00

that the relative error of the greedy solution becomes extremely small, which makes the bounding process more efficient and the number of branches more less. In Table 4, $F_n(C)$ in the case of $C \geq 100$ is nearly equal to the optimal value without cost constraint, which is $V_1 + V_2 + V_3 + V_4 = 20$. Therefore, the optimal condition in the case of large C is not so tight that there are many good solutions around the optimal one and perhaps the greedy solution is one of them. This computational characteristic about the BAB method generally appears in other cases.

(2) Change of the number of targets or the number of missile types

Here, we analyze the sensitivity of n and m to the computational time for two methods. We vary n from 2 through 16, m from 2 through 16 with fixing $C = 50$. Other parameters are determined as follows. For a combination of n and m, we randomly select the unit prices of each type of missile c_is, values of targets V_is and SSKPs β_{ij}s, during $[1, 10]$, $[1, 10]$ and $[0.01, 0.9]$, respectively. By this way, we make 10 problems and measure CPU-times for solving these problems by each of two methods. CPU-times in Table 5 are presented by the mean value. The upper figures are data for the

Table 5. CPU-Time(sec.) with varying n and m

♯ of missiles	♯ of targets							
	2	4	6	8	10	12	14	16
2	2.2×10^{-3}	5.8×10^{-3}	9.5×10^{-3}	1.3×10^{-2}	1.7×10^{-2}	2.1×10^{-2}	2.4×10^{-2}	2.8×10^{-2}
	1.1×10^{-3}	1.3×10^{-2}	3.9×10^{-2}	1.7×10^{-1}	1.6×10^{-1}	8.7×10^{-1}	7.1×10^{0}	7.2×10^{0}
4	2.4×10^{-3}	6.3×10^{-3}	1.0×10^{-2}	1.4×10^{-2}	1.8×10^{-2}	2.2×10^{-2}	2.6×10^{-2}	3.0×10^{-2}
	1.2×10^{-3}	8.7×10^{-3}	1.5×10^{-2}	2.3×10^{-1}	3.3×10^{-1}	1.7×10^{0}	2.9×10^{0}	9.5×10^{0}
6	2.7×10^{-3}	7.0×10^{-3}	1.1×10^{-2}	1.5×10^{-2}	1.9×10^{-2}	2.3×10^{-2}	2.7×10^{-2}	3.1×10^{-2}
	1.5×10^{-3}	7.2×10^{-3}	3.7×10^{-2}	8.9×10^{-2}	3.3×10^{-1}	1.6×10^{0}	2.7×10^{0}	5.0×10^{0}
8	3.0×10^{-3}	7.5×10^{-3}	1.2×10^{-2}	1.6×10^{-2}	2.0×10^{-2}	2.5×10^{-2}	2.9×10^{-2}	3.4×10^{-2}
	1.6×10^{-3}	7.5×10^{-3}	7.1×10^{-2}	1.3×10^{-1}	3.9×10^{-1}	1.1×10^{0}	4.2×10^{0}	12.0×10^{0}
10	3.1×10^{-3}	8.1×10^{-3}	1.3×10^{-2}	1.7×10^{-2}	2.2×10^{-2}	2.6×10^{-2}	3.1×10^{-2}	3.5×10^{-2}
	1.7×10^{-3}	8.7×10^{-3}	1.3×10^{-2}	1.0×10^{-1}	2.9×10^{-1}	1.2×10^{0}	1.9×10^{0}	4.4×10^{0}
12	3.5×10^{-3}	8.7×10^{-3}	1.3×10^{-2}	1.8×10^{-2}	2.3×10^{-2}	2.8×10^{-2}	3.2×10^{-2}	3.7×10^{-2}
	2.0×10^{-3}	6.2×10^{-3}	3.1×10^{-2}	1.0×10^{-1}	3.4×10^{-1}	7.9×10^{-1}	1.3×10^{0}	4.6×10^{0}
14	3.8×10^{-3}	9.5×10^{-3}	1.5×10^{-2}	2.0×10^{-2}	2.5×10^{-2}	3.0×10^{-2}	3.4×10^{-2}	3.9×10^{-2}
	2.0×10^{-3}	8.3×10^{-3}	3.3×10^{-2}	8.6×10^{-2}	4.0×10^{-1}	8.0×10^{-1}	3.0×10^{0}	4.3×10^{0}
16	4.1×10^{-3}	9.6×10^{-3}	1.5×10^{-2}	2.1×10^{-2}	2.6×10^{-2}	3.1×10^{-2}	3.5×10^{-2}	4.1×10^{-2}
	2.3×10^{-3}	4.9×10^{-3}	2.1×10^{-2}	9.1×10^{-2}	4.2×10^{-1}	7.7×10^{-1}	1.3×10^{0}	1.6×10^{0}

DP method and the lower figures for the BAB method. Data for the TE method are omitted since it is clearly inferior to other two methods in any case.

Figure 1-1 shows how the number of the targets changes the CPU-time by the DP method in the cases of the number of missile types $m = 2$, 8 and 14. As you see, the CPU-time for the DP method shows a linear increase for the number of targets n, which can be estimated from the discussion about the computational complexity in Section 3. Figure 1-2 shows the change for the BAB method but it uses the logarithm as a unit of the axis of ordinate. That is, the change of CPU-times is very steep as the number of the targets increases. This is by the reason that the enumeration tree of the BAB method has the depth of n and the number of the branching or its leaves is enlarged by the n-th power. However, only concerning with the greedy solution, its relative error is less than 2.0×10^{-2} over the whole cases and the greedy solution can be obtained easily. In the case of $n = 10$ and $m = 10$, the greedy algorithm requires 1.4×10^{-3} seconds of CPU-time and yields only 3.4×10^{-3} of the relative error on the average.

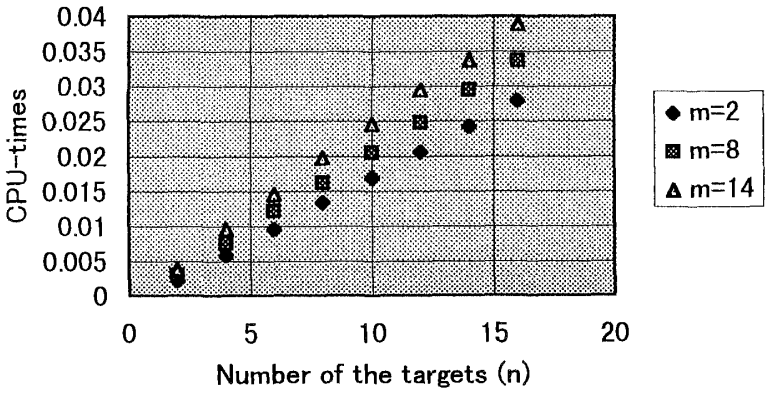

Fig.1-1 CPU-time for the DP method

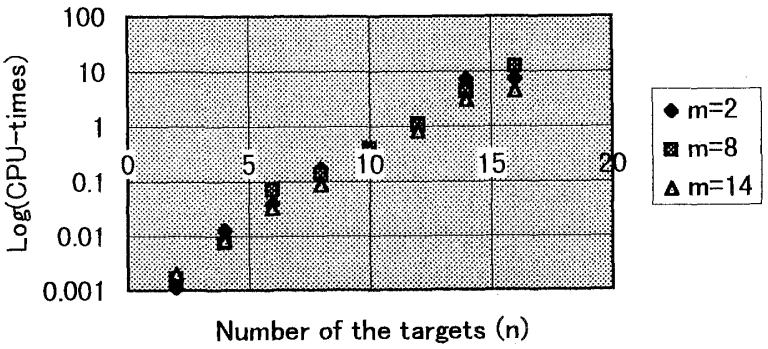

Fig.1-2 CPU-time for the BAB method

The number of items in a knapsack problem to be solved on the way is nothing but m in our problem and the total number of instances of the knapsack problem are at

most as much as the number of targets for each method. Therefore, the increase of the CPU-time by the increase of m is mainly caused by the fact that the size of the knapsack problem becomes larger. The CPU-time increases as m is larger but its rate is a little. This is the fact for the DP method, which is seen in Table 5. For the BAB method, this is true for small n but not true for large n. In the case of large n, the change of system parameters m, n, C, c_i, α_{ij} produces much effects on the number of searched nodes on its large enumeration tree. By this reason, the CPU-time does not necessarily increase by m, remaining its increasing or decreasing rates small. In fact, the variance of CPU-times becomes larger as n increases, which means that the CPU-time varies irregularly for each problem. For example, the variances for the DP and the BAB methods are 1.0×10^{-7} and 0.8×10^{-7}, respectively, for $m = 10$, $n = 2$ and it becomes 2.0×10^{-7} and 8.1×10^{-2}, respectively, for $m = 10$, $n = 10$

In the previous two examples concerning with the computational analysis, it seems that we take comparatively small size of problems. However, as the practical missile assignment problem or the practical search problem, they have enough large size. And it is thought that results obtained by the previous examples give a guideline for applying the proposed methods to other unknown size of problems. Now, we are on the position to itemize the characteristics of the methods.

(1) For the dynamic programming method,
  (i) Cost constraint C indicates a capacity of the knapsack problem as a subproblem. The computational time increases linearly or a little more strongly by the increase of C.
  (ii) The number of the types of missiles indicates the number of items included in the knapsack problem. The increasing rate of the computational time by m is small.
  (iii) The number of the targets indicates the number of repeats in the recursive equation (6). The increasing rate of the computational time by n is a little more than linear.
  (iv) In the middle- and large-size problems for m and n, the DP method is superior to the BAB method if the exact solutions are required.

(2) For the branch and bound method,
  (i) The increasing rate of the computational time by C is as small as the DP method.
  (ii) The effect of m on the computational time is small comparing with other system parameters' effects.
  (iii) In many cases, the number of targets has the effect of the exponential increase on the computational time because it determines the depth of the enumeration tree.
  (iv) In the following case, the bounding procedure of the BAB method has a good effect on the computational time because the relative errors of the greedy solutions to the exact solutions are small.
    - In the case that the cost constraint C is large comparing with the unit costs $\{c_i\}$ and the number of the missiles can be regarded as real values.
    - In the case that the SSKPs or $\{\alpha_{ij}\}$ are large and the small difference of $\{n_{ij}\}$ has not so large effect on the values of the objective function.

6. Conclusions

This paper deals with an integer programming problem with cost constraint and an objective function of the exponential form and proposes two methods, the dynamic programming

method and the branch and bound method, to exactly solve the problem. The proposed dynamic programming method is available to the general objective function that has the form of $\sum_{i=1}^{n} f_i\left(\sum_{j=1}^{m} \alpha_{ij} n_{ij}\right)$. For the branch and bound method, if the function $f_i(\cdot)$ is concave and non-decreasing, the relaxed solution might be derived easily as it is done in Theorem 1.

Our problem is NP-hard, which has been proved, and to shorten its computational time is one of the interesting points. By numerical examples, we can say the followings concerning with the computational time. The dynamic programming method has the linear increasing feature for the number of targets n and the number of types of missiles m, and the squared increasing feature of the cost permission C. For the branch and bound method, there is some possibility that the computational time decreases in spite of the increasing cost permission. However it has the steep increasing feature for n, which ought to be improved further. In result, the dynamic programming method is superior to the branch and bound method in many cases. However, the greedy algorithm used in the branch and bound method gives good approximate solutions with small CPU-times, especially in large-size problems.

References

[1] P.C. Gilmore and R.E. Gomory: The theory and computation of Knapsack function. *Opns. Res.*, **14** (1966) 1045–1074.

[2] R.S. Garfinkel and G.L. Nemhauser: *Integer programming* (John Wiley & Sons, 1972).

[3] F. Lemus and K.H. David: An optimal allocation of different weapons to a target complex. *Opns. Res.*, **11** (1963) 787–794.

[4] A.S. Manne: A target-assignment problem. *Opns. Res.*, **7** (1959) 258–260.

[5] R.H. Nickel and M. Mangel: Weapon acquisition with target uncertainty. *Naval Research Logistics*, **32** (1985) 567–588.

[6] J.B. Kadane: Discrete search and the Neyman-Pearson lemma. *J. of Mathematical Analysis and Applications*, **22** (1968) 156–171.

[7] T. Ibaraki and N. Katoh: *Resource allocation problem* (The MIT Press, London, 1988).

Ryusuke Hohzaki
Department of Applied Physics
National Defense Academy
1-10-20 Hashirimizu, Yokosuka,
239-8686, Japan
E-mail: hozaki@cc.nda.ac.jp