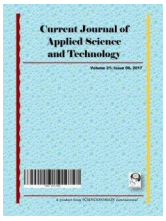

Current Journal of Applied Science and Technology

26(5): 1-21, 2018; Article no.CJAST.40420
ISSN: 2457-1024
(Past name: British Journal of Applied Science & Technology, Past ISSN: 2231-0843, NLM ID: 101664541)

Exact Solution Algorithms for Multi-dimensional Multiple-choice Knapsack Problems

Farhad Ghassemi-Tari[1*], Hamed Hendizadeh[2] and Gary L. Hogg[3]

[1]Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran.
[2]Department of Mechanical and Manufacturing Engineering, Faculty of Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.
[3]Department of Industrial Engineering, Arizona State University, USA.

Authors' contributions

This work was carried out in collaboration between all authors. Author FGT designed the study, proposed the algorithms, wrote the protocol and wrote the first draft of the manuscript. Author HH performed the computational analysis and Author GLH managed the literature searches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/CJAST/2018/40420
Editor(s):
(1) Wei Wu, Professor, Department of Applied Mathematics, Dalian University of Technology, China.
Reviewers:
(1) Sridevi, Karnatak University, India.
(2) Pasupuleti Venkata Siva Kumar, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/24014

Original Research Article

Received 19th January 2018
Accepted 29th March 2018
Published 6th April 2018

ABSTRACT

We propose a first depth dual approach branch and bound (B&B) routine for solving the general form of multi-dimensional multiple-choice knapsack problems (MMKPs).We call this approach a discriminatory branch and bound method. This name is chosen due to the selection of a node for further branching based on a discriminatory criterion. Three selection mechanisms are developed and are incorporated in an algorithmic procedure to develop three algorithms. An extensive computational experiment is performed, to compare the number of nodes enumerated by the proposed algorithms against the traditional B&B. The results revealed that all the proposed algorithms lead to a considerable node enumeration reduction.

Keywords: Multiple-dimensional multiple-choice knapsack; Branch and bound; Exact solution approach; Selective branching mechanisms.

**Corresponding author: E-mail: ghasemi@sharif.edu;*

1. INTRODUCTION

The basic 0-1 knapsack problem takes into account a number of objects, each with a profit value and a resource cost designated by the weight. The objective is to place a set of objects in a knapsack so that the total value of packed items, subject to the resource capacity of the knapsack is maximized. The multiple-choice knapsack problem (MCKP) is a generalization of the ordinary knapsack problem in which, the set of items is partitioned into classes. The binary choice of taking an item is replaced by the selection of exactly one item out of each class of items. Ultimately, another variant of the problem is defined as the multidimensional multiple-choice knapsack problem (MMKP) where a multi-dimensional resource cost is considered. This is a well-known NP-hard combinatorial optimization problem with a vast number of applications [1,2]. Many practical and real-life problems are modeled as the MMKP. The most typical application of MMKP is on portfolio optimization, where two or more resource constraints such as; total budget, cumulative risk or other restraints need to be satisfied [3]. Other applications of MMKP have been reported in solving the problems such as Stigler diet [4], the cargo loading [5] the capital budgeting [6] the bin packing [7] the bidding in ad auctions [8] and many other problems.

Solving the large-size MMKPs optimality, by B&B algorithms, is a CPU time intensive. Although B&B allows considerable reduction in solution space, the exploration time using a bounding mechanism remains significant. Therefore, the use of solution space reduction to speed up the execution time has become a significant resolution. B&B algorithms are characterized by four basic operations known as, branching, bounding, selection and elimination. In order to reach higher computational performance, we have focused on some characteristics of MMKP for the reduction of the solution spaces.

In this paper, we have proposed three branching mechanisms to enhance the computational efficiency of the B&B algorithm for solving MMKPs. These mechanisms can be applied to the general form of MMKPs. An algorithmic procedure, based on the B&B with three different selective branching mechanisms is proposed for the reduction of the solution space of MMKPs. The proposed algorithm is called “discriminatory Branch and bound” (DBB)) due to the fact that a selection mechanism is employed for the branching procedure of the traditional B&B. Through this mechanism, a vast number of the solution space is enumerated implicitly, which extensively reduces the explicit enumeration of nodes, compared to the traditional B&B.

After the introduction (Section 1) we present the state of art of the problem by reviewing the existing literatures (Section 2). In Section 3 the scope of the problem including the mathematical model of the MMKP is presented. Section 4 is devoted to the illustration of three different variants of the proposed B&B algorithm. The computational experiments, through which the solution space reduction of the proposed algorithms is evaluated, will be presented in Section 5. Finally, the paper is concluded in Section 6.

2. LITERATURE REVIEW

Exact solution approaches for the MMKP evolved during many decades and encompassed the Lagrangian and surrogate relaxation technique. Special enumeration techniques and reduction schemes such as dynamic programming, and B&B methods are studied. Many early research attempts in the general field of zero-one mathematical programs and specifically the knapsack problems have provided some framework for the computational burden of the MMKPs. Among these early attempts is work of Balas [9] who proposed a B&B algorithm for solving a mathematical program with zero-one variables. Two classes of the exact solution approaches are mostly devoted to the reduction of the solution spaces through the subrogate constraint and Lagrangian relaxation mechanisms. Both surrogate constraint and Lagrangian relaxation mechanism are engaged for converting the multidimensional knapsack to a one-dimensional knapsack problem. The surrogate approach is first presented by Glover [10] by substituting the original constraints by one surrogate constraint. Greenberg and Pierskalla [11] speculated the first principal handling of surrogate constraints in the setting of general mathematical programs. This study is then thrived by the works conducted in [12-16].

Freville [17] argued that although further effort is needed to compute the bounds, the methods of surrogate relaxation are more beneficial in resolving the MMKP than the methods which use the Lagrangian relaxation. Shih [18] presented a B&B algorithm by use of the particular MMKP structure and found a top bound through

resolving m- single-constrained knapsack problems instead of an m-constrained problem. Gavish and Pirkul [19] reached to the conclusion that the major deficiencies of Shih's method included its extreme space requirements as well as its inability to resolve problems of tight resource restraints.

Another powerful approach for solving MMKP is dynamic programming (DP) techniques. A hybrid algorithm by integration of the DP procedure, surrogate constraints routine, and several other bounding schemes for reducing the state space solution has been presented in [20-21]. Dyer et al. [22] proposed a hybrid dynamic algorithm for the MMKP. Bao and Yang [23] presented a solution algorithm based on DP for the solution of the MMKP. Also, various other DP approach for solving the MMKP are given in [24-29].

Lagrangian duality is incorporated in a computationally efficient manner to compute tight bounds on every active node of the search tree. Vercellis [30] presented a Lagrangian decomposition technique for solving multi-project planning problems. Later Pisinger [31] proposed a simple partitioning algorithm for developing the optimal linear solution. In contrast to the most of the methods of reducing the solution space, the proposed approach can solve the linear MMKP without sorting of the solution space in enumeration processes. Later, Pisinger [32] presented an algorithm for the budgeting problem. The proposed problem is transformed to an equivalent MMKP. The results of the computational experiments revealed that the developed algorithm was order of magnitude faster than a general LP/MIP algorithm.

Almost all successful exact methods proposed for MMKP are based on the B&B approaches. Cherfi & Hifi [33,34] used a search tree to represent the solution space and proposed a linear programming to find tighter bounds. In this work, the use of a column generation and a local search meta-heuristic for solving the MMKP are also introduced. Lin and Bricker [35] reviewed the theoretical background of two partitioning strategies, the "weighted-mean method" and the "reformulation & transformation technique", incorporated in the B&B procedure. Sung and Cho [36] developed a B&B method for reliability optimization of a series systems with multiple-choice and budget constraints. Kozanidis and Melachrinoudis [37] proposed a B&B algorithm for the 0-1 mixed integer knapsack problem with linear multiple choice constraints. Ohtagaki et al. [38] introduced a surrogate multiplier for transforming the multidimensional nonlinear knapsack problem into a one-dimensional problem. Hifi et al. [39] also presented an optimal algorithm for the MMKP. The main principle of the approach is twofold: (i) to generate an initial solution, and (ii) at different levels of the tree search to determine a new upper bound used with a best-first search strategy. A more efficient B&B algorithm for solving the MMKP was proposed by Sbihi [40]. In this research some selected items were kept fixed while by the use of linear programming the bounds were computed during the searching procedure.

Razzazi and Ghasemi [41] presented a B&B algorithm to solve the MMKP. The B&B tree is arranged based on the orderings in the core and navigated in a depth-first manner, in which consuming low memory effectively causes the core to be expanded by need. Obtaining a tight upper bound for the 0-1 quadratic knapsack problem has been considered in [42]. In this article the problem of a wind farm layout optimization is modeled by a 0-1 knapsack model and an upper bound proposed for the model. Nirmala and Parvathi [43] proposed a mechanism for radio access network selection based on MMKP. They devised a DP and a heuristic to solve the above network selection problem. Cokce and Wilhelm [44] combined DP and B&B to produce a hybrid algorithm for the MMKP.

To find an exact solution of the MMKP, Goyal and Parashar [45] modeled the solution space as a tree and then navigated the tree exploring the most promising sub-trees first. They contemplated a DP and a B&B to solve the proposed problem. Bettinelli et al. [46] studied the MMKP with conflict graph to select the maximum profit set of compatible items while satisfying the knapsack capacity constraint. They presented a new B&B to derive optimal solutions to the problem in short computing times. Wang [47] considered concave knapsack problems with integer variables and presented an exact and efficient algorithm. The proposed algorithm combines the contour cut with a special cut to improve the lower bound and reduce the duality gap gradually in the iterative process. The lower bound of the problem is obtained by solving a linear relaxation version of the problem. A special cut is performed by exploiting the structures of the objective function and the feasible region of the primal problem. The computational results showed that the

algorithm is efficient. In spite of these efforts, the need for solution space reduction of the MMKP is still a challenging issue.

The presented literatures provided a basic understanding of methods for improving the computational efficiency of the MMKP and motivated further investigation of introducing a more efficient solution approach. In this paper we proposed three branching mechanisms in a core algorithmic framework to enhance the computational efficiency of the B&B algorithm for solving the MMKP. Based on these mechanisms three B&B algorithms are proposed. Before presenting these algorithms, the scope of the problem is introduced in the following section.

3. THE SCOPE OF THE PROBLEM

The scope of the multiple-choice knapsack problems is as follows:

$$Maximize\ z = \sum_{j=1}^{n}\sum_{k=1}^{K_j} c_{jk}x_{jk},$$

$$subject\ to:$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K_j} a_{ijk}x_{jk} \le B_i,\quad i = 1,2,\cdots,m$$

$$\sum_{k=1}^{K_j} x_{jk} = 1,\quad j = 1,2,\cdots,n$$

$$x_{jk} = 0\ or\ 1,\quad j = 1,2,\cdots,n,\ \text{and}\ \ k = 1,2,\cdots,K_j$$

where, $c_{jk} \ge 0$, and $a_{ijk} \ge 0$ for all the values of i, j and k.

In the above model, there are K_j mutually exclusive alternatives for each x_{jk}. There are two sets of constraints in this model; the first set of *m* constraints is called the resource constraints, and the second set of *n* constraints is called the multiple-choice constraints. For the purpose of describing algorithms we need to add a slack variable to the left side of each resource constraint. Therefore, we reformulate the problem as follows:

$$Maximize\ z = \sum_{j=1}^{n}\sum_{k=1}^{K_j} c_{jk}x_{jk},$$

$$subject\ to:$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K_j} a_{ijk}x_{jk} + s_i = B_i,\quad i = 1,2,\cdots,m$$

$$\sum_{k=1}^{K_j} x_{jk} = 1,\quad j = 1,2,\cdots,n$$

$$x_{jk} = 0\ or\ 1,\quad j = 1,2,\cdots,n,\ s_i > 0,\ \text{and}\ \ k = 1,2,\cdots,K_j$$

A basic application of the multiple-choice knapsack models is the project selection problem in which x_{jk} represents the decision variable taking the value of one if the k^{th} alternative of project j is selected, and is zero otherwise. In this mathematical model c_{jk} is the return of selecting the k^{th} alternative of project j, a_{ijk} is the consumption of the k^{th} alternative of project j of resource *i*, and B_i is the maximum availability of resource *i*. The first alternative of each project is do-nothing alternative, and therefore $c_{j1} = 0$ and it's associated $a_{ij1} = 0$, for all values of *i*'s and *j*'s.

For the purpose of describing the developed algorithm, we consider variable X_j to represent a vector with the elements x_{jk} 's, that is $X_j = (x_{j1}, x_{j2}, \cdots, x_{jK_j})$, and from now on we call it "major variable", for all the values of *j*'s, and we call its associated multiple-choice alternatives, i.e. x_{jk}'s the "alternative" of the variable X_j, for all the values of *k*. Let us define the vector $C_j = (c_{1j}, c_{2j}, \cdots, c_{K_j j})$ which is denoted as returns and the vector $A_{ij} = (a_{ij1}, a_{ij2}, \cdots, a_{ijk_j})$ which is denoted as the consumption of the resources. Using these notations, the mathematical model can be converted to the following form:

$$Max\ z = \sum_{j=1}^{n} Cj \circ X_j^T$$

Subject To:

$$\sum_{j=1}^{n} A_{ij} \circ X_j^T \qquad \forall i = 1,2,\cdots,m$$

$$X_j = (1,2,\cdots K_j) \qquad \forall j = 1,2,\cdots,n)$$

By this conversion all the multiple-choice conditions are incorporated inherently in definition of the major variable X_j and therefore all the multiple-choice constraints are discarded. This is the main benefit of the converted mathematical model.

4. DEVELOPMENT OF THE DBB ALGORITHMS

In this research, a depth-first B&B algorithm is proposed. Three different branching mechanisms for selecting the node with the most appropriate decision variable are designed and incorporated in the proposed algorithm for developing three different B&B algorithms. The proposed algorithms are named discriminatory B&B of DBB1, DBB2, and DBB3. The core structure of three algorithms is identical differing by the type of the branching mechanism. All three algorithms are based on the dual approach, starting with a dual feasible solution but supper optimal with respect to the main problem. The branching is incorporated in each algorithm for intending faster fathoming of the branches, and providing a remarkable reduction in explicit enumeration of the number of branches. Through the proposed discriminatory branching, the majority of the solution space is enumerated implicitly. Hence, in the final analysis only a small portion of the solution space needs to be enumerated explicitly in order to obtain the optimal solution.

First, we present the core structure of DBBs algorithms which is identical in all three variants of the algorithm. For simplifying the further discussion, we consider a slight modification of the above mathematical model in which c_{jk}'s are lexicographically ordered. In terms of the modified multiple-choice knapsack problem, the core structure is implemented as follows.

Initially, assume that among all the alternatives of each major variable, one with the largest value of c_{jk} is selected (i.e. $x_{jK_j} = 1 \forall j = 1,2,\ldots,n$). By implementing this assumption for all the major variables, the algorithms start with a solution having the maximum possible objective function value. If the initial solution is a feasible solution (i.e., all slack variables s_i are non-negative), it is the optimal solution and no further investigation is needed. However, this is a trivial case that unlikely happens. Therefore, it will be necessary to select one of the other alternatives of each major variable to force the solution towards feasibility, that is having $s_i \geq 0$ for all *i*. The procedures call for the selection of one other alternative of a major variable at a time, provided there is evidence that this step will be moving the solution towards feasibility. The DBBs algorithms can be represented for data manipulation in a very simple manner. To accomplish this, the following definitions are needed:

Definition 4.1. Free variables: At any node of the branching tree, a major variable is called free if it is not fixed by any branch leading to this node. An alternative of a free variable with the largest value of c_{jk} *is automatically considered to be selected and its associated variable is assigned the value of one. Therefore, at the final solution the best alternative of each major variable which does not appear in the partial solution is automatically selected. The set of the free variables at node t is designated by* F^t.

Definition 4.2. Partial solution: A partial solution provides a specific binary assignment for one of the alternatives of each major variable in the sense that it fixes the value of an alternative of each major variable at zero or one. A convenient way to summarize the information for the purpose of the algorithm is to express the partial solution as an ordered set. Let PS^t *represents the partial solution at the* t^{th} *node and let the notation* $X_j(+k)$ *represent* $x_{jk} = 1,$ *and notation* $X_j(-k)$ *represent* $x_{jk} = 0.$ *The set* PS^t *must be ordered in the sense that each new element is always assigned on the right most of the elements, ie. it appears as the last element in the partial solution set.*

The partial solution is used to define the nodes in the B&B tree. The use of partial solutions eliminates the need for recording generated nodes of the B&B trees. This means that a partial solution (nodes) can be generated successively from a previous partial solution. As an example, let us consider a partial solution at the k^{th} iteration, determined as $PS^k = \{X_i(j), X_l(m)\}.$ If any of the algorithm selects alternative *n* from the major variable X_s, then the next partial solution at *k+1* iteration would be determined as $PS^{k+1} = \{= \{X_i(j), X_l(m), X_s(n)\}.$ The detailed procedures for generating successive partial solutions are the key element of the DBB algorithms which differentiate the three algorithms. We first describe the main procedure, core DBB algorithm, and later we will describe discriminatory mechanism of each algorithms, DBB1, DBB2, and DBB3.

4.1 The Core DBB Algorithm

To describe the general procedure for generating the partial solution, the algorithm starts with an empty partial solution $PS^0 = \phi,$ which means all the major variables are free (with their associated variables with the largest value of c_{jk} that are considered to be one), and therefore the value of the objective function is calculated as $z_0 = \sum_{j=1}^{n} c_{jK_j}$. If this solution is a feasible solution the procedure is stopped, otherwise the next partial solution is generated. The next partial solution is generated by assigning the decision variable corresponding to one of the other alternatives of a major variable to the right of the last element of the partial solution, with the value of one. It is an essential matter to note that at any node; the best alternative of all major variables is presumably selected unless an alternative of a major variable is appeared in the partial solution with a positive sign, and hence there would be no need for further branching of the best alternatives for any major variables. The procedure for generating the partial solutions will continue until a branch is fathomed. The partial solution is said to be fathomed if either it cannot be lead to a better objective function value, or a feasible solution is reached. A fathomed partial solution means that it is not promising to further branch its associated node, since all the solutions that could be generated from the node are either inferior or infeasible. After a branch is fathomed the backtracking starts. We incorporated a mechanism to ensure that all the possible solutions are enumerated either implicitly or explicitly. The general rule for generating the next partial solution after a node is fathomed is as follows. If all the elements of the fathomed partial solution are negative, i.e. all elements of the partial solution represent $X_j(-k)$ for all values $k \in \{1,2,\cdots,K_j\}$ and all $X_j \in PS^t$ or if the set of the free variables are empty, the enumeration is completed and the optimal solution is obtained. Otherwise, the algorithm selects the right most positive element, complemented with the selection of another alternative and then it deletes all the negative elements to its right. This elimination is a key point for granting the optimality since by which we ensure that all solution points have been enumerated. After all the negative elements adjacent to the right most positive element is deleted or if all the corresponding alternatives of the right most major variable have already been in the partial solution we delete this major variable, otherwise this major variable is kept in the partial solution until all its alternatives are fathomed. We can see the significance of the negative element here. Since, the algorithm always adds variables at level one, a negative element means that a preceding partial solution has been fathomed. Thus when all the elements of a partial solution are negative, the associated variables have been considered and as a result, there are no more branches to consider and the enumeration is completed. In order to increase the efficiency of the proposed algorithm we incorporate some tests for reducing the state space solution, using the following propositions.

Proposition 4.1.1. Let x_{jk} *for* $k = 1,2,\cdots,K-1,$ *be the all other alternatives of the major variable* X_j *(except* x_{jK} *which is selected by defaults). The current solution dominates all the alternatives having less profit and consuming more resources.*

Proof: Let the current objective function value be $z^t = z^{t-1} + c_{lk}$ *selecting any decision variable* x_{jk} *for* $j = 1,2,\cdots K-1, \& j \neq l,$ *we have* $z^{t+1} = z^t - c_{lk} + c_{jk}.$ *since* $c_{lk} \leq c_{jk} \Rightarrow z^{t+1} \leq z^t.$ *and since* $a_{jk} \geq a_{lk} \forall k = 1,2,\cdots K \ \& \ j \neq l,$ *the solution won't lead toward feasibility and therefore the current solution dominates these alternatives.*

Based on the above discussion, we can summarize the steps of the DBB algorithms, which are identically employed in DBB1, DBB2 and DBB3 algorithms. For the simplicity of describing the algorithms, assume that for each major variable, the alternatives are sorted according to the non-decreasing order of their objective function coefficients. The steps of the DBB algorithms are as follows:

4.1.1 Core DBB algorithm

Step 1. Let $t = 0,$

$F^0 = \{X_1(1,2,\ldots K_1 - 1), X_2(1,2,\ldots K_2 - 1),$

$\ldots, X_n(1,2,\ldots K_n - 1)\}$

, (the best alternative (K_j) of all major

variables are implicitly selected and are not considering for further selection at any iteration), $PS^0 = \phi$, $z^0 = \sum_{j=1}^{n} c_{jK_j}$, let the initial lower bound denoted by $\underline{z}$ be zero, $\underline{z} = 0,$ and determine all s_i values.

Step 2. If this solution is a feasible solution, stop.

Step 3. Let $t = t + 1,$ if F^t is empty stop; otherwise select another alternative of a free major variable by discriminatory procedure and remove it from F^t and assigned on the right of the last element of PS^t. Then determine the value of z^t. If $z^t \le \underline{z},$ go to Step 5, otherwise go to Step 4.

Step 4. Calculate the values of s_i's,. If all s_i's are nonnegative, let $\underline{z} = z^t$, and go to Step 5, otherwise go to Step 3.

Step 5. Change the sign of the last element of the partial solution to a negative sign, and go to Step 5.1.

Step 5.1. If its all associated alternatives have the negative sign, delete this major variable, and go to Step 3, otherwise go to Step 5.2.

Step 5.2. Select another alternative of the right most major variable by the discriminatory procedure, and perform the dominancy test. If it is not dominated by the existing partial solution alternative, remove it from F^t and assigned on the right of the last element of PS^t, otherwise select the other alternative of this major variable by discriminatory procedure which is not dominated by the existing alternative. Then let $t = t + 1$ and determine the values of z^t. If $z^t \le \underline{z},$ go to Step 5, otherwise go Step 4.

In the following sections, we describe the discriminatory procedures of DBB1, DBB2 and DBB3.

4.2 DBB1 Algorithm

In this algorithm, the selection of the next partial solution is performed based on a concept similar to the steepest accent gradient. The basic idea is to select a decision variable which consumes less of the average resources and provide a better objective function value. Due to the multiple resource constraints, we cannot simply take the ratio of the objective function coefficient by the constraint coefficient of each variable and select the candidate variable with the largest ratio. However, using the same concept a matrix (*G*), is defined for selecting the next decision variable. The elements of this matrix are determined as:

$$G = \left\| g_{kj} \right\|_{K_j \times n}, \ where \ g_{kj} = \frac{c_{jk}}{\sum_{i=1}^{m} \frac{a_{ijk}}{A_i}} \qquad (1)$$

Using this matrix, we initially flag the element of the best alternative for each major variable. That is, these alternatives are selected in the initial step of the algorithm. Now if the algorithm calls for substituting an alternative of any major variable (Step 3), an element of *G* from the un-flagged elements which has the maximum value is selected. If on the other hand, algorithm calls for substituting an alternative for a given major variable (Step 5.2), we select an element of *G* from its corresponding column excluding the alternatives which are already flagged. Then the selected element is flagged. Therefore, the value of the available resources is decreased by the amount of the consumptions of this variable and is increased by the amount of the consumption of the substituted alternative. For normalizing the consumption of the resources, it is important to perform the adjustment of the resources. Therefore, after selection of a decision variable the algorithm updates all the elements of the gradient matrix. We assembled an algorithmic procedure called DBB1 algorithm by incorporating this discriminatory procedure.

4.3 DBB2 Algorithm

Consider node *t* with an infeasible solution. That is; the left-hand side value of the resource constraint *i* is A_i^t , where $A_i^t \ge A_i$ for at least one of the *i*'s. The selection of the next major variable and its associated alternative is the one with the steepest decent projected vector for moving the solution towards feasibility more rapidly, i.e. forcing $A_i^{t+1} \le A_i$ for all the values of *i* in the minimum number of iterations.

Definition 4.3.1. The steepest decent projection vector: At any iteration if $A_1, A_2, \cdots, A_m$, *represent the amount of the right hand side of the resource constraints 1,2,…,m, denoting by the vector V₁ and* $A_1^t, A_2^t, \cdots, A_m^t$, *represent the amount of the left hand side of the resource constraints 1,2,…,m at iteration t, (which is an infeasible solution,* $A_i^t \geq A_i$ *for at least one of the i's) denoting by vector V₂. Letting*

$$V_3 = V_1 - V_2 = (A_1 - A_1^t, A_2 - A_2^t, \cdots, A_m - A_m^t), \quad (2)$$

as the amount of the left hand side of the resource constraints 1,2,…,m, at iteration t+1, denoting by V₄ then, by defining V_5 *as:*

$$V_5 = V_4 - V_2 = (A_1^{t+1} - A_1^t, A_2^{t+1} - A_2^t, \cdots, A_m^{t+1} - A_m^t), \quad (3)$$

then the projection vector of V₅ on V₃ is determined as

$$\left|V_6^+\right| = \left|V_5\right| * Cos\,\theta = \left|V_5\right| * \frac{V_3 \circ V_5}{\left|V_3\right| * \left|V_5\right|} =$$

$$\left|V_5\right| * \frac{(A_1 - A_1^t)(A_1^{t+1} - A_1^t) + (A_2 - A_2^t)(A_2^{t+1} - A_2^t) + \cdots + (A_m - A_m^t)(A_m^{t+1} - A_m^t)}{\left|V_3\right| * \left|V_5\right|} =$$

$$\frac{(A_1 - A_1^t)(A_1^{t+1} - A_1^t) + (A_2 - A_2^t)(A_2^{t+1} - A_2^t) + \cdots + (A_m - A_m^t)(A_m^{t+1} - A_m^t)}{[(A_1 - A_1^t)^2 + (A_2 - A_2^t)^2 + \cdots + (A_m - A_m^t)^2]^{1/2}} \quad (4)$$

Considering a problem with two resource constraints, by letting the vertical and the horizontal axes representing the first and second resource consumptions, the vectors are presented in Fig. (1).

Proposition 4.3.1. For selecting the next alternative of the major variable X_j *(i.e.* x_{jk}*), the alternative with the largest vector projected which is connecting the point in a m Euclidian space representing the resources consumption of the current solution to the point representing the available resources provides a solution with lowest infeasibility.*

Proof: Let V_6^i *and* V_6^k *denote the projected vector of the alternative i and k of the major variable* X_j *respectively. Let,* $d_6^i = V_3 - V_6^i$ *and* $d_6^k = V_3 - V_6^k$ *denote the amount of infeasibility of selecting alternative and k an alternative of the major variable* X_j. *If* $V_6^k \geq V_6^i$ *then* $d_6^k \leq d_6^i$, *if one select the alternative k. it provides a solution more forward to the feasibility.*

Using this formula (4), depending on what the algorithm is called, it either calculates the steepest decent projected vector, for all the unselected alternatives of all free variables (Step 3). or all unselected alternatives of a given major variable (Step 5.2), and /or chooses the vector with the maximum length. This vector corresponds to a decision variable which will be added to the partial solution. In DBB2 procedure, the updating routine for calculating the length of the steepest decent projected vector is incorporated. By integrating the DBB2 discriminatory procedure using the core structure of the DBB, the DBB2 algorithm is assembled.

4.4 DBB3 algorithm

The selection mechanism employed for developing DBB1is considered as a mechanism which is based on the optimality concept. On the other hand, DBB2 utilizes the concept of feasibility as the selection mechanism. In DBB both optimality and feasibility concepts are concurrently employed. Considering that alternatives of each major variable are sorted according to the non-decreasing order, we have *n* project and *K* alternatives (or more generally K_j alternatives for project *j*), we can construct a matrix called *D*, having *n* columns and C_n^2 rows, as follows:

$$D = \left\|d_{ij}\right\| = \begin{bmatrix} c_{12} - c_{11} & \cdots & c_{j2} - c_{j1} & \cdots & c_{n2} - c_{n1} \\ c_{13} - c_{11} & \cdots & c_{j3} - c_{j1} & \cdots & c_{n3} - c_{n1} \\ c_{14} - c_{11} & \cdots & c_{j4} - c_{j1} & \cdots & c_{n4} - c_{n1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{1K} - c_{11} & \cdots & c_{jK} - c_{j1} & \cdots & c_{nK} - c_{n1} \\ c_{13} - c_{12} & \cdots & c_{j3} - c_{j2} & \cdots & c_{n3} - c_{n2} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{1K} - c_{12} & \cdots & c_{jK} - c_{j2} & \cdots & c_{nK} - c_{n2} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{1K} - c_{1,K-1} & \cdots & c_{jK} - c_{j,K-1} & \cdots & c_{nK} - c_{n,K-1} \end{bmatrix}$$

In this matrix columns represent the major variables and rows represent the rate of change of the objective function values. The following proposition reveals the appropriateness of the selection mechanism of the DBB3 algorithm.

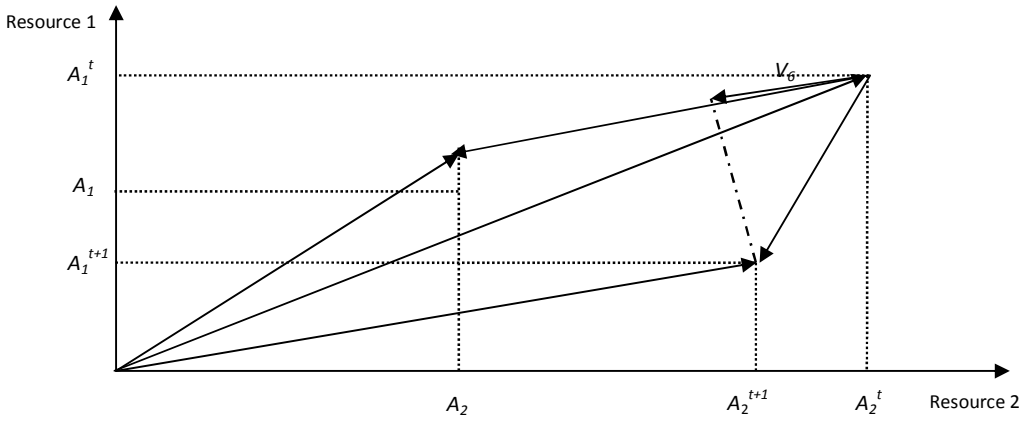

Fig. 1. Graphical illustration of the projected vector

Proposition 4.4.1. For selecting the next alternative of the major variable X_j *(i.e.* x_{jk}*), with smallest value* $d_{jk} / \left|V^{+6}\right|$ *constitutes a solution with sharpest towards feasibility and with less amount of penalizing the objective function.*

Proof: Let z^t *be the value of objective function at iteration t. By proposition 4.3.1 we showed the selection of a variable with the largest value of the projected vector,* V_6*, moves the current solution toward feasibility be the largest amount. Now let* z^{t+1} *be the value of the objective function at iteration t+1, by selecting the alternative of k of the major variable* X_j*, through the above-mentioned selection mechanism, i.e.* $c_{jl} - c_{jk} \le c_{ji} - c \ \forall$ *all values of i's. The objective function value becomes* $z^{t+1} = z^t - c_{jl} + c_{jk} \ge z^t - c_{ji} + c_{jk}$*, therefore selecting the kth alternative have a better objective function value.*

Proposition 4.4.2. For selecting the best alternative a free variable the smallest value $d_{lk} / \left|V^{+6}\right|$ *constitutes a solution with sharpest towards feasibility and with less amount of penalizing the* objective *function.*

Proof: Let z^t *be the value of objective function at iteration t. By proposition 4.3.1 we showed the selection of an alternative of a free variable with the largest value of the projected vector,* V_6*, moves the current solution toward feasibility be the largest amount. Now let* z^{t+1} *be the value of the objective function at iteration t+1, by selecting the best alternative of the major variable* X_l*, through the above-mentioned selection mechanism, i.e.* $c_{lk} - c_{li} \le c_{lj} - c_{lk} \ \forall$ *all values of i's. The objective function value becomes* $z^{t+1} = z^t - c_{li} + c_{jk} \ge z^t - c_{ji} + c_{jk}$*, therefore selecting the k[th] alternative have a better objective function value.*

Since, in the backtracking procedure of the DBB algorithms we have substituted an *n* alternative of a major variable with another alternative of the same major variable, for pushing the solution towards feasibility, the value of the objective function is decreased by the difference of the objective function coefficients of these two alternatives. It is to be noted that the row's elements of this matrix indicates the rate of change of the objective function value when an alternative is substituted by another alternative. Moreover, the fact that that length of vector V_{+6} indicates the rate of change in the right hand sides towards feasibility when an alternative is substituted by another alternative, the ratio of d_{ij} to its corresponding V_{+6} , i.e. $d_{ij} / \left|V^{+6}\right|$ measures the amount of penalizing the objective function value and rewarding the feasibility of the solution. The smaller value of this ratio indicates the lower penalty we pay and larger reward we

acquire. Therefore, depending on what the algorithm is called, it may either selects an unselected alternative from the set free variables (Step 3) or an unselected alternative of a given major variable (Step 5.3). In the first case, it calculates this ratio for all the free variables, and in the second case it only needs to calculate this ratio for all the elements of one column of matrix *D*. In either case it selects the variable and its alternative corresponding to minimum ratio, for the further branching. DBB3 algorithm is constructed by integrating the DBB3 discriminatory procedure with the core structure of the DBB algorithm.

5. COMPUTATIONAL EXPERIMENTS

In this section, we report the computational experiments of three variants of the DBB algorithm. The three variants of the DBB algorithm are implemented in C^{++} on a personal computer. Performance measures for three algorithms include the average number of nodes, which are enumerated, and the average computational time in CPU seconds. As a bench mark, we also determined the same performance measures by a traditional B&B algorithm. We refer to a traditional B&B algorithm as an algorithm in which the nodes for further branching are selected based on best dept mechanism. We then compared the DBB's performance measures with the traditional B&B to evaluate the computational efficiency of each algorithm.

We employed the pseudo random number generation for constructing the test problems. For each subgroup of test problems with the size of *n* projects, each having K_j alternatives, with *m* resource constraints, we generated 10 test problems and calculated the average of the performance measures for each subgroup. The parameters of the problems were generated using the uniform density function. Specifically, the values of $c_{jk}'s, \text{and } a_{ijk}'s$ were generated using the following functions:

$$c_{jk} = 1 + \text{integer}[rand() * 19], \text{and } a_{ijk} = 0.1 + rand() * 0.20.$$

The value of the right hand side of each resource constraint is calculated by the following function:

$$A_i = (\sum_{j=1}^{n} \max_k \{a_{ijk}\}) * CTF$$

Where *CTF* (Constraints Toughness Factor) is defined to implement the scarceness of the available resources. We let the value of *CTF* varies from 0.50 to 0.80.

We employed DBB1, DBB2 and DBB3, and the traditional B&B algorithms for solving the generated test problems in each subgroup with different *CTF* values. Then we calculated the average number of the nodes which were enumerated by each algorithm.

Table (1) reports the computation results of the three algorithms for solving generated problems in the subgroup of 5 projects, each with 4 alternatives and with 3 resource constraints for different values of the *CTF*. Similarly, Tables 2 through 4 reports the similar results for problems in the subgroups of 10, 15, and 20 projects, respectively. Table (5) reports the relative performance of the three algorithms comparing with the traditional B&B. For each subgroup with a given value of the *CTF*, we took the ratio of the number of nodes enumerated by each algorithm to the number of nodes enumerated by traditional B&B.

Figs. 1 through 4 illustrate the effect of different *CTF* values on the number of nodes enumerated by each of the DBB algorithm, for problems in the subgroups of 10, 15, and 20 projects, respectively. Figure 5 demonstrates the performance of the developed algorithms comparing with the traditional B&B with respect to the variation of the number of decision variables. The performance of each of the developed algorithm is measured by the formulae $P = 1 - \frac{NON_D}{NON_c}$ in which *P* is the performance of the developed algorithm, NON_D is the average number of nodes enumerated by individual developed algorithm, and NON_C is the average number of the nodes enumerated by the traditional B&B, averaged over the number of nodes enumerated using different *CTF* values. Fig. 6 illustrates the computational efficiency of the developed algorithm compared to the traditional B&B with respect to the variation of the number of decision variables. The computational efficiency of each of the developed algorithm is simply measured by formulae $E = \frac{NON_D}{NON_c} \times 100.$

As it is seen the computational efficiency of DBB increases as the number of decision variables increases.

Table 1. Performance measures for the problem size of $n = 5, K_j = 4, m = 3$

CTF	Classical B & B		DBB1 Algorithm		DBB2 Algorithm		DBB3 Algorithm	
	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.
0.80	10	0.00	11	0.00	16	0.00	12	0.00
0.75	41	0.00	33	0.00	40	0.00	36	0.00
0.70	114	0.00	62	0.00	64	0.00	70	0.00
0.65	139	0.00	65	0.00	72	0.00	68	0.00
0.60	229	0.00	83	0.00	72	0.00	92	0.00
0.55	434	0.00	90	0.00	91	0.00	115	0.00
0.50	586	0.00	111	0.00	110	0.00	116	0.00
Average	**222**	**0.00**	**65**	**0.00**	**66**	**0.00**	**73**	**0.00**

Table 2. Performance measures for the problem size of $n = 10, K_j = 4, m = 3.$

CTF	Classical B & B		DBB1 Algorithm		DBB2 Algorithm		DBB3 Algorithm	
	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.
0.80	1	0.00	1	0.00	1	0.00	1	0.00
0.75	224	0.00	72	0.00	51	0.00	107	0.00
0.70	923	0.00	223	0.00	174	0.00	370	0.01
0.65	4,401	0.01	697	0.01	391	0.01	955	0.03
0.60	11,986	0.02	1,649	0.02	964	0.02	1,959	0.07
0.55	36,909	0.08	3,584	0.04	2,583	0.06	3,713	0.14
0.50	119,989	0.28	8,688	0.10	6,541	0.18	8,413	0.31
Average	**24,919**	**0.06**	**2,131**	**0.02**	**1,529**	**0.04**	**2,217**	**0.08**

Table (3) Performance measures for the problem size of $n = 15, K_j = 4, m = 3.$

CTF	Classical B & B		DBB1 Algorithm		DBB2 Algorithm		DBB3 Algorithm	
	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.
0.80	1	0.00	1	0.00	1	0.00	1	0.00
0.75	220	0.00	209	0.01	185	0.00	115	0.00
0.70	4,676	0.01	1,868	0.03	1,108	0.03	1,339	0.06
0.65	61,863	0.21	9,966	0.19	5,016	0.16	7,986	0.39
0.60	509,680	1.58	65,330	0.97	23,473	0.78	35,814	1.80
0.55	3,536,919	10.13	297,939	4.61	107,999	3.84	136,237	7.49
0.50	22,170,047	69.52	910,408	15.61	354,545	13.94	423,850	15.24
Average	**3,754,773**	**11.64**	**183,674**	**3.06**	**70,332**	**2.68**	**86,477**	**3.57**

Table 4. Performance measures for the problem size of $n = 20, K_j = 4, m = 3.$

CTF	Classical B & B		DBB1 Algorithm		DBB2 Algorithm		DBB3 Algorithm	
	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.	Average no. of nodes	Average CPU sec.
0.80	1	0.00	1	0.00	1	0.00	1	0.00
0.75	1	0.00	1	0.00	1	0.00	1	0.00
0.70	140	0.00	140	0.00	152	0.01	105	0.01
0.65	11,092	0.05	2,064	0.04	1,627	0.06	4,204	0.12
0.60	440,698	2.79	33,290	0.63	11,853	0.56	61,148	1.29
0.55	9,873,763	36.70	504,597	7.59	150,001	2.53	466,697	10.53
0.50	215,490,347	808.80	5,529,618	89.92	1,353,250	62.15	3,580,100	107.84
Average	**32,259,434**	**121.05**	**867,102**	**14.02**	**216,698**	**9.33**	**587,465**	**17.11**

Table 5. Ratio of the nodes enumerated by DBBs with respect to the classical B & B

CTF	*N =5, K_j = 4, m=3*			*N =10, K_j = 4, m=3*			*N =15, K_j = 4, m=3*			*N =20, K_j = 4, m=3*		
	DBB1	**DBB2**	**DBB3**	**DBB1**	**DBB2**	**DBB3**	**DBB1**	**DBB2**	**DBB3**	**DBB1**	**DBB2**	**DBB3**
0.80	112.24	159.18	120.41	100.00	100	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.75	79.51	96.59	87.32	32.29	22.57	47.73	95.10	83.83	52.13	100.00	100.00	100.00
0.70	54.48	56.06	61.16	24.15	18.88	40.15	39.94	23.69	28.64	100.00	108.56	74.61
0.65	46.34	51.36	49.07	15.84	8.89	21.71	16.11	8.11	12.91	18.60	14.67	37.90
0.60	36.16	31.53	40.26	13.76	8.04	16.34	12.82	4.61	7.03	7.55	2.69	13.88
0.55	20.84	21.07	26.51	9.71	7.00	10.06	8.42	3.05	3.85	5.11	1.52	4.73
0.50	18.98	18.78	19.87	7.24	5.45	7.01	4.11	1.60	1.91	2.57	0.63	1.66
Average	29.28	29.90	32.80	8.55	6.14	8.90	4.89	1.87	2.30	2.69	0.67	1.82

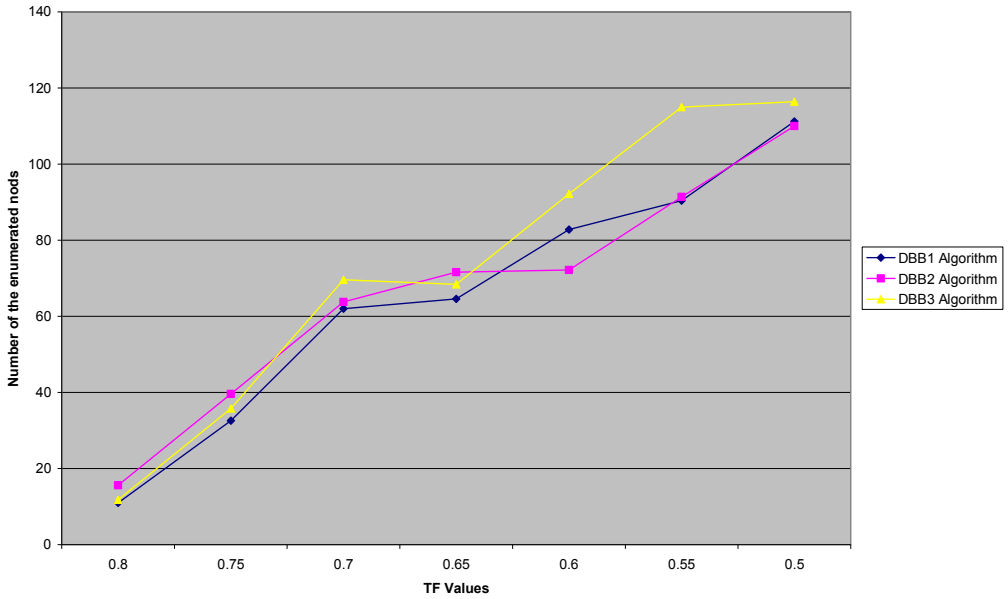

Fig. 1. Effect of *CTF* on the number of nodes for *n*=5, *K*=4, *m*=3 problems

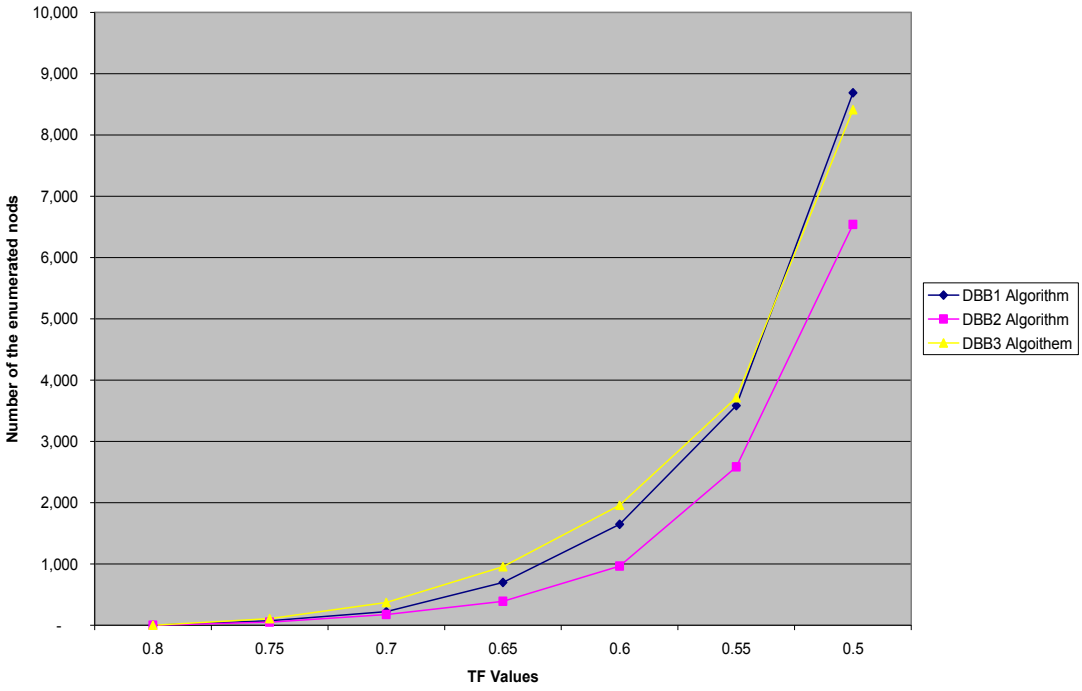

Fig. 2. Effect of *CTF* on the number of nodes for *n*=10, *K*=4, *m*=3 problems

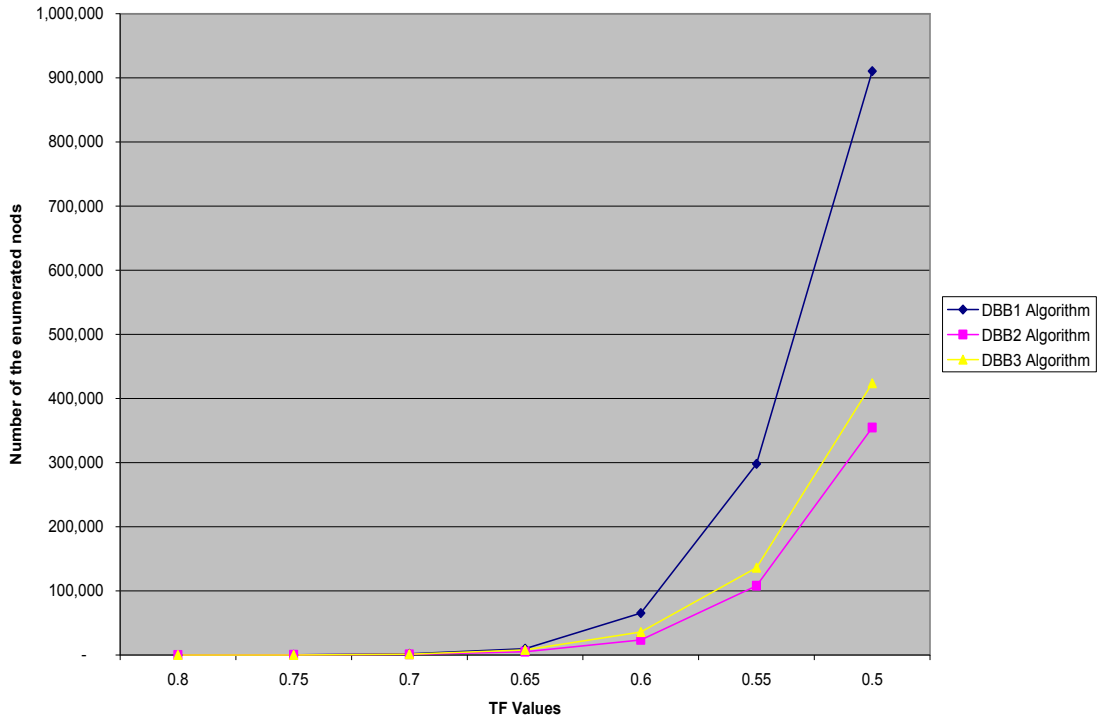

Fig. 3. Effect of *CTF* on the number of nodes for *n*=15, *K*=4, *m*=3 problems

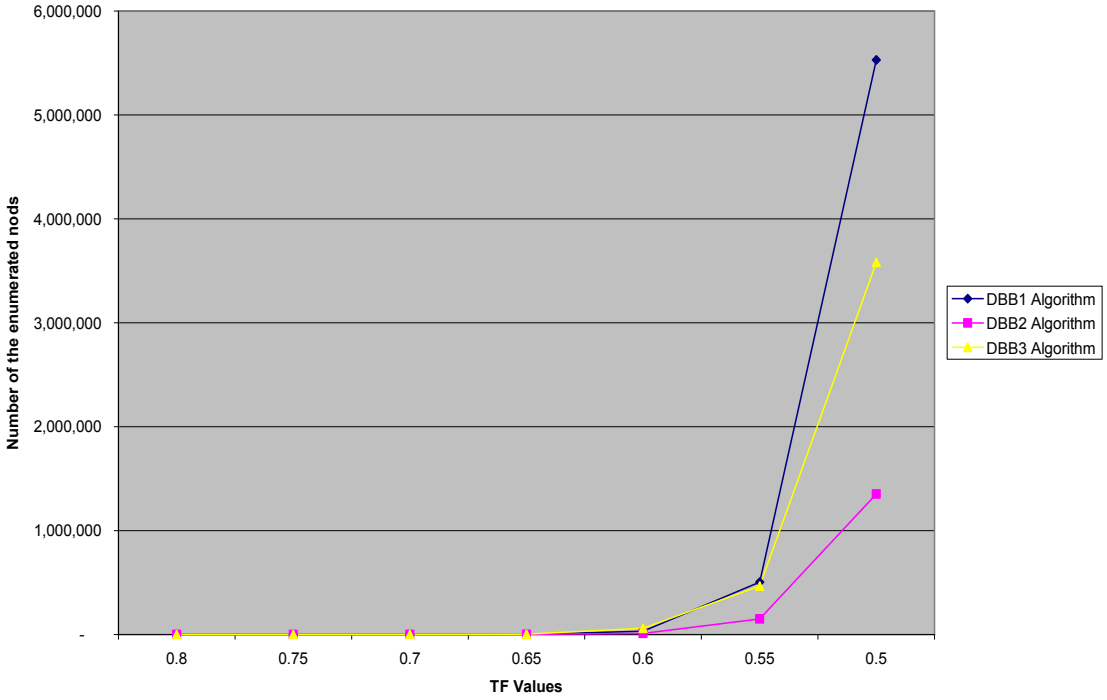

Fig. 4. Effect of *CTF* on the number of nodes for *n*=20, *K*=4, *m*=3 problems

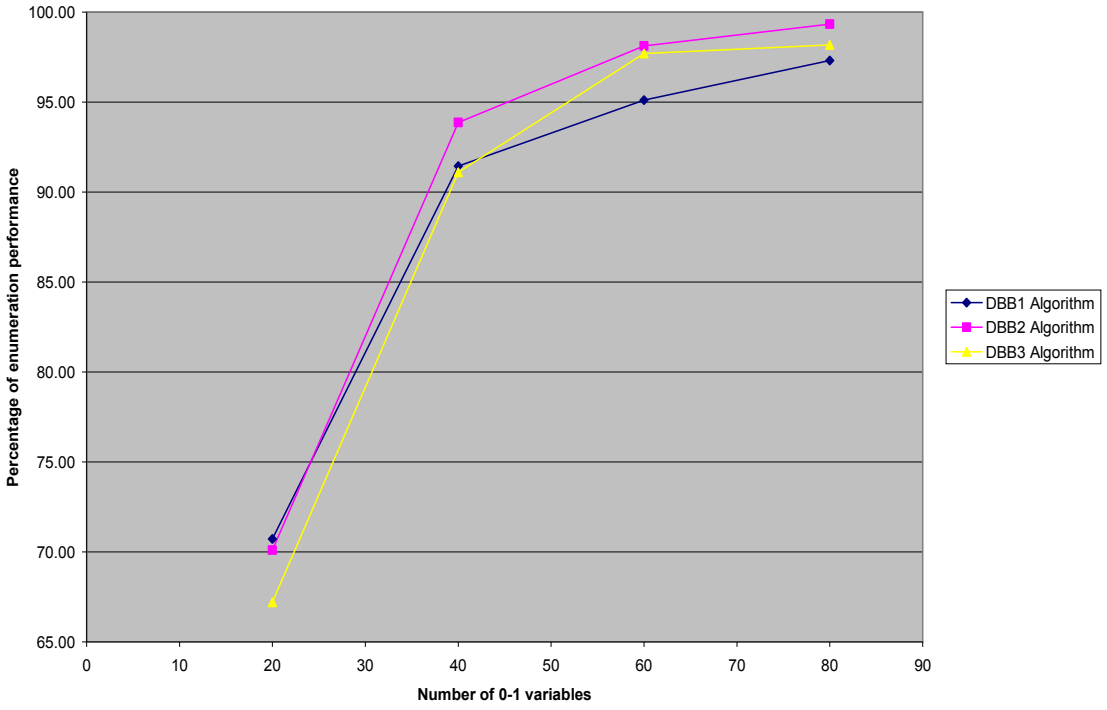

Fig. 5. Performance of the developed algorithms comparing with the traditional B&B

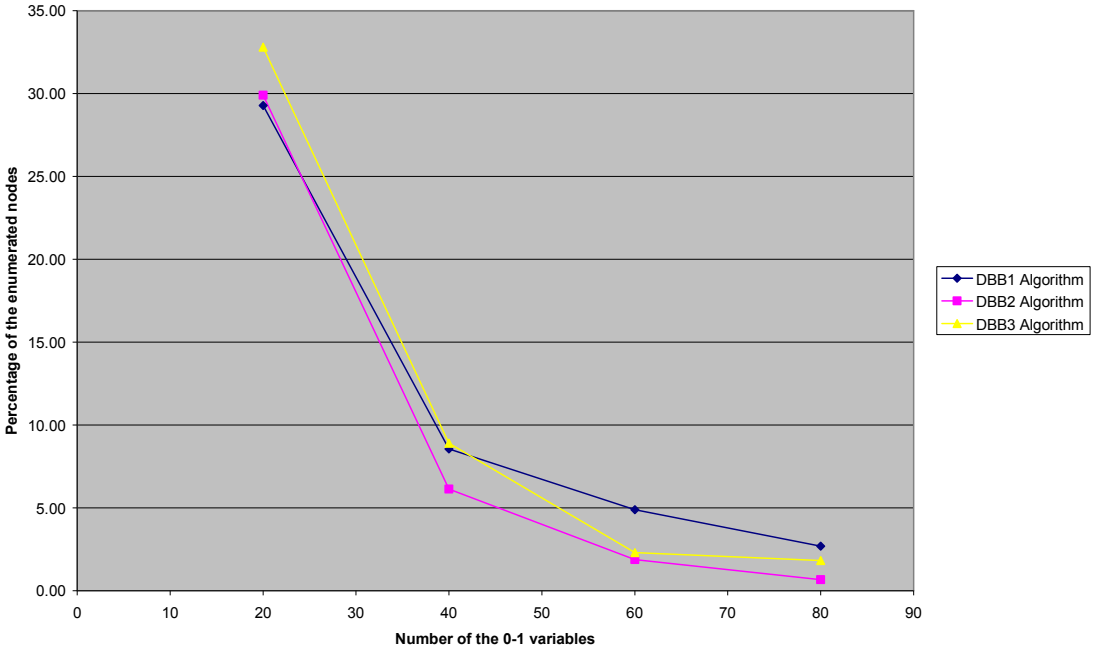

Fig. 6. Percentage of the nodes enumerated by the developed algorithms comparing with the traditional B&B

6. CONCLUSIONS AND REMARKS

In this paper, we presented a first depth dual approach B&B for solving the general form of MMKP. The approach is called the discriminatory B&B due to the selection of a node for further branching based on a discriminatory criterion. Three selecting mechanisms are developed for obtaining the optimal solution of MMKPs. To enhance the computational efficiency of B&B, the discriminatory mechanisms are incorporated in the proposed algorithmic procedure to develop three exact solution algorithms. An extensive computational experiment was performed to evaluate the effects of the discriminatory selection of the variables for reducing the implicit enumerations. The results revealed that all the variants of the variable selection lead to a considerable reduction of the nodes to be enumerated for obtaining the optimal solution of MMKPs. Referring to Table 5 it can be seen that for the problems with the size of 100 zero-one decision variables, the developed algorithms enumerated a small portion of the nodes comparing to the traditional B&B. More specifically, DBB1 enumerated only 2.69 percent, DBB2 enumerated only 0.67 percent and DBB3 enumerated only 1.82 percent of the total nodes enumerated by the traditional B&B procedure.

The main contributions of this paper are two folds. First, a special algorithmic procedure based on dual approach first-depth B&B algorithm is proposed. In this algorithm by setting all the major variables equal to their highest return values the enumeration of this variable is conducted explicitly without any further computational effort. Then by performing a successively systematic procedure, it assigns the lower return value to the selected variables in such a way that after trying a small part of all the possible combinations, one obtains either an optimal solution, or evidence of the fact that no feasible solution exists. Also, a smart mechanism is incorporated for ensuring the total enumeration has been performed for all possible combinations, before the termination step of the algorithm.

The second contribution is development of three selecting mechanisms. The first mechanism is based on the steepest accent gradient by which the variable consuming less of the average resources and provide a better objective function value. The second mechanism is based on the steepest decent projected vector by which it is ensured that the variables are selected based on the most promising objective function value with lowest infeasibility. Finally, by the third mechanism a variable with sharpest vector

towards feasibility and with less amount of penalizing the objective function is selected.

Reduction of the solution space and consequently enumeration of the smaller number of nodes in B&B approach has been a challenging effort in solving knapsack problems. Respectfully, in this paper three mechanisms are proposed. Proposing another type of selection mechanism or combining the proposed mechanisms in a single algorithm can be an interesting research to be further explored.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Kellerer H, Pferschy U, Pisinger D. Introduction to NP-Completeness of Knapsack Problems. Springer, Berlin, Heidelberg; 2004.
2. Chen y, Hao CJ. A “reduce and solve” approach for the multiple-choice multidimensional knapsack problem. European Journal of Operational Research. 2014;239(2):313-322.
3. Lin EYH. A bibliographical survey on some well-known non-standard knapsack problems. INFOR, Information Systems and Operational Research.1998;36(4):274-317.
4. Lancaster LM. The history of the application of mathematical programming to menu planning. European Journal of Operational Research. 1992;57:339-347.
5. Petersen CC. Computational experience with variants of the Balas algorithm applied to the selection of R & D projects. Management Science. 1967;13:736-750.
6. Akpan NP, Etuk EH, Essi ID. A deterministic approach to a capital budgeting problem. American Journal Scientific Industrial Research. 2011;2(3): 456-460.
7. Li Y, Tang X, Cai W. Dynamic bin packing for on-demand cloud resource allocation. IEEE Transactions on Parallel and Distributed Systems. 2016;27(1):1-14.
8. Ibrahim M. A novel approach to fully private and secure auction: a sealed-bid knapsack auction. International Journal of Research and Reviews in Applied Science. 2011;9(2):260-269.
9. Balas E. An additive algorithm for solving linear programs with zero–one variables. Operations Research. 1965;13:517–546.
10. Glover F. A multiphase-dual algorithm for the zero-one integer programming problem. Operation Research. 1965;13:879–919.
11. Greenberg H, Pierskalla W. Surrogate mathematical programs. Operations Research. 1970;18:924-939.
12. Golver F. Surrogate constraints duality in mathematical programming. Operations Research. 1975;23:434-451.
13. Ghassemi Tari F. A multiple-choice knapsack for rehabilitation and maintenance of Texas highway. A Dissertation in Partial Fulfillment of the degree of Doctor of Philosophy, Department of Industrial Engineering, Texas A & M University; 1980.
14. Phillips DT, Shanmugham CV, Ghassemi Tari F, Lytton RL. Rehabilitation and maintenance system state optimal fund allocation. Research Report 239-2, Texas Transportation Institute, Texas A & M University, USA; 1981.
15. Boyer V, Baz DE, Elkihed M. Solution of multidimensional knapsack problems via cooperation of dynamic programming and branch and bound. European J. of Industrial Engineering. 2010;4(4):434-449.
16. Ghassemi Tari F. A hybrid dynamic programming for solving fixed cost transportation with discounted mechanism. Journal of Optimization. 2016;9.
17. Freville A. The multidimensional 0–1 knapsack problem: An overview. European Journal of Operations Research. 2004; 155(1):1-21.
18. Shih W. A branch and bound method for the multiconstraint zero-one knapsack problem. Journal of the Operational Research Society. 1979;30:369–378.
19. Gavish B, Pirkul H. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Mathematical Programming. 1985;31:78–105.
20. Jahangiri E, Ghassemi Tari F. A dynamic programming approach for solving nonlinear knapsack problems. Journal of Industrial Engineering International. 2006; 2(1):31-37.
21. Ghassemi Tari F, Jahangiri E. Development of a hybrid dynamic

programming approach for solving discrete nonlinear knapsack problems. Applied Mathematics and Computation. 2007; 188(1):1023-1030.

22. Dyer ME, Riha WO, Walker J. A hybrid dynamic programming branch-and-bound algorithm for the multiple-choice knapsack problem. Journal of Computational and Applied Mathematics. 1995;58(1):43-54.
23. Bao JH, Yang QG. Fast solution algorithm of multiple-choice knapsack problem. Journal of South China University of Technology. 2009;37(4):42-45.
24. Han B, Leblet J, Simon G. Hard multidimensional multiple choice knapsack problems, an empirical study. Computers & Operations Research. 2010;37(1):172-181.
25. Fomeni FD, Letchford AN. A dynamic programming heuristic for the quadratic knapsack problem. INFORMS Journal on Computing. 2014;26(1):173-182.
26. Chebil K, Khemakhem M. A dynamic programming algorithm for the Knapsack Problem with Setup. Computers and Operations Research. 2015;64(C):40-50.
27. He YC, Wang XZ, He YL, Zhao SL, Li WB. Exact and approximate algorithms for discounted {0-1} knapsack problem. Information Sciences. 2016;369:634–647.
28. Della Croce F, Pferschy U, Scatamacchia R. Dynamic programming algorithms, efficient solution of the LP-relaxation and approximation schemes for the penalized knapsack problem, Tech. Report 2017-03-5880, Optimization Online; 2017.
29. Pferschy U, Scatamacchia R. Improved dynamic programming and approximation results for the knapsack problem with setups. International Transactions in Operational Research. 2018;25(2):667–682.
30. Vercellis C. Constrained multi-project planning's problems: A Lagrangean decomposition approach. European Journal of Operational Research.1994; 78(2):267-275.
31. Pisinger D. Budgeting with bounded multiple-choice constraints. European Journal of Operational Research. 1995; 129(3):471-480.
32. Pisinger D. A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operational Research. 2002;83(2):394-410.
33. Cherfi N, Hifi M. Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem. Int. J. Operational Research. 2009;5(1):89-109.
34. Cherfi N, Hifi M. A column generation method for the multiple-choice multi-dimensional knapsack problem. Comput. Optim. Appl. 2010;46(1):51–73.
35. Lin EYH, Bricker DL. Computational comparison on the partitioning strategies in multiple choice integer programming. European Journal of Operational Research. 1996;88(1):182-202.
36. Sung CS, Cho YK. Reliability optimization of a series system with multiple-choice and budget constraints. European Journal of Operational Research. 2000;127(1):159-171.
37. Kozanidis G, Melachrinoudis E. A branch & bound algorithm for the 0-1 mixed integer knapsack problem with linear multiple choice constraints. Computers & Operations Research. 2004;31(5):695-711.
38. Ohtagaki H, Iwasaki A, Nakagawa Y, Narihisa H. Smart greedy procedure for solving a multidimensional nonlinear knapsack class of reliability optimization problems. Mathematical and Computer Modeling. 2000;31:283-288.
39. Hifi M, Sadfi S, Sbihi A. An exact algorithm for the multiple-choice multidimensional knapsack problem. Cahiers de la Maison des Sciences Economiques. b04024, Université Panthéon-Sorbonne (Paris 1). 2004;1-16.
40. Sbihi A. A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. J. Combinatorial Optimization. 2007;(4):337–351.
41. Razzazi MR, Ghasemi T. An exact algorithm for the multiple-choice multidimensional knapsack based on the core. In: Sarbazi-Azad H, Parhami B, Miremadi SG, Hessabi S. (eds) Advances in Computer Science and Engineering. Communications in Computer and Information Science, Springer, Berlin, Heidelberg. 2008;6.
42. Quan N, Kim H. A tight upper bound for quadratic knapsack problems in grid-based wind farm layout optimization. 2018;50(3): 376-381.
43. Nirmala MP, Parvathi MS. MMKP B&B based Heterogeneous handover for the next generation network's. International

Journal of Engineering Research & Management Technology. 2015;2(3):20-30.

44. Gokce EI, Wilhelm WE. Valid inequalities for the multi-dimensional multiple-choice 0-1 knapsack problem. Discrete Optimization. 2015;17(C):25-54.

45. Goyal S, Parashar A. A proposed solution to knapsack problem using branch & bound technique. International J. for Innovation Research Multidisciplinary Field. 2016;2(7):240-246.

46. Bettinelli A, Cacchiani V, Malaguti E. A branch-and-bound algorithm for the knapsack problem with conflict graph. INFORMS Journal on Computing. 2017; 29(3):457-473.

47. Wang F. A new exact algorithm for concave knapsack problems with integer variables. International Journal of Computer Mathematics. 2018:1-19.

APPENDIX A

Example: consider the following multiple-choice knapsack problem:

$$Max.\, z = (0,.13,.21)\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix} + (0,.12,.16,.24)\begin{pmatrix} x_{21} \\ x_{22} \\ x_{23)} \\ x_{24} \end{pmatrix} + (0,.11,.23)\begin{pmatrix} x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} + (0,.14,.25)\begin{pmatrix} x_{41} \\ x_{42} \\ x_{43)} \end{pmatrix}$$

$$(0,4,7)\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix} + (0,3,4,9)\begin{pmatrix} x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{pmatrix} + (0,2,7)\begin{pmatrix} x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} + (0,3,7)\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix} \le 23$$

$$(0,3,8)\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix} + (0,4,5,8)\begin{pmatrix} x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{pmatrix} + (0,6,7)\begin{pmatrix} x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} + (0,4,7)\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix} \le 25$$

The equivalent mathematical form is as follow:

$$Max.\, z = (0,.13,.21)X_1^T + (0,.12,.16,.24)X_2^T + (0,.11,.23)X_3^T + (0,.14,.25)X_4^T$$
$$(0,4,7)X_1^T + (0,3,4,9)X_2^T + (0,2,7)X_3^T + (0,3,7)X_4^T \le 23$$
$$(0,3,8)X_1^T + (0,4,5,8)X_2^T + (0,6,7)X_3^T + (0,4,7)X_4^T \le 25$$
$$X_1 = (1,2,3), X_2 = (1,2,3,4), X_3 = (1,2,3), X_4 = (1,2,3).$$

Solution: We first determine the initial G matrix of DBB1 procedure as follows:

$$G = \begin{bmatrix} 0+ & 0 & 0 & 0 \\ .44+ & .41 & .30 & .48+ \\ .34 & .43 & .42 & .43 \\ - & .34 & - & - \end{bmatrix}$$

The steps of the DBB1 algorithm are summarized in the following table:

t	**Set of the partial solution** SP^t	(s_1^t, s_2^t)	z^t	$\underline{z}$	**Set of the free variables** F^t	**Note**
0	Φ	(-7,-5)	.93	0	$\{X_1(1,2), X_2(1,2,3), X_3(1,2), X_4(1,2)\}$	
1	$\{X_4(2)\}$	(-1,-1)	.82	0	$\{X_1(1,2), X_2(1,2,3), X_3(1,2), X_4(1)\}$	
2	$\{X_4(2), X_1(2)\}$	(0,3)	.74	.74	$\{X_1(1), X_2(1,2,3), X_3(1,2), X_4(1)\}$	Fathomed by feasibility
3	$\{X_4(2), X_1(-2)\}$	(-1,-1)	.82	.74	$\{X_1(1), X_2(1,2,3), X_3(1,2), X_4(1)\}$	
4	$\{X_4(2), X_1(-2,1)\}$	(4,6)	.61	.74	$\{X_2(1,2,3), X_3(1,2), X_4(1)\}$	Fathomed by lower bound
5	$\{X_4(2), X_1(-2,-1)\}$	(-1,-1)	.82	.74	$\{X_2(1,2,3), X_3(1,2), X_4(1)\}$	
6	$\{X_4(2), X_1(-2,-1), X_2(3)\}$	(2,1)	.74	.74	$\{X_2(1,2), X_3(1,2), X_4(1)\}$	Fathomed by feasibility
7	$\{X_4(2), X_1(-2,-1), X_2(-3)\}$	(-1,-1)	.82	.74	$\{X_2(1,2), X_3(1,2), X_4(1)\}$	
8	$\{X_4(2), X_1(-2,-1), X_2(-3,2)\}$	(2,-1)	.70	.74	$\{X_2(1), X_3(1,2), X_4(1)\}$	Fathomed by lower bound
9	$\{X_4(2), X_1(-2,-1), X_2(-3,-2)\}$	(-1,-1)	.82	.74	$\{X_2(1), X_3(1,2), X_4(1)\}$	
	$\{X_4(2), X_1(-2,-1), X_2(-3,-2,1)\}$	(2,3)	.58	.74	$\{X_3(1,2), X_4(1)\}$	
10	$\{X_4(2), X_1(-2,-1), X_2(-3,-2,-1), X_3(2)\}$	(2,-1)	.70	.74	$\{X_3(1), X_4(1)\}$	Fathomed by lower bound
11	$\{X_4(2), X_1(-2,-1), X_2(-3,-2), X_3(-2)\}$	(-1,-1)	.82	.74	$\{X_3(1), X_4(1)\}$	
12	$\{X_4(2), X_1(-2,-1), X_2(-3,-2), X_3(-2,1)\}$	(4,5)	.59	.74	$\{X_4(1)\}$	Fathomed by upper bound
13	$\{X_4(2), X_1(-2,-1), X_2(-3,-2), X_3(-2,-1)\}$	(-1,-1)	.82	.74	$\{X_4(1)\}$	
14	$\{X_4(-2)\}$					Since all the elements in the set of partial solution are negative the algorithm is terminated

Since all the elements in the set of partial solution are negative the algorithm is terminated. The optimal solution is obtained from the partial solution which has the updated lower bound. The updated upper bound is achieved in step *t* = 6 with the following optimal solution:

The optimal partial solution: $\{X_4(2), X_1(-2,-1), X_2(3)\}$, z*=0.74.

The optimal values of the original variables are: $z^* = 0.74,$ and $x_{42}^* = 1, x_{23}^* = 1, x_{13}^* = 1, x_{33}^* = 1,$ and $x_{ij}^* = 0,$ for all the other *i*'s and *j*'s.

It is to be noted that in this example an alternative optimal solution can also recognized in Step2 as follow:

The optimal partial solution: $\{X_4(2), X_1(2)\}$ z*=0.74.

The optimal values of the original variables are: $z^* = 0.74,$ and $x_{42}^* = 1, x_{12}^* = 1,$ and $x_{ij}^* = 0,$ for all the other *i*'s and *j*'s.

© 2018 Ghasemi-Tari et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history/24014