Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2011, Article ID 218078, 11 pages
doi:10.1155/2011/218078

Research Article

A Heuristic Algorithm for Resource Allocation/Reallocation Problem

S. Raja Balachandar and K. Kannan

Department of Mathematics, School of Humanities & Sciences, SASTRA University, Tamil Nadu, Thanjavur 613401, India

Correspondence should be addressed to S. Raja Balachandar, srbala09@gmail.com

Received 3 February 2011; Revised 16 June 2011; Accepted 19 July 2011

Academic Editor: Yuri Sotskov

This paper presents a *1-opt* heuristic approach to solve resource allocation/reallocation problem which is known as 0/1 multichoice multidimensional knapsack problem (MMKP). The intercept matrix of the constraints is employed to find optimal or near-optimal solution of the MMKP. This heuristic approach is tested for 33 benchmark problems taken from OR library of sizes upto 7000, and the results have been compared with optimum solutions. Computational complexity is proved to be $O(klmn^2)$ of solving heuristically MMKP using this approach. The performance of our heuristic is compared with the best state-of-art heuristic algorithms with respect to the quality of the solutions found. The encouraging results especially for relatively large-size test problems indicate that this heuristic approach can successfully be used for finding good solutions for highly constrained NP-hard problems.

1. Introduction

A cellular network is a mobile network in which resources are managed in cells. Each cell, an abounded area, is served by an antenna or base station. Cell size and shape depend on signal strength, the presence of obstacles to signal propagation, customer capacity, and cost constraints. Allocation/reallocation problem is formulated as 0/1 multichoice multidimensional knapsack problem (MMKP), which is NP-hard combinatorial optimization problem and detailed study of this problem is presented in [1]. Literally MMKP is defined as follows: Given a set of groups of variables, one tries to select the best variable in each group. Each variable in a group has a value in an objective function and consumes a certain amount of resources as well. The problem is to select the variables, subject to resource constraints, so that the objective function is maximized.

The 0/1 multichoice multidimensional knapsack problem may thus be mathematically formulated as follows: given n groups $l_1, \ldots, l_n$ of items to pack in some knapsack of capacity

b_k. Each item has a profit p_{ij} and a weight r_{kij}, and the problem is to choose one item from each group such that the profit sum is maximized without having the weight sum to exceed b_k:

$$\text{Maximize} \quad z = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j \in l_i} p_{ij} x_{ij}, \tag{1.1}$$

$$\text{Subject to} \quad \sum\nolimits_{i=1}^{n} \sum\nolimits_{j \in l_i} r_{kij} x_{ij} \le b_k, \quad k = 1, \dots, m, \tag{1.2}$$

$$\sum\nolimits_{j \in l_i} x_{ij} = 1, \quad i = 1, \dots, n, \tag{1.3}$$

$$x_{ij} \in \{0, 1\}. \tag{1.4}$$

All coefficients p_{ij}, r_{kij}, and b_k are positive integers, and the classes $l_1, \dots, l_n$ are mutually disjoint. Totally there are $\sum_{i=1}^{n} l_i$ variables, and they are divided into n groups.

In the resource allocation/reallocation problem a group represents a terminal that needs to be reallocated. For terminals subjected to reallocation the m constraints contain the power estimation constraints, and the constraints express that a channel cannot allocate more than one terminal from a particular cell at a time. Equation (1.3) expresses that the terminals must be reallocated to a channel. For terminals to be allocated the m constraints contain the power estimation constraints, the constraints that express that a terminal cannot be allocated to more than one channel from a particular cell at time.

The other applications of MMKP are quality of service degradation model, utility model, multisession adaptive multimedia system, problem of allocation of resources on a packet network, and nursing personnel scheduling problem. MMKP has been solved by several algorithms, and they have been cited and compared as benchmarks many times in the literature.

This paper is organized as follows. A brief survey of various researchers' works pertaining to this problem is elucidated in Section 2. The dominant principles of intercept matrix, dominance principle-based Heuristic (DPH) approach for solving MMKP, and computational complexity of DPH are explained in Section 3. We have furnished the results obtained by DPH for all the benchmark problems in Section 4. This section also includes the extensive comparative study of results of our heuristic with known optimum or best solutions of MMKP. Salient features of this algorithm are also enumerated in Section 4, and finally concluding remarks and future direction are also given in Section 5.

2. Previous Work

Depending on the nature of the solution, the existing algorithm for MMKP can be divided into two groups, namely, exact algorithm striving for exact solutions and heuristic algorithms producing near-optimal solution. The Exact algorithm includes branch and bound and Lagrange multipliers technique [2, 3].

Khan et al. [2, 4] presented an exact algorithm for the MMKP based on branch and bound with linear programming technique. A solution is explored in each iteration of this algorithm by picking the items of a particular group. All the possible alternative picks of a group are estimated by applying linear programming. The solution with highest estimated total value is selected for further exploration. A solution is termed as the optimal solution if it gives the highest total value among all the explored solutions and an item from each group is already picked. Khan et al. [4] had applied the concept of aggregate resource consumption [5] to pick a new candidate item in a group to solve the MMKP which is resulted in a heuristic named HEU. In HEU, a new item of group is selected for possible upgrade if it gives the highest change in earned value per unit change of aggregate resource consumption. Akbar et al. [6] presented a modified version of HUE, namely, M-HEU. In M-HEU, a preprocessing step to find a feasible solution and a postprocessing step to improve the total value of the solution with one upgrade followed by one or more downgrades were added.

Moser et al. [3] used the concept of graceful degradation from the most valuable items based on Lagrange multipliers to solve the MMKP. The selected highest valued item might be infeasible because of high resource consumption. That is why graceful degradation is done in multiple iterations towards selection of a feasible suboptional solution. Hifi et al. [7, 8] proposed two different approximate approaches. The first approach is a guided local search-based heuristic in which the trajectories of the solutions were oriented by increasing the cost function with a penalty term; it penalizes bad features of preciously visited solutions.

The second approach is a reactive local search where an explicit check for the repetition of configuration is added to the local search. The algorithm starts by an initial solution and improves by using a fast literature procedure. Later both deblocking and degrading procedure are introduced in order to (i) escape to local optimal and to (ii) introduce diversification in the search space. Finally, a memory list is applied in order to forbid the repetition of configuration. Recently Cherfi and Hifi [9, 10] have developed two hybrid algorithms, namely, HLB and AHLB, and compared the solutions with another procedure called column generation method (ALGO) [10]. HLB is the combination of local branching and column generation solution procedure, and AHLB is known as augmented HLB. These two algorithms are extended version of Hifi et al.'s [7, 8] previous work. The computational time of these algorithms was fixed from 300 to 1200 sec. The overall best solutions of these two algorithms are presented and compared with our heuristic in Section 4.

Drexl [11] presented a simulated annealing approach to solve a slightly different variant of MMKP (without choice constraints (1.3)), namely, the multidimensional knapsack problem (MDKP). Genetic algorithm approaches are not suitable to solve for real-time admission control, as they require long time to find a suboptional solution.

Parra-Hernández and Dimopoulos [12] proposed another heuristic HMMKP, called linear programming relaxations of the MDKP reduced from the MMKP problem. A PRAM model approximation algorithm was devised by Newton et al. [13] for solving the MMKP in $O(\log n(\log n + \log m + \log l))$ time using $O(n \log n(n + lm))$ operations. Shahriar et al. [14] proposed a multiprocessors-based heuristic algorithm (MP-HEU) for MMKP which runs $O(t/p+s(p))$ time, where t is the time requested by the algorithm using single processors, p is the number of processors, and $s(p)$ is a function of p, the synchronization overhead. Sbihi [15] has presented a best first search exact algorithm for the MMKP. The main principle of the algorithm is twofold: (i) to generate an initial feasible solution as a starting lower bound and (ii) at different levels of the search tree to determine an intermediate upper bound obtained by solving an auxiliary problem and perform the strategy of fixing items during the exploration.

In this paper, we propose a *1-opt* heuristic algorithm based on dominance principle of intercept matrix to solve MMKP. The dominance principle-based heuristic algorithm has been implemented successfully to solve 0-1 multiconstrained knapsack problem [16]. The main principle of the algorithm is twofold: (i) to generate an initial feasible solution as a starting lower bound and (ii) to improve the initial feasible solution to optimal or near optimal by using this heuristic iteratively.

3. Dominance Principle-Based Heuristic (DPH)

Many researchers have included all the possible constraints and variables in linear programming (LP) model of the real-life optimization problems, but some of the constraints and variables may not involve in the optimality and it may consume additional computational complexity. Such variables and constraints are known as redundant constraints and variables.

The preprocessing techniques are necessary to remove such redundant constraints and variables. Researchers [4, 17–25] have proposed many algorithms for LP models; in particular, Paulraj et al. [26] used the intercept matrix of the constraints to identify redundant constraints prior to the start of the solution process in his heuristic approach. Here we used the intercept matrix to identify redundant variables (0 valued variable) as well as selected variables (1 valued variable) for solving MMKP, since MMKP is a well-known 0-1 integer programming problem and many variables have zero values.

The intercept matrix of the constraints (1.2) is used to identify the variables of value 1 and 0. The algorithm starts by selecting the lowest cost-valued item of each group as (initial) feasible solution (Step 1) and improves the objective value by using dominance principle of intercept matrix (Step 3 to Step 5). This sequence of operations is performed ten times.

The construction of the intercept matrix (by dividing b_k values by coefficients of (1.2)) is explored in Step 3. The elements of intercept matrix are used to find total_j and $c_j * \text{total}_j$ which are arranged in decreasing order, and the leading element is the dominant variable (Step 5). We use this dominant variable to improve the current feasible solution, and this procedure provides optimum or near-optimum solution of MMKP. The dominant principle focuses at the resource matrix with lower requirement come forward to maximize the profit. The intercept matrix of the constraints (1.2) plays a vital role in achieving the goal, in a heuristic manner. Step 4 is used to identify the 0-value variables, that is, redundant variables. The various stages of DPH are presented in Algorithm 1.

3.1. Example

Consider an MMKP with 3 groups, 8 items, and 2 resources, that is, $n = 3$, $\sum_{i=1}^{n} l_i = 8$, and $m = 2$:

$$\begin{aligned} \text{Maximize} \quad & 10x_{11} + 14x_{12} + 9x_{13} + 11x_{21} + 13x_{22} + 12x_{31} + 7x_{32} + 17x_{33}, \\ \text{subject to} \quad & 5x_{11} + 4x_{12} + 5x_{13} + 8x_{21} + 6x_{22} + 7x_{31} + 5x_{32} + 10x_{33} \le 17, \\ & 7x_{11} + 7x_{12} + 5x_{13} + 2x_{21} + 4x_{22} + 7x_{31} + 3x_{32} + 8x_{33} \le 15 \end{aligned} \tag{3.1}$$

(we convert two-dimensional problem into one-dimensional notation for our convenience),

$$\begin{aligned} \text{Maximize} \quad & 10x_1 + 14x_2 + 9x_3 + 11x_4 + 13x_5 + 12x_6 + 7x_7 + 17x_8, \\ \text{subject to} \quad & 5x_1 + 4x_2 + 5x_3 + 8x_4 + 6x_5 + 7x_6 + 5x_7 + 10x_8 \le 17, \\ & 7x_1 + 7x_2 + 5x_3 + 2x_4 + 4x_5 + 7x_6 + 3x_7 + 8x_8 \le 15. \end{aligned} \tag{3.2}$$

Initial feasible solution: by using Step 1 of the algorithm, we have found the initial feasible solution of MMKP (001, 01,010) and objective value is 29. Next we update this solution by using DPH algorithm iteratively (from Step 3 to Step 5). Table 1 illustrates the first iteration

```
Notation:
        $z^*$: current objective function value
        $z$: new objective function value
        $x^*$: current solution vector
        $x$: new solution vector
initial value:
        $x, x^*$: zero valued vectors
        $z, z^*$: zero
        $k$: 1 (iteration)
Step 1: For $g = 1$ to $n$ do begin
        $t = \arg\min_{1 \le j \le l_g} \{\sum r_{gij}\},\ i = 1$ to $m$,
        $x_{gt} = 1$,
        $z = z + p_{gt}$
        end for
do begin
    $k = k + 1,\ z^* = z,\ x^* = x$
Step 2: For our convenience convert
        $p_{ij}$ in to $c_j$,   $j = \sum l_i$,
        $r_{kij}$ in to $a_{ij}$, $i = 1$ to $m$,
                          $j = \sum l_i$
For $g = 1$ to $n$ do begin
Step 3: Intercept matrix $d_{ij} = \begin{cases} b_i / a_{ij}, & a_{ij} > 0, \\ M(\text{a large value}) & \text{otherwise} \end{cases}$
Step 4: Redundant variables $rv$
        $rv = \arg_{1 \le j \le n} \{d_{ij} < 1\},\ i = 1, \dots, m$,
        $a_{irv} = 0,\ \forall i$,
        $x_{rv} = 0$
Step 5: Dominant variable $s$
        $\text{total}_j = \sum d_{ij},\ i = 1, \dots, m,\ j = 1, \dots, l_g$,
        $s = \arg\max\{\text{total}_j * c_j\},\ \forall j$
        if $x_s = 0$ and $p_s > p_{j^*}$ ($j^*$—the variable which is already in the solution)
        then
        $x_s = 1$,
        $z = z - p_{j^*} + p_s$,
        $b_i = b_i - a_{is} + a_{ij^*},\ \forall i$
        else
        $z = z + p_{j^*}$,
        $b_i = b_i - a_{ij^*},\ \forall i$
        end if
end for
while ($z^* < z$ and $k \le 10$)
Display $x$ and $z$.
```

Algorithm 1: DPH algorithm for MMKP.

reports of DPH algorithm. This heuristic updates the solution vector and objective function value.

Iteration 1. *Consider* $g = 1$.

2nd item in group 1 dominates other items, and it improves the objective function value from 29 to 34. Thus, the new solution vector is (010, 01,010) and objective function value is 34.

Table 1: First Iteration reports of DPH.

$g = 1$	Group 1			Group 2		Group 3		
Const 1	3.4	4.25	3.4	2.125	2.833333	2.428571	3.4	1.7
Const 2	2.142857	2.142857	3	7.5	3.75	2.142857	5	1.875
Total = $\sum_{i=1}^{m} b_i / a_{ij}$	5.542857	6.392857	6.4	9.625	6.583333	4.571429	8.4	3.575
Total $*\ c_j$	55.42857	**89.5**	57.6	105.875	85.58333	54.85714	58.8	60.775
$g = 2$	Group 1			Group 2		Group 3		
Const 1	—	—	—	1.625	2.166667	2.428571	2.6	$*$
Const 2	—	—	—	2	2	2	2.666667	$*$
Total = $\sum_{i=1}^{m} b_i / a_{ij}$	—	—	—	3.625	4.166667	4.428571	5.266667	$*$
Total $*\ c_j$	—	—	—	39.875	**54.16667**	53.14286	36.86667	$*$
$g = 3$	Group 1			Group 2		Group 3		
Const 1	—	—	—	—	—	2.428571	2.6	$*$
Const 2	—	—	—	—	—	2	2.666667	$*$
Total = $\sum_{i=1}^{m} b_i / a_{ij}$	—	—	—	—	—	4.428571	5.266667	$*$
Total $*\ c_j$	—	—	—	—	—	**53.14286**	36.86667	$*$

$*$: less than 1 in any one of the entry in that column, —: group is omitted.
There is no improvement in the 2nd iteration.

Consider $g = 2$.

There is no dominated variable other than the 2nd item in group 2. In this iteration no feasible upgrade is possible.

Consider $g = 3$.

2nd item in group 3 dominates other items, but it is available already.

In this iteration no feasible upgrade is possible. Thus, the final solution vector is (010, 01,010) and corresponding objective function value is 34.

Iteration 2. For $g = 1, 2$, and 3, there is no change in the objective function value. We terminate the iteration process and display the final solution, 45 which is optimum.

Theorem 3.1. *DPH can be realized in $O(klmn^2)$ time, polynomial in the number of groups (n), item types (l), constraints (m), and number of iterations (k).*

Proof. The computational complexity of finding the heuristic solution of MMKP using DPH can be obtained as follows. For simplicity, let us assume that the number of variables per group is a constant l (in case of different numbers of items per group, assume that l is the maximum number of elements in a group) and there are m resources and n groups. It is easy to verify that the procedure for construction of intercept matrix takes $O(lmn)$ operations. Step 3 and Step 4 take $O(ln)$ and $O(ln)$ operations, respectively. One iteration performs n times to complete the groupwise selection. So the overall running time of the procedure DPH for one iteration can be deduced as follows:

$$O\left(lmn^2\right) + O\left(ln^2\right) + O\left(ln^2\right) = O\left(lmn^2 + ln^2 + ln^2\right) = O\left(lmn^2\right). \tag{3.3}$$

For k iteration algorithm, the overall time complexity is $O(klmn^2)$. □

4. Experimental Design

The heuristic was tested on 33 instances corresponding to two groups; the first group contains existing instances, namely, I01 to I13 (13 instances) which was tested by Khan [2], and the second group consists of 20 instances, namely, Ins01 to Ins20 which were randomly generated by Hifi et al. [8]. Our algorithm was coded in C++ and performed on Pentium IV 2.40 GHz computer with 512 MB memory, running under Windows XP Professional. For all the 33 instances DPH has reached the optimum/best/improved solutions. Table 2 illustrates the computational results of DPH, Cplex, ALGO, and HLB, AHLIB solutions with known optimum/best solutions [8] for the test instances (°indicates optimum solution and *indicates best solution). DPH reaches the best solution for the problem I07 in the first iteration itself, and the first iteration objective function values of this problem for $n = 1$ to 100 are presented in Figure 1.

The comparative study of DPH with other existing heuristic algorithms (HMS: Hifi et al. algorithm [7]; RLS, MRLSa, MRLSb: Reactive Local Search, Modified Reactive Local Search algorithm [8]; Cplex: Cplex Solver; KLMA: Khan et al.'s algorithm [4]; MOSER: Moser et al.'s algorithm [3]; Opt/best: optimum or best solutions [8]; ALGO: column generation method [10]; HLB_{CGSP}: local branching and column generation [9]; $\text{AHLB}_{\text{CGSP}}$: augmented hybrid procedure [9]) has been furnished in Tables 3(a) and 3(b) in terms of the number of optimal or best solutions, the average deviation from optimal/best solution obtained for group 1 problem, and the number of optimum/overall best solutions.

The MOSER et al. [3] approach is heuristic and is based on Lagrange's multiplier method. Khan et al. [4] use an iterative improvement procedure, namely, KLMA based on the concept of aggregate resources, which is presented in [5]. Both of these methods have failed to find the optimum/best solution for group-1 instances, but KLMA identifies the optimum solution for the problem I06. The average deviations of KLMA and MOSER [3] approaches from optimum/best solutions are 1.46% and 5.99%, respectively.

Hifi et al. [7] has developed two greedy approaches for MMKP, namely, constructive procedure (CP) and complementary constructive procedure (CCP). The detailed description of CP and CCP can be found in [9]. To compare the performance of DPH, we consider the best solution of Hifi et al.'s [7] algorithm referred to herein as HMS. The HMS determines the optimum solutions for I01 and I06 only, and the average deviation of HMS from the optimum/best is known to be 1.92%. Hifi et al. [8] presented two more algorithms for MMKP, namely, reactive local search (RLS) and modified reactive local search (MRLS). RLS approach improves the solution obtained by CCP. The core of the algorithm is mainly based on two strategies, namely, degrading and deblocking. The detailed procedures are available in [8]. RLS presents the optimum solutions for 4 instances, namely, I01, I02, I05, and I06. The average deviation is 1.07%. The modified versions of RLS are known as MRLSa and MRLSb. MRLSa has obtained 4 optimum solutions (I01, I02, I03, and I06), and average deviation is 0.91%. For MRLSb, the average deviation is zero and finds optimum/best solutions for all the test instances except I12; the deviation from the best solution is 0.002%. The best results of Cplex Solver are also compared with DPH; Cplex obtained 8 optimum/best solutions out of 13 instances (I01 to I06, I10, and I12), and the average deviation is 0.01%.

Cherfi and Hifi [9, 10] have developed three approaches for solving MMKP, namely, ALGO, HLB, and AHLB. The column generation method (ALGO) identified the optimum/best solution for all the instances of group-1 problems. It has reached overall best solution for 7 instances (I01 to I07). Cherfi and Hifi [9, 10] have used three time limits for HLB. We consider the best solution among the three time bounds. HLB has found optimum/best solution for all test instances in group 1 and obtained 7 overall best solutions (I01 to I07).

Table 2: Computational results of Cplex, ALGO, HLB, AHLB, and DPH with known optimum/best [8].

Ins	Optimum/best	DPH multiple run solutions (maximum 10 iterations)			Cplex	ALGO	HLB	AHLB
		Initial/min	Max	Iter				
I01	173°	127	173	2	173	173	173	173
I02	364°	304	364	1	364	364	364	364
I03	1602°	1202	1602	3	1602	1602	1602	1602
I04	3597°	2497	3597	2	3597	3597	3597	3597
I05	3905.70°	2907.70	3905.70	2	3905.70	3905.70	3905.70	3905.70
I06	4799.30°	3796.30	4799.30	4	4799.30	4799.30	4799.30	4799.30
I07	24587*	21356	24587*	1	24584	24587*	24587*	24587*
I08	36877	29467	36894*	5	36869	36892	36878	36894*
I09	49167	40457	49179*	6	49155	49176	49171	49179*
I10	61437	53466	61464*	3	61446	61461	61450	61464*
I11	73773	62731	73783*	6	73759	73775	73777	73783*
I12	86071	66045	86080*	6	86071	86078	86078	86080*
I13	98429	72457	98438*	7	98418	98431	98431	98438*
Ins01	10714	4554	10738*	8	10709	10732	10732	10738*
Ins02	13598*	9459	13598*	4	13597	13598*	13598*	13598*
Ins03	10943	7305	10944*	2	10934	10943	10944*	10944*
Ins04	14429	10249	14442*	1	19422	14440	14432	14442*
Ins05	17053*	12034	17053*	2	17041	17053*	17053*	17053*
Ins06	16823	11237	16828*	2	16815	16825	16826	16827
Ins07	16423	12235	16440*	4	16407	16435	16432	16440*
Ins08	17506	11068	17510*	2	17484	17510*	17508	17510*
Ins09	17754	14545	17761*	3	17747	17760	17758	17761*
Ins10	19314	13422	19316*	2	19285	19314	19315	19316*
Ins11	19431	13109	19441*	5	19424	19434	19434	19441*
Ins12	21730	16253	21732*	4	21725	21731	21731	21732*
Ins13	21569	19345	21577*	2	21569	21575	21571	21577*
Ins14	32869	27109	32874*	2	32866	32870	32871	32874*
Ins15	39154	32568	39160*	2	39154	39157	39151	39160*
Ins16	43357	39733	43363*	3	43357	43361	43357	43362
Ins17	54349	34453	54360*	5	54349	54349	53454	54360*
Ins18	60456	40340	60464*	3	60455	60460	60457	60464*
Ins19	64921	59931	64924	2	64919	64923	64924	64925*
Ins20	75603	67890	75611	3	75603	75611	75609	75612*

AHLB has also been executed with two time limits, and the best results are considered for this comparative study. AHLB provides optimum/best solution for the entire test instances in group 1 and has got overall best for all the 13 instances. DPH was set to perform maximum 10 iterations and got the optimum/best solution for all the test instances. Since the working environment is different, we have not compared the running time of all the algorithms. DPH reaches high-quality solutions within 10 seconds, whereas HLB and AHLB needs more computational time to achieve this kind of high-quality solutions. ALGO, HLB, and AHLB

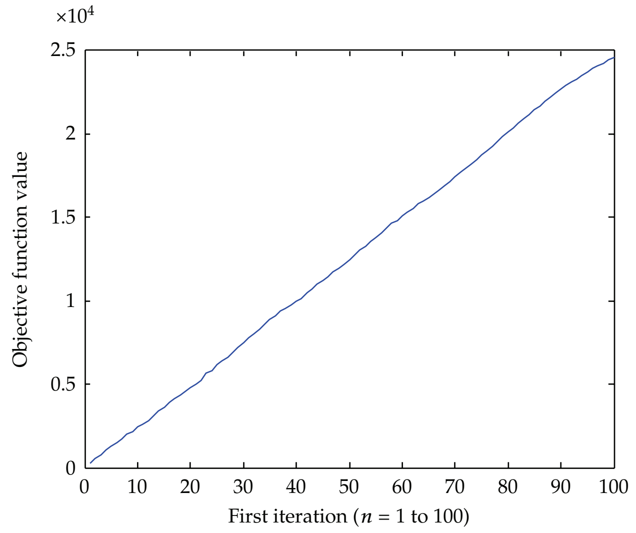

Figure 1: First iteration objective function value for I07.

were coded in C++ and run on an Ultra-Sparc 10. The maximum time requirement for this high-quality solution, ALGO, HLB, and AHLB is found to be 1200 sec.

For group-2 instances, the optimum solutions are not available in the literature because of hardness of the problem. The computational results of MRLSb, Cplex, ALGO, HLB, AHLB, and DPH are reported in Table 3(b). From the best solutions of Hifi et al. [8], we observe that ALGO, AHLB, and DPH reached the best for all the 20 instances, and the other algorithms, HLB, MRLSb, and Cplex, have reached 19, 18, and 6 instances of group 2, respectively. In the same time ALGO, HLB, AHLB, and DPH have given the improved solutions for some of the test instances which are superior to Hifi et al. [8]. In Table 3(b), we have also presented the number of overall best solutions for these test instances.

From Tables 3(a) and 3(b), we conclude that DPH is an efficient technique than the other algorithms. The reason is that DPH reached these solutions in 10 seconds (maximum) for all the test instances, where ALGO, HLB, and AHLB reached with more computational time [9, 10]. The time complexity of our algorithm is $O(klmn^2)$, where k is the number of iterations (we fixed k as 10).

4.1. Salient Features of DPH

This heuristic is used to reduce the search space to find the near-optimal solutions of the MMKP. The computational complexity is $O(klmn^2)$, and the space complexity is $O(lmn)$. It reaches the optimum or near-optimum point in n iterations, where n is the number of groups. Due to dominance principles, this heuristic identifies the zero-value variables instantaneously. DPH takes maximum CPU time of 10 seconds for large-size problem. From the comparison table we observe that our algorithm is the effective one.

5. Conclusion

In this paper, we presented the dominant principle-based heuristic approach for tackling the NP-hard 0/1 multichoice multidimensional knapsack problem. This approach has been

Table 3: (a) Summarized results for the solution quality of group 1 problem. (b) Summarized results forthe solution quality of group 2 problem.

(a)

	HMS	RLS	MRLSa	MRLSb	Cplex	MOSER	KLMA	ALGO	HLB	AHLB	DPH
Number of problems	13	13	13	13	13	13	13	13	13	13	13
Number of optimum/best	2	5	4	12	8	0	1	13	13	13	13
Average deviation	1.92	1.07	0.91	0.00	0.01	5.99	1.46	0.00	0.00	0.00	0.00
Number of overall best/optimum	0	0	0	0	0	0	0	7	7	13	13

(b)

	MRLSb	Cplex	ALGO	HLB	AHLB	DPH
Number of problems	20	20	20	20	20	20
Number of best	18	6	20	19	20	20
Average deviation	0.002	0.04	0.00	0.0003	0.00	0.00
Number of overall best	0	0	3	3	18	18

tested on 33 state-of-art benchmark instances and has led to give optimal/best solutions for all the test instances given in the literature. The maximum computational time is 10 seconds, where the other recent algorithm requires 1200 seconds. This heuristic is with $O(klmn^2)$ complexity, and it requires k iterations to solve the MMKP. The experimental data show that the optimality/best achieved by this heuristic is always close to 100 percentages. The basic idea behind the proposed scheme may be explored to tackle other NP-hard problems.

Acknowledgment

The authors wish to express their gratitude to the referees for their careful reading of the manuscript and helpful suggestions.

References

[1] R. Parra-Hernández and N. Dimopoulos, “Channel resource allocation/reallocation in cellular communication and linear programming,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2983–2989, October 2003.

[2] S. Khan, *Quality adaptation in a multi-session adaptive multimedia system: model and architecture*, Ph.D. dissertation, University of Victoria, Victoria, Canada, 1998.

[3] M. Moser, D. P. Jokanovic, and N. Shiratori, “An algorithm for the multidimensional multichoice knapsack problem,” *IEICE Trans Fundam Electron*, vol. 80, no. 3, pp. 582–589, 1997.

[4] S. Khan, K. F. Li, E. G. Manning, and M. D. M. Akbar, “Solving the Knapsack problem for adaptive multimedia systems,” *Studia Informatica*, vol. 2, pp. 154–174, 2002.

[5] Y. Toyoda, “A simplified algorithm for obtaining approximate solutions to zero-one programming problems,” *Management Science*, vol. 21, no. 12, pp. 1417–1427, 1974/75.

[6] M. M. Akbar, E. G. Manning, G. C. Shoja, and S. Khan, “Heuristic solutions for the multiple-choice multi-dimension knapsack problem,” in *the International Conference on Computional Science*, V. N. Alexandrov, J. Dongarra, B. A. Juliano, R. S. Renner, C. Jeng, and K. Tan, Eds., Lecture Notes in Computer Science, pp. 659–668, San Francisco, Calif, USA, May 2001.

[7] M Hifi, M Micharfy, and A Sbihi, “Heuristic algorithms for the multiple-choice multi-dimensional knapsack problem,” *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1323–1332, 2004.

[8] M. Hifi, M. Michrafy, and A. Sbihi, “A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem,” *Computational Optimization and Applications*, vol. 33, no. 2-3, pp. 271–285, 2006.

[9] N. Cherfi and M. Hifi, “Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem,” *International Journal of Operational Research*, vol. 5, no. 1, pp. 89–109, 2009.

[10] N. Cherfi and M. Hifi, “A column generation method for the multiple-choice multi-dimensional knapsack problem,” *Computational Optimization and Applications*, vol. 46, no. 1, pp. 51–73, 2010.

[11] A. Drexl, “A simulated annealing approach to the multiconstraint zero-one knapsack problem,” *Computing*, vol. 40, no. 1, pp. 1–8, 1988.

[12] R. Parra-Hernández and N. J. Dimopoulos, “A new heuristic for solving the multichoice multidimensional knapsack problem,” *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 35, no. 5, pp. 708–717, 2005.

[13] M. A. H. Newton, M. W. H. Sadid, and M. M. Akbar, “A parallel heuristic algorithm for multiple-choice multidimensional knapsack problem,” in *the International Conference on Computer and Information Technology*, pp. 181–184, Dhaka, Bangladesh, December 2003.

[14] A. Z. M. Shahriar, M. M. Akbar, M. S. Rahman, and M. A. H. Newton, “A multiprocessor based heuristic for multi-dimensional multiple-choice knapsack problem,” *Journal of Supercomputing*, vol. 43, no. 3, pp. 257–280, 2008.

[15] A. Sbihi, “A best first search exact algorithm for the multiple-choice multidimensional knapsack problem,” *Journal of Combinatorial Optimization*, vol. 13, no. 4, pp. 337–351, 2007.

[16] S. Raja Balachandar and K. Kannan, “A new polynomial time algorithm for 0-1 multiple knapsack problem based on dominant principles,” *Applied Mathematics and Computation*, vol. 202, no. 1, pp. 71–77, 2008.

[17] A. L. Brearley, G. Mitra, and H. P. Williams, “Analysis of mathematical programming problems prior to applying the simplex algorithm,” *Mathematical Programming*, vol. 8, pp. 54–83, 1975.

[18] J. Gondzio, “Presolve analysis of linear programs prior to applying an interior point method,” *INFORMS Journal on Computing*, vol. 9, no. 1, pp. 73–91, 1997.

[19] I. Ioslovich, “Robust reduction of a class of large-scale linear programs,” *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 262–282, 2001.

[20] M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, *Redundancy in Mathematical Programming: A State-of-the-Art Survey*, vol. 206 of *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin, Germany, 1983.

[21] T. H. Mattheiss, “An algorithm for determining irrelevant constraints and all verticles in systems of linear inequalities,” *Operations Research*, vol. 21, pp. 247–260, 1973.

[22] Cs. Mészáros and U. H. Suhl, “Advanced preprocessing techniques for linear and quadratic programming,” *Spectrum*, vol. 25, no. 4, pp. 575–595, 2003.

[23] N. V. Stojković and P. S. Stanimirović, “Two direct methods in linear programming,” *European Journal of Operational Research*, vol. 131, no. 2, pp. 417–439, 2001.

[24] J. Telgen, “Identifying redundant constraints and implicit equalities in systems of linear constraints,” *Management Science*, vol. 29, no. 10, pp. 1209–1222, 1983.

[25] J. A. Tomlin and J. S. Welch, “Finding duplicate rows in a linear programming model,” *Operations Research Letters*, vol. 5, no. 1, pp. 7–11, 1986.

[26] S. Paulraj, C. Chellappan, and T. R. Natesan, “A heuristic approach for identification of redundant constraints in linear programming models,” *International Journal of Computer Mathematics*, vol. 83, no. 8-9, pp. 675–683, 2006.