Chapter 5

MULTI-DEMAND MULTI-KNAPSACK PROBLEM

\(^1\) A part of this chapter has been accepted for publication in the Applied Mathematics and Computation as "A heuristic approach for the multi-demand multi-knapsack problem", 

\(^1\)
CHAPTER 5. MULTI-DEMAND MULTI-KNAPSACK PROBLEM

5.1 Introduction

Zero-One Multi Demand Multidimensional Knapsack Problem (MDMK) is an extension of multidimensional knapsack problem in which there are greater-than-or-equal-to inequalities, in addition to the standard less-than-or-equal-to constraints. Moreover, the objective function coefficients are not constrained in sign. It can be formulated as

\[
\text{maximize} \sum_{j=1}^{n} c_j x_j \quad (5.1)
\]

subject to

\[
\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \quad (5.2)
\]

\[
\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = m + 1, \ldots, m + q \quad (5.3)
\]

\[
x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n \quad (5.4)
\]

where \(b_i > 0, \forall i \in \{1, \ldots, m + q\}\),

\[a_{ij} \geq 0, \quad \forall i \in \{1, \ldots, m + q\}, \forall j \in \{1, \ldots, n\}\]

we also assume that \(\sum_{j=1}^{n} a_{ij} > b_i, \forall i \in \{1, \ldots, m + q\}\), \(max_j \{a_{ij}\} \leq b_i, \forall i \in \{1, \ldots, m\}\)

and \(min_j \{a_{ij}\} < b_i, \forall i \in \{m + 1, \ldots, m + q\}\), since any violation of these conditions would result in some \(x_j\)'s being fixed to zero or some constraints not
being satisfied or being redundant. Each of the m constraints of family (5.2) is called a knapsack constraint, while we refer to each of the q constraints of family (5.3) as a demand constraint.

Many practical problems have a structure of MDMK embedded, including portfolio location selection and capital budgeting [20] and obnoxious and semi-obnoxious facility location problem [29, 30, 169, 175]. These problems are NP-hard, and the use of heuristic search methods is highly competitive for even moderately sized instances. In this chapter, we propose a heuristic algorithm based on dominance principle of intercept matrix to solve MDMK.

Organisation of this chapter follows: In section 5.2 theoretical bounds for MDMK have been obtained for subsequent use in probabilistic performance analysis. A brief survey of various researchers’ work pertaining to this problem is explained in section 5.3. The dominant principle based Heuristic (DPH) approach for solving MDMK, worst-case analysis, probability performance analysis and computational complexity of DPH are explained in section 5.4. The results obtained by DPH for all the benchmark problems are furnished in section 5.5. This section also includes the comparative study of the results of our heuristic with known optimum or best solutions of MDMK. This chapter is concluded in section 5.6.

This heuristic is used here in the first stage, to find the feasible solution for demand constraints (5.3) and second stage is to find the optimal or near optimal solution by using another heuristic called column dominant principle and row dominant principle.

5.2 Bounds

Consider the linear relaxation of MDMK
\[ Z(\mathcal{P}) \quad z(\mathcal{P}) \overset{\text{Max}}{=} \sum_{j=1}^{n} c_j x_j \]

s.a. \[ \sum_{j=1}^{n} a_{ij} x_j \leq b_j, \: i=1,2,...,m+q \quad 0 \leq x_j \leq 1 \quad j=1,...,n. \]

solving this linear problem provides an upper bound (UB), which is much better than the bound consistency propagation of all the constraints \( \left( \sum_j c_j \text{ in this case} \right) \), and it provide also the so-called reduced costs.

Let \( (\bar{c}, \bar{\pi}) \) be the reduced costs associated to \( (x, s) \). Then

\[ Z(\mathcal{P}_{eq}) \quad \text{Max UB} + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{\pi}_i s_i \] \hspace{1cm} (5.5)

s.a. \[ \bar{A}x + \bar{S}s = \bar{b} \] \hspace{1cm} (5.6)

\[ 0 \leq x \leq 1, \: s \geq 0 \] \hspace{1cm} (5.7)

where \( \bar{A}, \bar{S}, \bar{b} \) depend of the basis associated to \( (\bar{x}, \bar{s}) \). and

\[ N^- = \{ j \in N/x_j \text{ is a non basic variable at its lower bound} \} \]

\[ N^+ = \{ j \in N/x_j \text{ is a non basic variable at its upper bound} \} \]

Actually, the equivalence is still valid if the integrality constraint \( x_j \in \{0,1\}; j=1,...,n; s_i \in N; i=1,...,m \) are maintained. Hence, the 01-MKP can be written as

\( (P_{eq}) \quad \text{Max} \quad UB + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{\pi}_i s_i \) \hspace{1cm} (5.8)

s.a. \[ \bar{A}x + \bar{S}s = \bar{b} \] \hspace{1cm} (5.9)
\[ x_j \in \{0, 1\}; j = 1, \ldots, n; s_i \in N; i = 1, \ldots, m \quad (5.10) \]

Nevertheless, since a feasible integer solution is known, we are interested in finding only better solutions, that is in finding feasible solutions satisfying the following "Reduced Costs Constrait":

\[
UB + \sum_{j \in N^-} c_j x_j - \sum_{j \in N^+} c_j (1 - x_j) + \sum_{i \in M} \pi_i s_i \geq LB 
\]

\[ (5.11) \]

\[
- \sum_{j \in N^-} c_j x_j + \sum_{j \in N^+} c_j (1 - x_j) - \sum_{i \in M} \pi_i s_i \leq UB - LB
\]

\[ (5.12) \]

where LB is the lower bound obtained by the heuristic (or the best known solution).

### 5.3 Previous Work

Most of the researches on knapsack problems deal with the much simpler constraint version \((m=1 \text{ and } n=1)\). For the single constraint case the problem is not strongly NP-Hard and effective approximation algorithms have been developed for obtaining near-optimal solutions. A good review of the single knapsack problem and its associated exact and approximate algorithm is given by Martello and Toth [143]. A brief review of knapsack problem variants is given below. We will particularly discuss the multidimensional knapsack problem (MDKP), the multiple-choice knapsack problem (MCKP), and the MDMK.
5.3.1 Exact Algorithms

The MDKP is a generalization of the classical binary knapsack problem for multiple resource constraints. For more details, one can refer to Chu and Beasley [41], Freville [71] and Plateau and Shih [169, 183]. Another variant of the knapsack problem is the MCKP, where the picking criterion for item \( w \) is more restricted. For the later variant of the knapsack problem there are one or more disjoint classes of items, for more details, reference may be made to [152]. Finally, the MDMK can be considered as a more generalization of the MDKP and MCKP variants of the binary knapsack problem (0-1 KP). Most algorithms for optimal solutions of knapsack problem variants are also based upon branch-and-bound procedures (Nauss [152], Khan [120] and Pisinger [167]).

A greedy algorithm has been proposed for approximately solving the knapsack problems (Martello and Toth [143, 144]). For the classical binary knapsack problem, the approach is composed of two stages (i) sort the items in decreasing order of value-weight ratio and (ii) pick as many items as possible from the left of the ordered list until the resource constraint is violated. By using the same principle for the MDKP, Toyoda [191] used the aggregate resources consumption. The solution of the MDKP needs iterative picking of items until the resource constraint is violated. Shih [183] presented a branch-and-bound algorithm for MDKP. In this method, an upper bound was obtained by computing the objective function value associated with the optimal fractional solution algorithm (Dantzig [49]) for the \( m \) single constraint knapsack problems and selecting the minimum objective function value among those as the upper bound. In the recent past, great success has been achieved via the application of local search techniques and metaheuristics to MDKP. Most popular techniques have been tabu search, genetic algorithms, simulated annealing and hybrid algorithms (for more details the reader can refer
5.3.2 Heuristic Algorithms

Very few research papers dealing directly with MDMK are available in the literature. Cappanera and Trubian [30] designed a heuristic approach in 2005. They employed a sophisticated two-phase search, with additional use of domination criteria and variable ordering. Their search first attempted to find feasible solutions, and thereafter a local search phase was initiated from each of the feasible solutions found (upto a maximum number), using both single-flip, swap and double-swap neighborhoods, but remaining feasible all the time. Their search was based on strategic oscillation ideas of Glover and Kochenberger[81]. They also showed that the feasible regions may be disconnected for the neighborhoods they use. Finding feasible solutions is a daunting task for some of test problem classes(those with the most demand constraints), with exact solver CPLEX having the most trouble. Arntzen [9] implemented a simple tabu search-based metaheuristic method for MDMK, augmented by adaptive memory and learning principles derived from tabu search ideas. Hvattum [131] have proposed an alternating control tree search (ACT) method for this problem.

In this chapter, we propose a heuristic based on dominance principle of intercept matrix to solve MDMK. The main principle of the algorithm is twofold. (i) to generate an initial feasible solution for demand constarints(5.3) (ii) improve this feasible solution to satisfy the knapsack constraints (5.2) and optimal or near optimal of (5.1) by using this heuristic.
5.4 Dominance Principle based Heuristic

MDMK is a well known 0-1 integer programming problem and many variables have zero values called redundant variables. We use intercept matrix of the constraints (5.2) to identify the variables of value 1 and 0. The DPH algorithm consists of two phases. The first phase is used to assign the values of the variables from zero to one as long as feasibility is not violated for ≥ constraints (5.3). The second phase is used to assign the values of the variables from zero to one (or) one to zero as long as feasibility is not violated for ≤ constraints in (5.2). Both phases are designed by the dominant principle of constraints (5.2) and (5.3).

The solution obtained from phase one is known as initial feasible solution for ≥ constraints. We update the initial feasible solution in phase two by using dominant principle of intercept matrix of ≤ constraints (5.2). Construct the intercept matrix by dividing bk values by coefficients of (5.2). The elements of intercept matrix are arranged in decreasing order; the leading element is the dominant variable. This process of identifying the leading element from intercept matrix is known as dominance technique. We use this dominant variable to improve the current feasible solution and this procedure provides optimum or near optimum solution of MDMK. The dominant principle focusses at the resource matrix with lower requirement come forward to maximize the profit. The intercept matrix of the constraints (5.2) plays a vital role in achieving the goal, in a heuristic manner. The various stages of DPH are shown below.
5.4.1 DPH Algorithm

Step -1: Initialize the solution by assigning 0 to all $x_j$, called initial solution.
Step - 2: convert initial solution (infeasible) into feasible solution for $\geq$ constraints by assigning 1 to variables (first preference given to larger coefficients).
The corresponding objective value is denoted by $\text{obj}$.
Step - 3: Intercept matrix $D d_{jj} = b_i/a_{ij}$, if $a_{ij} > 0$,
$d_{jj} = M$, a large value; otherwise.
Step - 4: Identify 0 value variables (redundant): If any column has <1 entry in $D$, then the corresponding variable identified as a redundant variable.
Step - 5: Dominant variable: Identify the smallest element (dominant variable) in each column of $D$.
Step - 6: Multiply the smallest elements with the corresponding cost coefficients. If the product is Maximum in kth column and $x_k = 1$ in the current solution, then update the objective function value $f(x_1, x_2, ..., x_n)$. Suppose $x_k = 0$, then set $x_k = 1$ in the current solution and check the feasibility.
Step - 7: Update the constraint matrix: $b_i = b_i - a_{ij}$ for all i and set $a_{ik} = 0$ for all i.
Step - 8: If $a_{ij} = 0$ for all i and j, then go to step-(3). Otherwise display the value of $x'_j's$ and the objective function value.

ALGORITHM 5.1 DPH Algorithm for MDMK.

In the next subsection we present the worst-case bound of DPH and probabilistic analysis of DPH to catch the optimal solution of MDMK.

5.4.2 Worst-case bound

For our convenience, we use $m$ instead of $m+q$. 
Theorem 5.4.1. MDMK with positive constraint data and with optimal solution $x^*$, if $\bar{x}$ is the solution given by the DPH, then \[ \frac{cx - c\bar{x}}{cx - cx^*} \leq \frac{b_{\max}}{b_{\min}} H \left( \max_{1 \leq j \leq n} w_i \right). \]

The basic notations, assumptions, lemmas and proof for the above theorem are presented in section 2.4 of chapter 2.

5.4.3 Probabilistic performance analysis

It is possible to examine the probability that a vector $\bar{x}$ found by DPH reaches the optimal solution to MDMK. Sufficient condition for a solution found by the DPH to be optima can be obtained as follows:

To carryout an analysis, let us specify a probabilistic model of the MDMK. The profit coefficients $c_j$, constrain coefficients $a_{ij}$ and capacities $b_i$ are nonnegative independent identically distributed random variables.

Lemma 5.4.1. A vector $\hat{x}$ found by the DPH defines an optimal solution to MDMK if $\bar{r} - r < \frac{1}{\MAX}$. 

Theorem 5.4.2. Let $P(n)$ be the probability that the vector $x$ found by the DPH defines an optimal solution to the $n$-variable MDMK, and denote $Q(n) = 1 - (B(n, |I|, p_0) + B(n, |I|, p_1))$. Then $P(n) \geq Q(n)$ for all $n \geq |I|$, and $Q(n)$ is a strictly increasing function of $n$, such that $\lim_{n \to \infty} Q(n) = 1$.

Notations and proofs for above lemma and theorem are given in section 2.4 of chapter 2.
5.4.4 Computational Complexity

**Theorem 5.4.3.** DPH can be solved in $O(mn^2)$ time, polynomial in the number of variables and constraints.

**Proof.** The Worst-case complexity of finding the solutions of an MDMK using DPH can be obtained as follows. Assume that there are $n$ variables and $m$ constraints. The procedure initialization (Step-1) and (Step-2) require $O(n)$ and $O(qn)$ running time respectively.

The formation of $D$ matrix iterates $n$ times, identification of less than one entry in each column, finding smallest intercept in each column, identification rows which consists of more than one smallest intercept and updating of constraint matrix $A$. Since there are $m$ constraints, Step-3, Step-4, Step-5, Step-6 respectively, $O(mn)$. Step-8 requires $O(n)$ operations to multiply cost with corresponding smallest intercept and updating the corresponding row of the constraint matrix. The Step-7 requires $O(1)$. The maximum number of iterations required for DPH is $n$. So the overall running time of the procedure DPH can be deduced as $O(mn^2)$. 

5.4.5 Illustration

An example of MDMK solution by DPH.

Maximize: $6x_1+14x_2+11x_3$

Subject to

$13x_1+14x_2+18x_3 \leq 34$

$22x_1+21x_2+28x_3 \leq 53$

$33x_1+39x_2+34x_3 \leq 80$
\[17x_1 + 11x_2 + 16x_3 \geq 22\]

In the above example of MDMK have 3 variables, 3 \(\leq\) constraints, and one \(\geq\) constraint. The algorithm begins with initial solution \((0,0,0)\). We convert this infeasible solution into feasible solution for \(\geq\) constraint by assigning 1(step-2) to the larger coefficients as long as feasibility is not violated i.e., we arrange the coefficients of \(\geq\) constraint in decreasing order \((17, 16, 11)\) and assign 1 to \(x_1\) and \(x_3\). At the end of this stage the solution is \((1, 0, 1)\).

We update this solution by using dominant principle of constraints (6.1.2) iteratively.

Iteration 1: The variable \(x_3\) dominates the other two variables (step-3 to step-6 of the algorithm) and this variable is already in the current solution. The improved objective function value is 11 and solution is \((1, 0, 1)\).

Iteration 2: The variable \(x_2\) dominates the variable \(x_1\) (step-3 to step-6 of the algorithm) and this variable is not available in the current solution. We modify the current solution by assigning 1 to variable \(x_2\) and 0 to \(x_1\).

Iteration 3: There is no updation in the solution(step-8).

Thus the new solution is \(x_1 = 0; x_2 = 1; x_3 = 1\); objective function value = 25 which is the optimum value.

### 5.5 Experimental Analysis

#### 5.5.1 DPH solutions of Test Problems

The heuristics were tested on 810 instances which can be found in [30]. These test instances are divided into 2 sets. The first set consists of positive coefficients in
the objective function and second consists of both positive and negative coefficients. Each set consists of nine classes, each class contains 45 test instances. The MDMK test instances have the following characteristic values.

- $n$, the number of variables: 100, 250 or 500.
- $m$, the number of knapsack constraints: 5, 10 or 30.
- $q$, the demand constraints: 1, $m/2$ or $n$.
- $r$ is 1, if the objective function has negative coefficients; otherwise, 0.
- $s = 0, 1, ..., 14$ is the instance number within the class.

We use $n-m-q-r-s$ for each instances and $n-m-r$ for each class.

The computational results are given in TABLE 5.1. The first column in TABLE 5.1 indicate the name of the problem set. The next two columns report for the CPLEX, the number of optimum solutions and number of infeasible solutions out of 45 problems in each problem set. The last two columns report for DPH that the number of proven optimal solutions found and the the number of failure per class. It is clear that from TABLE 5.1 that our DPH finds the optimal in 344 test problems but CPLEX fails to catch the feasible solution in 72 test problems. The application of DPH algorithm for 100-5-1-0-0 is presented in FIGURE 5.1. The maximum number of iterations is fixed as 100. DPH reaches the solution 30909 optimum at 32nd iteration.

### 5.5.2 Comparison with other Heuristics

The comparative study of DPH with other existing heuristic algorithms (CPLEX:Problem solver, NT:nested tabu search[30], Almha:Adaptive memory search[9]) has been furnished in TABLE 5.2 in terms of the average deviation for problems and the number of optimal solutions obtained for the test instances. It can be seen from TABLE 5.2 that DPH is found to be better than the other
Figure 5.1. Iteration wise Objective function value for 100-5-1-0-0
Table 5.1. Computational results of DPH.

<table>
<thead>
<tr>
<th>class</th>
<th>CPLEX</th>
<th>DPH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opt</td>
<td>Fail</td>
</tr>
<tr>
<td>100-5-0</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>100-5-1</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>100-10-0</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>100-10-1</td>
<td>39</td>
<td>0</td>
</tr>
<tr>
<td>100-30-0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>100-30-1</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>250-5-0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>250-5-1</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>250-10-0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>250-10-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>250-30-0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>250-30-1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>500-5-0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500-5-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500-10-0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500-10-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500-30-0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>500-30-1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>total</td>
<td>207</td>
<td>72</td>
</tr>
</tbody>
</table>

Opt = Optimal solution
Fail = Infeasible solution

heuristics. As both tables and figure clearly demonstrate, the DPH is able to catch the optimum or best solution for all the test problems in quick time. The heuristic is used to reduce the search space to find the near-optimal solutions of the MMKP. The computational complexity is O(nn^2) and the space complexity is O(nn+qn). 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 of the variables instantaneously. The maximum CPU time is 386 seconds for large-size problems. From the comparison table we observe that the proposed algorithm is the effective one.
Table 5.2. Summarized results for the solution quality of MDMK

<table>
<thead>
<tr>
<th>Class</th>
<th>Cplex</th>
<th>Almha</th>
<th>NT</th>
<th>DPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-5-0</td>
<td>0.00</td>
<td>0.04</td>
<td>0.07</td>
<td>0.00</td>
</tr>
<tr>
<td>100-5-1</td>
<td>0.00</td>
<td>0.12</td>
<td>0.26</td>
<td>0.00</td>
</tr>
<tr>
<td>100-10-0</td>
<td>0.13</td>
<td>0.22</td>
<td>0.69</td>
<td>0.62</td>
</tr>
<tr>
<td>100-10-1</td>
<td>0.16</td>
<td>0.30</td>
<td>0.88</td>
<td>1.49</td>
</tr>
<tr>
<td>100-30-0</td>
<td>4.44</td>
<td>4.83</td>
<td>5.92</td>
<td>5.99</td>
</tr>
<tr>
<td>100-30-1</td>
<td>8.92</td>
<td>12.63</td>
<td>10.76</td>
<td>10.91</td>
</tr>
<tr>
<td>250-5-0</td>
<td>0.04</td>
<td>0.09</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>250-5-1</td>
<td>0.20</td>
<td>0.43</td>
<td>1.15</td>
<td>0.44</td>
</tr>
<tr>
<td>250-10-0</td>
<td>0.41</td>
<td>0.58</td>
<td>0.96</td>
<td>0.47</td>
</tr>
<tr>
<td>250-10-1</td>
<td>0.79</td>
<td>1.10</td>
<td>1.80</td>
<td>0.24</td>
</tr>
<tr>
<td>250-30-0</td>
<td>4.38</td>
<td>2.18</td>
<td>2.75</td>
<td>4.83</td>
</tr>
<tr>
<td>250-30-1</td>
<td>13.15</td>
<td>4.56</td>
<td>5.63</td>
<td>14.58</td>
</tr>
<tr>
<td>500-5-0</td>
<td>0.05</td>
<td>0.11</td>
<td>0.25</td>
<td>0.06</td>
</tr>
<tr>
<td>500-5-1</td>
<td>0.17</td>
<td>0.44</td>
<td>0.99</td>
<td>0.06</td>
</tr>
<tr>
<td>500-10-0</td>
<td>0.21</td>
<td>0.39</td>
<td>0.64</td>
<td>7.16</td>
</tr>
<tr>
<td>500-10-1</td>
<td>0.54</td>
<td>0.93</td>
<td>1.50</td>
<td>0.70</td>
</tr>
<tr>
<td>500-30-0</td>
<td>3.56</td>
<td>1.40</td>
<td>1.73</td>
<td>4.46</td>
</tr>
<tr>
<td>500-30-1</td>
<td>7.28</td>
<td>2.35</td>
<td>2.91</td>
<td>8.72</td>
</tr>
<tr>
<td>N.O.O</td>
<td>207</td>
<td>93</td>
<td>61</td>
<td>344</td>
</tr>
</tbody>
</table>

N.O.O = number of optimum solution

5.6 Conclusion

In this chapter, the dominant principle based approach for tackling the NP-Hard Zero-One Multidemand Multidimensional knapsack problem (MDMK). This approach has been tested on 810 state-of-art benchmark instances and has led to given near optimal solutions for 344 instances given in literature. For rest of the instances the average percentage of deviation of DPH solution from the optimum solution is observed to be 3.38. This heuristic is with O(mn^2) complexity and it requires n iterations to solve the MDMK. The performance of this heuristic (TABLE 5.2) has been compared with other heuristics: Tabu Search[30], Adaptive memory search[9]. The experimental data also shows the effectiveness of proposed
algorithm. In the next chapter we design the gravitational emulation search based meta heuristic to solve symmetric traveling salesman problem.