Hybrid Method for Binary Multi-Objective Multiconstaint Knapsack Problems

Chahrazad Adiche[1,2] and Méziane Aïder[2]

[1] UMBBoumerdes, Fac. Sciences, Dep. Mathematics, Algeria
[2] USTHB, Fac. Mathematics, LaROMaD, BP 32 El Alia, 16111 Algiers, Algeria
adichechahra@yahoo.fr, m-aider@usthb.dz

1 Introduction

The multi-objective multiconstraint knapsack problem can be expressed by the following mathematical programming:

$$(MOMCKP)\begin{cases} \text{``}\max\text{''}\ Z^k(x) = \displaystyle\sum_{j=1}^{n} c_j^k x_j & k = 1,\dots,p \\ \displaystyle\sum_{j=1}^{n} \lambda_j^i x_j \le \mu_i & i = 1,\dots,m \\ x_j \in \{0,1\} & j = 1,\dots,n \end{cases}$$

where n is the number of items, m the number of knapsack constraints and c_j^k is the value of the item j $(j = 1,\dots,n)$ for the criterion k $(k = 1..p)$. All the parameters are assumed to be positive integers. The problem is obviously NP-hard, since its mono-objective version is.

In Pareto optimization, the aim is to find the set of "efficient" solutions in an exact or in an approximate way. Exact methods seek to solve a problem to guarantee optimality but their execution on large real world problems usually requires too much computation time. The branching sequence has a great impact on the convergence of the branch and bound approach.

This work presents a new hybrid method to solve binary multi-objective multiconstraint knapsack problems. The main idea of the proposed hybridization is to incorporate an heuristic method based on a fuzzy dominance relation into a multi-objective branch and bound scheme. The strength of this hybridization is in quickly determining a set of "efficient" solutions.

2 Multi-objective Multiconstraint Knapsack and Branch & Bound

We propose one adaptation of the branch-and-bound method dedicated to the multi-objective knapsack problem type in $0-1$ [1], to $(MOMCKP)$. The addition of constraints of the considered problem involves too many possible combinations to evaluate and so, causes new levels of complication in the solution process.

In the branch-and-bound scheme, the solution space is explored by dynamically building a tree and by using the following three basic procedures: separation, evaluation and sterilization. Partial solutions (nodes of the search tree) are created by assigning zeros and ones to subsets of items denoted B_0 and B_1, respectively. Items not yet assigned (neither zero nor one) define the set $\mathcal{F} \subseteq \{1,\dots,n\}$ of the *free items* and we then have $\{1,\dots,n\} = B_0 \cup B_1 \cup \mathcal{F}$.

The branching sequence is crucial for the performance of the method. Let θ be the order according to which variables (items) of a partial solution will be assigned a value. The order θ can be defined as in Florios et al. [2] according to the increasing values of the following heuristics rules $(1)-(2)$:

$$Ave\_sort_j = \frac{1}{p.m}\sum_{k=1}^{p}\sum_{i=1}^{m}\frac{c_j^k}{\lambda_j^i}, \quad j = 1,\dots,n. \tag{1}$$

$$max_j = \max_{k=1,\dots,p;\ i=1,\dots,m}\frac{c_j^k}{\lambda_j^i}, \quad j = 1,\dots,n. \tag{2}$$

Assume that the items $a_1, a_2, \ldots, a_n$ are labeled in a decreasing order according to one of the rules $(1)-(2)$.

The branch-and-bound algorithm starts by fixing many items according to the θ order to quickly find a good feasible solution. Thus, many branches of the tree can be pruned early. The list $\mathcal{N}$ of nodes is maintained as a $LIFO$ stack (Last In First Out). When a node is pruned, the algorithm backtracks and creates a new node by moving the last item ($t < n$) in B_1 to B_0. In addition, all items in B_0 after this new item become free ($\mathcal{F} \leftarrow \{t+1, \ldots, n\}$). If, however, n was the last item in B_1 ($t = n$), let be u the smallest index such that $\{u, u+1, \ldots, t-1, t\} \subset \beta_1$ and s the last index of $\beta_1 \setminus \{u, \ldots, t\}$. Then, the algorithm removes all items $\{s, \ldots, n\}$ in B_1 and defines B_0 to be all previous elements of B_0 up to $s-1$ and to include s. Furthermore, all items after the $s-th$ item become free ($\mathcal{F} \leftarrow \{s+1, \ldots, n\}$). When a node is not pruned, the algorithm progresses deeper down the tree and creates a new successor node. Indeed, as many items as possible are included in B_1, according to order θ, i.e. as they appear in $\mathcal{F}$. But if the remaining vector $\overline{\mu}$ does not allow item l to be added to B_1, the first possible item r of $\mathcal{F}$, which can be added to B_1 is sought and item r is added to B_1. Of course, all items $\{i, \ldots, r-1\}$ must be added to B_0.

3 Heuristic rule based on fuzzy dominance relation

Let $\widehat{E} = \{a_1, a_2, \ldots, a_n\}$ be a set of items. The vector $(U^1(a_i), U^2(a_i), \ldots, U^J(a_i))$, where $U^j(a_i) = U(a_i, \pi^j) = \sum_{k=1}^{m} \pi_k^j c^k(a_i), j = 1..J$, represent the multiple utilities of the item a_i according to the various randomly generated weight vectors $\pi^1, \pi^2, \ldots, \pi^J$, $(\pi^j = (\pi_1^j, \ldots, \pi_k^j, \ldots, \pi_p^j))$.

The credibility of the proposition "a_i is at least as good as a_h" is computed by the following fuzzy dominance relation on $\widehat{E} \times \widehat{E}$:

$$\mu_D(a_i, a_h) = \max(P_U(a_i, a_h) - P_U(a_h, a_i), 0),$$

where $P_U(a_i, a_h)$ represents the proportion of utility for which a_h is not preferred to a_i and is defined by:

$$P_U(a_i, a_h) = \begin{cases} \frac{|\{j, U^j(a_i) \geq U^j(a_h)\}|}{J}, & \text{if } \nexists j^0 \text{ such that } U^{j^0}(a_i) + v < U^{j^0}(a_h); \\ 0, & \text{otherwise,} \end{cases}$$

and v is a threshold of veto (for example: $v = 0.2$).

If for one weight vector j^0, the difference between $U^{j^0}(a_i)$ and $U^{j^0}(a_h)$ is too unfavorable to a_i, then we refuse any credibility to the upgrade of a_h by a_i whatever are the performances of these two items for the other weight vectors.

For a fixed item a_i, $\mu_D(a_i, a_h)$ represents the fuzzy subset of items a_h dominated by a_i. Its complementary, defined by the membership function $1 - \mu_D(a_i, a_h)$, is the fuzzy subset of items non-dominated by a_i. The intersection of all the fuzzy subsets of the items non-dominated by a_i, when a_i goes through $\widehat{E}$ gives the subset of the items that are dominated by no other one. The corresponding membership function is defined by:

$$\mu^{ND}(a_h) = \inf[1 - \mu_D(a_i, a_h), a_i \in \widehat{E}] = 1 - \sup[\mu_D(a_i, a_h), a_i \in \widehat{E}].$$

$\mu^{ND}(a_h)$ can be interpreted as the degree of truth of the assertion: "a_h is dominated by no item in $\widehat{E}$".

When we look for the best items, it is thus logical to choose the one for which the value of μ^{ND} is closest to 1. To obtain a complete ranking, θ, of items, it is necessary to proceed by successive steps, by eliminating the items already ranked and by recomputing μ^{ND} at every time.

References

1. Ehrgott, M.: Multicriteria Optimisation. Springer, march (2005).
2. Florios, K. ; Mavrotas, G. ; Diakoulaki D.: Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. European Journal of Operational Research, 14–21 (2010).