Simulated Annealing Algorithm for the Multiple Choice Multidimensional Knapsack Problem

Shalin Shah
sshah100@jhu.edu

Abstract

The multiple choice multidimensional knapsack problem (MCMK) is a harder version of the 0/1 knapsack problem, and is ever more complex than the 0/1 multidimensional knapsack problem. In MCMK, there are several groups of items. The objective is to maximize the value (profit) by choosing exactly 1 item from each group such that all the constraints are satisfied. It is difficult and NP-hard even to find a solution that does not violate all constraints. In this work, we present a simulated annealing based algorithm with open source C++ code to find good solutions to the multidimensional multiple choice knapsack problem. In all of the benchmark instances we used, the algorithm is able to find optimum (or close) solutions, thereby proving that the algorithm is suitable for solving larger instances for which optimal solutions are unknown.

Keywords: multiple choice multidimensional knapsack problem, simulated annealing, greedy algorithm, C++ code, open source implementation, fast heuristic, randomized algorithm

1 Introduction

The multiple choice multidimensional knapsack (MCMK) problem has m constraints and n groups. Each group contains m_i of items. Each item has a value (profit) associated with it. The goal is to choose one item from each group such that all m constraints are satisfied and the objective is maximized. Even finding a feasible solution to the MCMK problem is NP-hard. In this paper, we present a simulated annealing [1] algorithm with open source C++ code that is able to solve all benchmark instances to optimality (or close).

The MCMK can be described by the following equations:

$$\begin{aligned} \text{Maximize: } & \sum_{i=1}^n \sum_{j=1}^{m_i} x_{ij} v_{ij} \\ \text{Such that: } & \sum_{i=1}^n \sum_{j=1}^{m_i} x_{ij} w_{ij}^k \leq W^k \end{aligned}$$

Where: $\sum_{i=1}^n \sum_{j=1}^{m_i} x_{ij} = n$

i.e. 1 item is chosen from each group.

And $x_{ij} \in \{0, 1\}$

Where, n is the number of groups, G_i is the i^{th} group, there are m_i items in each group, x_{ij} is binary (0/1) indicating the choice of the j^{th} item in the i^{th} group, v_{ij} is the value (profit) of the j^{th} item in the i^{th} group, W^k is a set of k constraints, w_{ij}^k is the weight (constraint) of the k^{th} constraint of the j^{th} item in the i^{th} group.

The 0/1 multidimensional knapsack problem (MKP) [2] [3] is a related problem in which there is only one group and there is no limit to the number of objects chosen. This makes it considerably easier to solve and a simple linear time greedy algorithm exists for the MKP based on a utility ratio. [3] uses Lagrangian multipliers as constraint weights for the MKP with good results.

The work in [4] presents three heuristics for the multiple choice multidimensional knapsack problem, one of them is called the guided local search, with good results. They use the same benchmark instances as we do, which were first described in [5]. They are called *I01, I02....*

The literature on the MCMK problem is limited. [6] provide a heuristic based on surrogate constraints. The work in [7] is based on a genetic algorithm, though the results are based on limited number of benchmarks.

Our algorithm is open source and the code is provided in [8]:

<https://github.com/shah314/samultichoiceknapsack>

2 Methods

Our algorithm is based on simulated annealing with a linear annealing schedule. We accept worse solutions with a probability that is reduced as the iterations progress. We use 10000 iterations (which can be changed in the global variables).

We first start with a greedy solution that is based on a utility ratio (based on an aggregate of all constraints for each item in all groups). The greedy solution might not be a feasible solution, so we use a randomized greedy method to make the solution feasible. We then run simulated annealing on the greedy estimate.

The simulated annealing algorithm is based on 1-opt moves. We randomly choose a group and then choose a random item in that group. We then change

Table 1: Results obtained by running the MCMK algorithm on Ixx instances first described in [5]

Instance	Best Known	This Algorithm
I01	173	173
I02	364	361
I03	1602	1595
I04	3597	3542
I05	3905	3900
I06	4799	4792
I07	23912	24225
I08	35979	36283
I09	47901	48308
I10	59811	60419
I11	71760	72494
I12	84141	84615
I13	96003	96791

the choice function for that group to choose the randomly chosen item instead of the current choice. This may result in an infeasible solution. We then correct the solution using randomized greedy moves till the solution is feasible. It is possible that no feasible solution exists for an instance of the multiple choice multidimensional knapsack problem in which case we quit after a certain number of iterations of trying to make the solution feasible.

3 Results

We run our algorithm on the *I01, I01, ...* benchmark instances first described in [5] and also used in [4]. We find that our open source solver is able to find optimal or close to optimal solutions to all of these instances. We provide the instances in our git repository at [9].

Results are shown in table 1.

4 Conclusion

In this paper, we presented a simulated annealing algorithm for the multiple choice multidimensional knapsack problem. This problem is significantly more difficult than the 0/1 knapsack problem, or even the 0/1 multidimensional knapsack problem. Finding feasible solutions is also NP-hard. Our algorithm begins with a greedy feasible solution and then improves upon it based on 1-opt moves. Feasibility correction happens based on a randomized greedy algorithm which

takes an infeasible solution and corrects it such that the solution satisfies all constraints. Results show that our algorithm performs quite good on the benchmark instances (some with known optimal solutions, some with only bounds). This indicates that our algorithm is likely to find good solutions to very large instances for which optimum solutions or even bounds are not available. Future work could perform some experiments with the annealing schedule to see if other annealing schedules are able to improve upon the run-time of the algorithm.

References

- [1] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- [2] Shalin Shah. Genetic algorithm for the 0/1 multidimensional knapsack problem. *arXiv preprint arXiv:1908.08022*, 2019.
- [3] Shalin Shah. Gknap: A java and c++ package for solving the multidimensional knapsack problem. *Journal of Open Source Software*, 4(42):1756, 2019.
- [4] Mhand Hifi, Mustapha Michrafy, and Abdelkader Sbihi. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, 55(12):1323–1332, 2004.
- [5] Shahadat Khan, Kin F Li, Eric G Manning, and Md Mostofa Akbar. Solving the knapsack problem for adaptive multimedia systems. *Stud. Inform. Univ.*, 2(1):157–178, 2002.
- [6] Skander Htiouech, Sadok Bouamama, and Rabeh Attia. Osc: solving the multidimensional multi-choice knapsack problem with tight strategic oscillation using surrogate constraints. *International Journal of Computer Applications*, 73(13):1–7, 2013.
- [7] Solving multidimensional multiple choice knapsack problem by genetic algorithm & measuring its performance.
- [8] <https://github.com/shah314/samultichoiceknapsack>.
- [9] <https://github.com/shah314/samultichoiceknapsack/tree/master/instances>.