The 7th International Symposium on Operations Research and Its Applications (ISORA'08)
Lijiang, China, October 31–Novemver 3, 2008
Copyright © 2008 ORSC & APORC, pp. 369–376

A Perturbation-Guided Oscillation Strategy for the Multidimensional Knapsack Problem with Generalized Upper Bound Constraints

Vincent Chiwei Li

Institute of Global Operations Strategy and Logistics Management
National Dong Hwa University
Hualien, Taiwan 974

Abstract The multidimensional knapsack problem with generalized upper bound constraints (referred as GUBMKP) can be seen as a special case of the multidimensional knapsack problem where variables are partitioned into several multiple-choice sets and at most one variable can be chosen from each set. To solve the GUBMKP, critical event tabu search oscillates around feasibility boundary. Critical solutions are collected upon crossing the boundary. Trial solutions can be explored around critical solutions. Within this critical event tabu search framework, we develop a perturbation-guided tabu search strategy where the constraint right-hand-sides will be perturbed and restored iteratively throughout the trial solution search at critical events. The strategy considers not only the change of objective function value but also the change of feasibility. Our computational runs show the approach is capable to obtain good solutions within a short period of time.

Keywords perturbation oscillation, multidimensional knapsack problem, generalized upper bound

1 Introduction

In this paper, we consider a multidimensional knapsack problems with generalized upper bound (GUB) constraints stated as below: A set of n items ($N = \{1, \cdots, n\}$) should be packed to maximize the overall profit (item j can contribute c_j) such that the capacity limitation ($b_i, i \in M = \{1, \cdots, m\}$) of each of the m knapsack constraints will be satisfied where selecting item j will consume a_{ij} units of resource i. All the items are partitioned into g subsets and at most one item can be selected from each of these subsets. This problem (referred to as GUBMKP) can be stated as follows.

Formulation 1 (GUBMKP).

$$\begin{aligned}
&\textit{maximize} && \sum_{j\in N} c_j x_j && (1)\\
&\textit{subject to} && \sum_{j\in N} a_{ij}x_j \le b_i, \forall i \in M \equiv \{1,\cdots,m\}, && (2)\\
& && \sum_{j\in N_k} x_j \le 1, \quad \forall k \in G \equiv \{1,\cdots,g\}, && (3)\\
& && x_j \in \{0,1\}, \quad \forall j \in N \equiv \{1,\cdots,n\}, && (4)\\
&\textit{where} && \bigcup_{k=1}^{g} N_k = N, &&\\
& && N_p \cap N_q = \emptyset \quad \forall p,q \in G, p \neq q, &&\\
& && a_{ij}, c_j, b_i \ge 0 \ \ \forall i \in M, j \in N. &&
\end{aligned}$$

Notice that $a_{ij} \le b_i$ and $b_i \le \sum_{j\in N} a_{ij}$.

This paper develops a perturbation-guided oscillation strategy in the framework of critical event tabu search for the GUBMKP. Computational runs were conducted to test the capability of this heuristic.

2 Literature Review

The GUBMKP is a special case of the multidimensional knapsack problems (MKP) as each of the GUB constraints can be treated as a knapsack constraint. The MKP is NP-hard in the strong sense [11] as is the GUBMKP. Many approaches have been applied to the MKP such as exact algorithms, bound based heuristics, greedy heuristic methods, and approximate dynamic programming. A substantial literature for the MKP utilizes surrogate constraints and Lagrange relaxation. Detailed reviews on MKP can be found in [2, 1].

The GUBMKP is also closely related to the multidimensional multiple-choice knapsack problem (MMCKP). The MMCKP can be formulated by substituting the inequality ("$\le$") by the strict equality ("$=$") for Equation 3 in the GUBMKP formulation. By adding a slack variable in each of the GUB constraint, the GUBMKP can be converted to the MMCKP. For a recent review on MMCKP, readers are referred to [7].

Traditional ascent or descent methods often become trapped in local optima when dealing with large-scale problems. Metaheuristics usually provide excellent feasible solutions quite quickly for many difficult combinatorial problems. Metaheuristics appear in various forms, operating either on individual solutions such as tabu search, simulated annealing or on populations of solutions such as genetic algorithms, scatter search and path relinking, just to name a few.

The critical event tabu search method for the MKP was first proposed by [6]. The critical event tabu search method for the MKP in [6] focuses on a flexible memory structure revised at critical events. Critical events tabu search consists of a constructive phase and a destructive phase. The constructive phase corresponds to the process of moving from a feasible solution to an infeasible solution while the destructive phase corresponds to the process of moving from an infeasible solution to a feasible solution. Critical events refer to solutions generated by the constructive phase at the final moment before becoming infeasible and solutions generated by the destructive phase at the first moment after becoming feasible. The search alternates between the constructive phase and the destructive phase with different depth of span controlled systematically by the heuristic. The search process navigates around the feasibility boundary and avoids duplicating critical solutions by using the recency and frequency information of the critical solutions.

[9] modified the critical event tabu search method for the GUBMKP. The GUB constraints are always satisfied in the search process. They demonstrated the merits of using surrogate constraint information versus a Lagrangian relaxation scheme as choice rules to determine which variables to add, drop or swap. Furthermore, constraint normalization proposed by [3] and [5] was applied to strengthen the surrogate constraints. The search at critical events is explored further in [8] by a tight oscillation approach by considering the objective function change as well as the feasibility change in its choice rules.

3 Methodology

3.1 Critical Event Tabu Search for the GUBMKP

[9] applied the critical event tabu search to solve the GUBMKP. This method oscillates around feasibility boundary with constructive and destructive phases. Solutions obtained immediately before or after crossing the boundary are referred to critical solutions. Even after crossing the feasibility boundary, more moves are generated in the same direction in order to diversify the search. A systematic span parameter helps to reach different depth of oscillation around the feasibility boundary.

3.2 Solving the GUBMKP using Perturbation-Guided Strategic Oscillations

Basic trial solutions are generated in [9] at critical events by considering the change of the objective function value in evaluating a move. Tight oscillation, a more advanced approach for finding trial solutions is explored in [8], which considers both the feasibility change and the objective value change in evaluating a move. In this paper, we use a perturbation scheme to explore trial solutions at critical events.

Consider an "infeasible variant" which utilizes a perturbation-guided strategic oscillation [4]. Imagine each of the right hand side value of the knapsack constraints is increased by an amount δ_i. For example, δ_i can be the minimum of the coefficients of the i^{th} constraint or some multiple of the corresponding minimum (a multiple of 2 is selected in our implementation). Then we can apply a local improvement approach to this perturbed problem. Upon terminating this method, we need to seek a feasible solution by considering the value of infeasibility worsening and infeasibility improvement described as follows.

With this in mind, when considering the addition of x_j, the value of this move is calculated as

$$c_j - k\,\overline{c}\ \text{(infeasibility worsening by adding } x_j\text{)} \tag{5}$$

where c_j is the objective coefficient of x_j, $\overline{c}$ is the average value of all objective coefficients, and k is a chosen constant to adjust the weight put on "infeasibility worsening".

The term “infeasibility worsening” refers to the amount by which a move causes constraints to become more violated, summed over all the constraints. Similarly, when considering dropping x_i, the value of this move is calculated as

$$-c_i + k\,\bar{c}\ (\text{infeasibility improvement by dropping } x_i). \tag{6}$$

The term “infeasibility improvement” refers to the amount by which a move causes constraints to become less violated, summed over all the constraints. In addition, when considering swapping x_j (0 now) and x_i (1 now) in the same GUB set, the value of this move is calculated as

$$\begin{split} c_j - c_i + k\,\bar{c}\ (\text{infeasibility improvement by dropping } x_i) - \\ k\,\bar{c}\ (\text{infeasibility worsening by adding } x_j). \end{split} \tag{7}$$

As mentioned earlier, when all GUB constraints are tight but the solution is still feasible, swap moves are required to help the process cross the feasibility boundary. The following steps are used for implementing the corresponding choice rules.

Step 1: Define $b_i^{'}$ as the RHS value for constraint i after assigning the current values to the variables. Let

$$u_i = \min\{0, b_i^{'}\}.$$

Thus u_i is negative and equals to $b_i^{'}$ only if the current solution is infeasible to constraint i; u_i is 0 otherwise.

Step 2: Consider the feasibility change on each constraint for a particular move (add, drop or swap). For example, if x_k (currently 1) is set to 0 while x_j (currently 0) is set to 1 (x_k and x_j are in the same GUB set), then let

$$v_i = \min\{0, b_i^{'} + a_{ik} - a_{ij}\}.$$

So v_i is negative and equals to $b_i^{'} + a_{ik} - a_{ij}$ only if the proposed solution is infeasible to constraint i; v_i is 0 otherwise.

Step 3: Let z_i be the infeasibility difference for constraint i, that is,

$$z_i = v_i - u_i.$$

If $z_i < 0$, then the infeasibility of constraint i is worsened by the amount $|z_i|$, while if $z_i > 0$, then the infeasibility of constraint i is improved by the amount z_i.

Step 4: Let “netchange” be the sum over the infeasibility change for each knapsack constraint. Thus,

$$\text{netchange} = \sum_{i \in M} z_i.$$

If netchange is negative, then the overall infeasibility is worsened, while if netchange is positive, then the overall infeasibility is improved.

Step 5: Finally calculate the weighted sum of the infeasibility change and the objective function change by multiplying the infeasibility change by a multiple of the average of all objective coefficients.

Notice that when evaluating which variables to drop (to recover feasibility), attention should be restricted to moves with positive infeasibility changes (infeasibility improving). When evaluating what variables to add, the infeasibility change is always negative or 0, and this term penalizes the gain contributed by the difference of the objective function values. When summing over infeasibility changes from all constraints, it is assumed that all constraints have been normalized by their RHS values (divided by the RHS value of each of the knapsack constraints, respectively). Other normalization schemes are also possible, such as the one used in generating a strong surrogate constraint approach by [9].

Once the solution becomes feasible (to the unperturbed problem), we can use local improvement method without allowing further infeasibility. When this method stops at a local optimum, we can choose to do another cycle. In our implementation, we limit ourselves to do another cycle only if the perturbation has improved the best solution ever found.

4 Computational Results

4.1 Test Problems

There are three original random test problems in [10]. Each of them is categorized according to the number of knapsack constraints and the number of variables: Problems 1, 2, and 3 consist of 10, 20, and 30 knapsack constraints, respectively. On the other hand, Problems 1, 2, and 3 consist of 20, 40, and 60 GUB constraints, respectively. Problem instances with 33% and 66% of the original available variables (randomly chosen) are generated and included in our test set. The dimensions (numbers of variables and numbers of constraints) of these nine instances are summarized in Table 1. This table shows that Problem 1 has ten knapsack constraints and twenty GUB sets and each of the instances of Problem 1 has 30 constraints with various numbers of variables. For example, Instance P1_33 (33% of variables chosen from Problem 1) has 1,328 binary variables while Instance P1_100 is the same as Problem 1.

Table 1: Nine Problem Instances for the GUBMKP

Problem Number	1		2		3	
# of knapsacks (m)	10		20		30	
# of GUB sets (g)	20		40		60	
Percentage w.r.t. Original Problem	Instance Name	# of Vars.	Instance Name	# of Vars.	Instance Name	# of Vars.
33%	P1_33	1328	P2_33	2765	P3_33	5102
66%	P1_66	2645	P2_66	5511	P3_66	10174
100%	P1_100	3994	P2_100	8321	P3_100	15365

4.2 Test Results

In each problem instance, the gap between Z_{LP}, the optimal solution of the LP relaxation and Z_H, the final integer solution obtained from the heuristic, is calculated as

$$\frac{Z_{\text{LP}} - Z_H}{Z_{\text{LP}}}. \tag{8}$$

This value indicates how close the solution obtained from the heuristic is to the optimal solution of the LP relaxation. The results from CPLEX MIP Solver indicate the LP relaxation is a very good approximation to the optimal solution as the close gaps (default value is 0.01%) have been reached in the branch-and-bound optimization for several instances. The close gap in CPLEX is calculated as

$$\frac{Z_{\text{Best Bound}} - Z_{\text{MIP}}}{Z_{\text{Best Bound}}}. \tag{9}$$

Consequently, we can use the LP relaxation optimal solution to estimate the optimal solution. This justifies our use of Equation 8.

To compare the results of the perturbation-guided strategy with other methods in the literature, various approaches are implemented and run at a PC of pentium 4 processor at 3.0 GHz with 1 GB of RAM. The computational results for these nine problem instances are summarized in Table 2 for a run time limit of 1 minute Gaps for six different approaches are listed from column 2 to column 7 in this table. The notation can be described as follows.

Gap(0): Gap from CPLEX MIP Solver
Gap(1): Gap from the critical event tabu search without any trial solutions
Gap(2): Gap from the critical event tabu search with basic trial solutions
Gap(3): Gap from the critical event tabu search with tight oscillation strategy
Gap(4): Gap from the perturbation-guided oscillation with a “Pre_Add” procedure
Gap(5): Gap from the perturbation-guided oscillation without a “Pre_Add” procedure

For examples, the second column of Table 2 shows the closing gap (evaluated by Equation 8) of each of the instances run by CPLEX MIP solver. The 1-minute closing gaps of Instance P1_33 by CPLEX and tight oscillation strategy are 1.87% and 3.67%, respectively. Notice CPLEX gets optimal solution for Instances P1_33, P1_66 and P1_100 (The value of Equation 9 $\leq$ 0.01%.) when run time limit is set to one minute. However, CPLEX runs out of memory for the other six instances when we try to tun them for one hour (the corresponding results are not shown here).

The results show that basic trial solution (Gap(2)) significantly improves the solution of the plain critical event tabu search (Gap(1)). This illustrates the benefit of searching for trial solutions. Moreover, the tight oscillation strategy helps intensify the search process more thoroughly after good trial solutions have been found as we can see Gap(3) is much less than Gap(2) in all the 27 cases. The perturbation with a preliminary add function (Gap(4)) also performs better than the basic trial solution approach in all the instances except for Instance P1_33 at one-minute limit. Both Columns 5 and 6 shows the results of perturbation. The difference between them is as follows. After restoring the feasibility during perturbation, we will proceed to do local search by swap moves to improve the solution. Before proceeding to this step, we can choose to keep adding variables until it is impossible to accomplish that. Consequently, each of the knapsack constraints becomes much tighter before we start to swap. The results of taking this option turns out to be better than the one without using this step in general. Notice Gap(4) is less than or equal to Gap(5) except for two cases (out of 27 cases). We also observe that tight oscillation (Gap(3)) performs better than the perturbation process (Gap(4) and Gap(5)) in general.

However, there is still more room to improve the perturbation as we will discuss in the next section. Last but not least, the tight oscillation and perturbation methods have the potential to get solutions with high quality in a very short time frame (e.g., less than 7% and 6% of gaps within one minute for perturbation and for tight oscillation, respectively) as shown in Table 2.

Table 2: Comparison among Different Approaches of 1-Minute Run Time Limit

Problem Instance	Gap(0) (%)	Gap(1) (%)	Gap(2) (%)	Gap(3) (%)	Gap(4) (%)	Gap(5) (%)
P1_33	1.87*	10.91	2.86	2.10	3.93	3.67
P1_66	1.24*	10.05	3.30	2.33	3.17	3.30
P1_100	1.39*	12.39	1.98	1.85	1.85	1.98
P2_33	2.11	17.55	10.99	5.36	5.59	3.93
P2_66	2.01	16.98	10.09	4.25	5.42	7.18
P2_100	1.98	18.97	11.81	3.53	5.90	8.90
P3_33	1.52	19.31	13.83	3.88	6.68	9.35
P3_66	1.58	19.24	12.91	3.07	3.19	6.59
P3_100	1.75	18.66	14.00	3.72	3.74	5.83

*: *Optimal* solution found (i.e., The gap value of Equation 9 $\leq 0.01\%$.)

5 Conclusions and Future Research Direction

In this paper, we discuss a perturbation-guided oscillation strategy on the GUBMKP. With a framework of the critical event tabu search, we implement other three methods and compare the results among these alternatives. We show that having good strategies of intensification in the search process is very crucial in addition to good diversification. The critical event tabu search oscillates around the feasibility boundary. It is very important not to duplicate these critical solutions. Since a critical solution is very close to the boundary, it is very natural to start a local search in its neighborhood. To obtain good trial solutions, the choice rule has to simultaneously capture important perspectives such as the objective function value change and feasibility change of a move. We demonstrate an efficient method to perturb a problem, restore the feasibility, and improve the solution through a local swap and a preliminary add strategy. Our methods show the capability of obtaining good solutions in a very short time frame.

There is still more room to improve the perturbation heuristic. As we have experimented different perturbation rates, we found a value of 1.5 performs better than 2.0 in general. It will be interesting to see the impact when the perturbation rate is adjusted dynamically. Within different stages of the perturbation process, how to set up the choice rules to conduct local swaps is also an interesting issue.

It might also be helpful to consider the tabu memory thoroughly within the perturbation process so that the search can avoid generating too much duplication. To lessen the burden of memory, the current setting doesn't use tabu memory in this process except for the local swap function. It will be of interest to see the trade-off of this modification.

References

[1] D. Bertsimas and R. Demir. An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science*, 48(4):550–565, April 2002.

[2] P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.

[3] F Glover. Surrogate constraints: Tutorial notes, October 1998.

[4] F. Glover. Personal Correspondence, 2000.

[5] F. Glover. Tutorial on surrogate constraint approaches for optimization in graphs. *Journal of Heuristics*, 9(3):175–227, June 2003.

[6] F. Glover and G.A. Kochenberger. Critical event tabu search for multidimensional knapsack problems. In I. H. Osman and J. P. Kelly, editors, *Metaheuristics: Theory and Applications*, pages 407–427. Kluwer Academic Publishers, Boston, 1996.

[7] C.S. Hiremath and R.R. Hill. New greedy heuristics for the multiple-choice multi-dimensional knapsack problem. *Int. J. Operational Research*, 2(4):495–512, 2007.

[8] V.C. Li. Tight oscillations tabu search for multidimensional knapsack problems with generalized upper bound constraints. *Computers and Operations Research*, 32(11):2843–2852, 2005.

[9] V.C. Li and G.L. Curry. Solving multidimensional knapsack problems with generalized upper bound constraints using critical event tabu search. *Computers and Operations Research.*, 32(4):825–848, 2005.

[10] V.C. Li, G.L. Curry, and E.A. Boyd. Towards the real time solution of strike force asset allocation problems. *Computers and Operations Research*, 31(2):273–291, 2004.

[11] S. Martello and P. Toth. *Knapsack Problems*. Wiley, New York, 1990.