A Steepest Decent Multiplier Adjustment Method for the Generalized Assignment Problem

Nejat Karabakal
James C. Bean
Department of Industrial & Operations Engineering
University of Michigan   Ann Arbor, MI 48109

Jack R. Lohmann
School of Industrail and Systems Engineering
Georgia Institute of Techology Atlanta, GA 30332

Technical Report 92-11
January 1992
Revised October 1993

A Steepest Descent Multiplier Adjustment Method for the Generalized Assignment Problem

Nejat Karabakal
Faculty of Business Administration
Bilkent University, 06533 Ankara, Turkey

James C. Bean
Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, Michigan 48109-2117

Jack R. Lohmann
School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332

January 1992
(Revised October, 1993)

Abstract

Multiplier adjustment methods (MAMs) have been used to solve the Lagrangian dual of the generalized assignment problem. We improve the traditional MAM by incorporating a post-optimality analysis for the 0-1 knapsack subproblems based on a dynamic programming formulation. Our approach guarantees a steepest descent improvement at each iteration and eliminates all intermediate knapsack solutions required by the traditional MAM for calculating a step length. We also describe a branch-and-bound algorithm that incorporates the improved MAM as a bounding tool. Computational results obtained with the algorithm are reported.

1 Introduction

The generalized assignment problem (GAP) seeks to optimally assign n tasks to m agents such that each task is assigned to exactly one agent but each agent is constrained only by the amount of a resource. Let c_{ij} be the profit of assigning task j to agent i, a_{ij} be the resource required by agent i to perform task j, and b_i be the resource available to agent i. Suppose a 0-1 variable, x_{ij}, is set to one if task j is assigned to agent i, zero otherwise. Then, an integer programming formulation for the GAP is:

$$\text{maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\,x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}\,x_{ij} \le b_i \qquad i = 1,\ldots,m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1,\ldots,n \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \text{all } i,j. \tag{4}$$

Aside from many practical problems that can be put into the above formulation directly, other applications can be reformulated to the GAP structure. Ross and Soland [17] show how to formulate a number of important facility location and location-allocation problems as GAPs. In more sophisticated applications, the GAP appears as a subproblem. Fisher and Jaikumar [4] describe a two-phase vehicle routing heuristic where, in the first phase, a GAP is solved to assign customers to vehicles and, in the second phase, traveling salesman heuristics are invoked to determine individual vehicle routes. The shopping mall design project in Bean *et al.* [2] could be formulated as a GAP. In that problem, $x_{ij} = 1$ if store j is located in area i of the mall.

Both GAP and finding a feasible solution to (2)-(4) are NP-hard problems ([5] and [15]). A common solution technique for the GAP has been branch-and-bound with upper bounds obtained by various relaxations. The straightforward linear programming relaxation, given by replacing (4) with $x_{ij} \ge 0$, for all i and j, is rarely used in the literature since it fails to exploit the structure and results in large duality gaps. Ross and Soland [16] used a relaxation replacing (2) with $a_{ij}x_{ij} \le b_i$, for all i and j. Martello and Toth [14] proposed another relaxation by simply removing multiple-choice constraints (3). They reported improved performance over the Ross-and-Soland algorithm for problems where knapsack constraints (3) are tight. Later, several Lagrangian relaxation approaches were proposed for the GAP. Chalmet and Gelders [3] compared the effects of relaxing either (2) or (3) with respect to duality gaps and computational times. A subgradient optimization is used in the search process. Jörnsten and Näsberg [9] used a variable-splitting (Lagrangian decomposition) approach in which two subproblems are generated, one relaxing (2) and the other relaxing (3). Fisher, Jaikumar, and Van Wassenhove [5] obtained a Lagrangian relaxation by dualizing the multiple-choice constraints (3). Instead of using the more common subgradient optimization to solve the Lagrangian dual, they developed a *multiplier adjustment method* (MAM) for determining good multipliers. MAMs are heuristic algorithms for solving Lagrangian duals exploiting the special structure of a particular problem. They also reported significant improvements over Ross-and-Soland and Martello-and-Toth algorithms in their computational tests. Later, Guignard and Rosenwein [7] suggested several enhancements over this MAM but the basic structure remained the same. We will refer to Fisher-Jaikumar-Van Wassenhove version of the MAM that incorporates some of the enhancements of Guignard and Rosenwein as the *traditional MAM* in subsequent discussions.

To construct the Lagrangian relaxation that dualizes (3), let u_j be the multiplier associated with multiple-choice constraint j. Then

$$L(u) = \sum_j u_j + \max_x \left\{ \sum_i \sum_j (c_{ij} - u_j) x_{ij} \mid x \text{ satisfies (2) and (4)} \right\}. \quad (5)$$

For a given u, $L(u)$ is an upper bound on the optimum GAP objective. The tightest bound is obtained by solving the Lagrangian dual problem: $\min_u L(u)$. Although it cannot guarantee bounds better than those obtained by a subgradient algorithm, the MAM usually provides faster bound improvements in solving the Lagrangian dual. It starts with given initial multipliers and solves the corresponding Lagrangian relaxation. If the Lagrangian solution, i.e. x satisfying only (2) and (4), is also feasible to (3), it is optimal; otherwise violated multiple-choice constraints give descent directions of $L(u)$ over the multiplier space. The method then finds a step length along any descent direction, updates multipliers and iterates. Further discussion is given in Section 2.

In this paper, we propose algorithmic improvements to increase the effectiveness of the traditional MAM as a bounding tool. These improvements are due to a post-optimality analysis for the 0-1 knapsack subproblems based on a dynamic programming (DP) formulation. In computing the value of $L(u)$ for a given u, m knapsack problems are first solved using forward recursions. Subsequent backward recursions generate useful post-optimality information that enables the MAM to achieve a guaranteed steepest descent improvement at each iteration. It also eliminates all intermediate knapsack solutions required by the traditional MAM for calculating a step length. The post-optimality analysis is explained in Section 3, followed by a description of its use in improving the traditional MAM in Section 4. In Section 5, we present a branch-and-bound algorithm that incorporates the improved MAM in upper bounding. Moreover, we describe a lower bounding heuristic that makes feasible an infeasible Lagrangian solution. Computational results obtained with the algorithm are also reported.

2 Traditional MAM

In this section we describe the traditional MAM and discuss the improvements proposed. First, we introduce some further notation. If $(d_1, \ldots, d_m)$ is a vector of real numbers then $\min 2_i\{d_i\}$ is the second smallest element of d over i. Similarly, $\operatorname{argmin} 2_i\{d_i\}$ designates the index of the second smallest element. Define $\max 2_i\{d_i\}$ and $\operatorname{argmax} 2_i\{d_i\}$ analogously. Given some solution, x, let $p_j = \sum_i x_{ij}$ for all j. The traditional MAM, as we define it, is stated in detail in [12].

Given a violated multiple choice constraint, j^*, and a current optimal solution, x, the key calculation in the traditional MAM finds Δ_{ij^*}, the least decrease (increase) in u_{j^*} required for setting $x_{ij^*} = 1$ ($x_{ij^*} = 0$) in an optimum knapsack solution of row i if $p_{j^*} = 0$ ($p_{j^*} > 1$). The traditional MAM uses the following approach in determining Δ_{ij^*} quantities. Define $\bar{c}_{ij} = c_{ij} - u_j$, and $z_i = \sum_j \bar{c}_{ij} x_{ij}$. Suppose we removed variable x_{ij^*} from knapsack i. We will call the remaining problem with $(n-1)$ variables a *restricted knapsack*. Let $z_i^R(\beta)$ be the optimum objective value of the restricted knapsack when its right hand side capacity is β. Now, if $p_{j^*} = 0$ and $x_{ij^*} = 0$, the least increment in $\bar{c}_{ij^*}$ that would cause $x_{ij^*} = 1$ in an alternative optimum solution is given by

$$\Delta_{ij^*} = z_i - \bar{c}_{ij^*} - z_i^R(b_i - a_{ij^*}). \tag{6}$$

Similarly, if $p_{j^*} > 1$ and $x_{ij^*} = 1$, the least decrement in $\bar{c}_{ij^*}$ that would cause $x_{ij^*} = 0$ in an optimum solution of knapsack i is given by

$$\Delta_{ij^*} = z_i - z_i^R(b_i). \tag{7}$$

In the Fisher-Jaikumar-Van Wassenhove strategy it is never necessary to increase a multiplier; that is, for all j, $p_j \leq 1$ is maintained throughout. Decreasing multiplier u_{j^*} for

$p_{j^*} = 0$ improves the bound by $\Delta_{i^*j^*}$ at each iteration, where i^* is $\text{argmin}_i\{\Delta_{ij^*}\}$. However, Guignard-Rosenwein showed that, although such a strategy forces primal feasibility, it restricts the search over the multiplier space in solving the Lagrangian dual. Therefore, we choose not to impose any restriction on p_j and allow multipliers to be both decreased and increased. Increasing multiplier u_{j^*} for $p_{j^*} > 1$ in the Guignard-Rosenwein version improves the bound by $(p_{j^*} - 1)\Delta_{i^*j^*}$ (see also [8]).

The MAM that serves as a basis for our improvements is a hybrid of the techniques of Fisher-Jaikumar-Van Wassenhove and Guignard-Rosenwein. Hence, it has a different bound improvement rate than either. The following result, which establishes the new bound improvement scheme, is used for selecting the steepest descent bound improvement in Section 4. Let $\Delta = \text{min2}_i\{\Delta_{ij^*}\}$ be the step length chosen by this hybrid, traditional algorithm, where the min operator ranges over

$$\begin{cases} \text{all } i, & \text{if } p_{j^*} = 0 \\ \{i \mid x_{ij^*} = 1\}, & \text{if } p_{j^*} > 1. \end{cases}$$

Proposition 1 *Adjusting multipliers from u^k to u^{k+1} reduces the upper bound from $L(u^k)$ to $L(u^{k+1})$, where*

$$L(u^{k+1}) = \begin{cases} L(u^k) - \Delta_{i^*j^*}, & \textit{if } p_{j^*} = 0 \\ L(u^k) - (p_{j^*} - 2)\Delta - \Delta_{i^*j^*}, & \textit{if } p_{j^*} > 1. \end{cases}$$

Proof: See [12] (refereed with paper). ■

Proposition 1 suggests that the bound improvement is not strictly monotonic for the traditional MAM and hence, the method may stall at a certain bound value for a number of iterations without making any progress.

We observe the following issues that impair the performance of the traditional MAM:

1. In computing a step length along a candidate descent direction, m restricted knapsack problems must be solved if $p_{j^*} = 0$, and p_{j^*} restricted knapsack problems must be solved if $p_{j^*} > 1$.

2. A candidate descent direction is selected by scanning the set $\{j \mid p_j \neq 1\}$ in any given order. The descent direction selected is the first encountered that gives a nonzero step length. Once it is found, all remaining candidate directions are discarded

from further consideration. Although this strategy reduces the number of restricted knapsack solutions, it can result in slow convergence.

This paper proposes improvements for both issues with the help of a 0-1 knapsack post-optimality analysis described in the next section. The post-optimality analysis directly generates Δ_{ij} quantities for all i, j and hence, eliminates restricted knapsack solutions. Also, with Δ_{ij} quantities readily available as a table, it is possible to select a direction/step length pair to guarantee a steepest descent improvement of $L(u)$.

3 A Post-optimality Analysis for 0-1 Knapsack

In this section, we concentrate on one of the m 0-1 knapsack problems and suppress the i-index in the variables for clarity. Consider the following problem defining s_j as the complement of x_j ($s_j = 1 - x_j$):

$$\begin{aligned} z = \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_j x_j \le b \\ & x_j + s_j = 1 && \text{for } j = 1, \ldots, n \\ & x_j \in \{0, 1\} && \text{for all } j. \end{aligned}$$

We assume that b and a_j are integers for all j. The post-optimality analysis is based on a DP solution of 0-1 knapsack problems ([6], [18]). Let the state space consist of the set of ordered pairs (j, β), where j enumerates variables ($1 \le j \le n$) and β enumerates the knapsack capacity ($0 \le \beta \le b$). The knapsack function, $f_j(\beta)$, is defined to be the optimum value of a subproblem which uses variables 1 through j with a knapsack capacity of β. Consider a DP network in which nodes are states and arcs represent variable selection decisions. Arc lengths are assigned the objective values of corresponding variables. A standard recursion is used to find $f_j(\beta)$ = the longest path from state $(0,0)$ to (j, β) for each feasible state (j, β) in the DP network. The longest path length from $(0,0)$ to (n, b) is denoted z.

Similarly, we can make a backward pass to compute $g_j(\beta)$ defined as z minus the length of the longest path from state (j, β) to (n, b). Specifically, *Boundary conditions*:

$$\begin{aligned} g_{n-1}(\beta) &= z & &\text{for } b - a_n < \beta \le b \\ g_{n-1}(\beta) &= \min\{z,\ z - c_n\} & &\text{for } 0 \le \beta \le b - a_n \end{aligned}$$

Recursive equations: For $j = n-2, \ldots, 1$

$$\begin{aligned} g_j(\beta) &= g_{j+1}(\beta) & &\text{for } b - a_{j+1} < \beta \le b \\ g_j(\beta) &= \min\{g_{j+1}(\beta),\ g_{j+1}(\beta + a_{j+1}) - c_{j+1}\} & &\text{for } 0 \le \beta \le b - a_{j+1} \end{aligned}$$

Theorem 1 *Suppose $x_j = 0$ in the current optimum solution. Assuming that all other objective coefficients remain unchanged, the minimum δ such that $c_j + \delta$ would cause $x_j = 1$ in an alternative optimum is given by Δ_j, where*

$$\Delta_1 = g_1(a_1) - c_1 \tag{8}$$

$$\Delta_j = \min_{a_j \le \beta \le b}\{g_j(\beta) - f_{j-1}(\beta - a_j) - c_j\} \qquad \textit{for } j = 2, \ldots, n-1 \tag{9}$$

$$\Delta_n = z - f_{n-1}(b - a_n) - c_n. \tag{10}$$

Proof: If $x_j = 0$ then $s_j = 1$ in the current solution. Consider the DP network in which the longest path from $(0,0)$ to (n,b) has an arc corresponding to $s_j = 1$ (see Figure 1). Let one of the arcs representing x_j be from $(j-1, \beta - a_j)$ to (j, β). Define z_1 and z_2 as the longest path lengths from $(0,0)$ to $(j-1, \beta - a_j)$ and from (j, β) to (n, b), respectively. Let z^R be the length of the longest path from $(0,0)$ to (n,b) constrained to include the arc from $(j-1, \beta - a_j)$ to (j, β). Then

$$z^R = z_1 + c_j + z_2 \le z.$$

In order for x_j to be selected in an alternative optimum solution, z^R should be at least as large as z. To enforce this selection, we have to increase c_j by at least $z - z^R$, where

$$\begin{aligned} z - z^R &= z - (z_1 + c_j + z_2) \\ &= (z - z_2) - z_1 - c_j \\ &= g_j(\beta) - f_{j-1}(\beta - a_j) - c_j. \end{aligned}$$

Since there may be more than one arc corresponding to the variable x_j in the DP network, it is necessary to look for the minimum increment of c_j over all these arcs.

$$\begin{aligned} \min_{a_j \le \beta \le b}\{z - z^R\} &= \min_{a_j \le \beta \le b}\{g_j(\beta) - f_{j-1}(\beta - a_j) - c_j\} \\ &= \Delta_j \end{aligned}$$

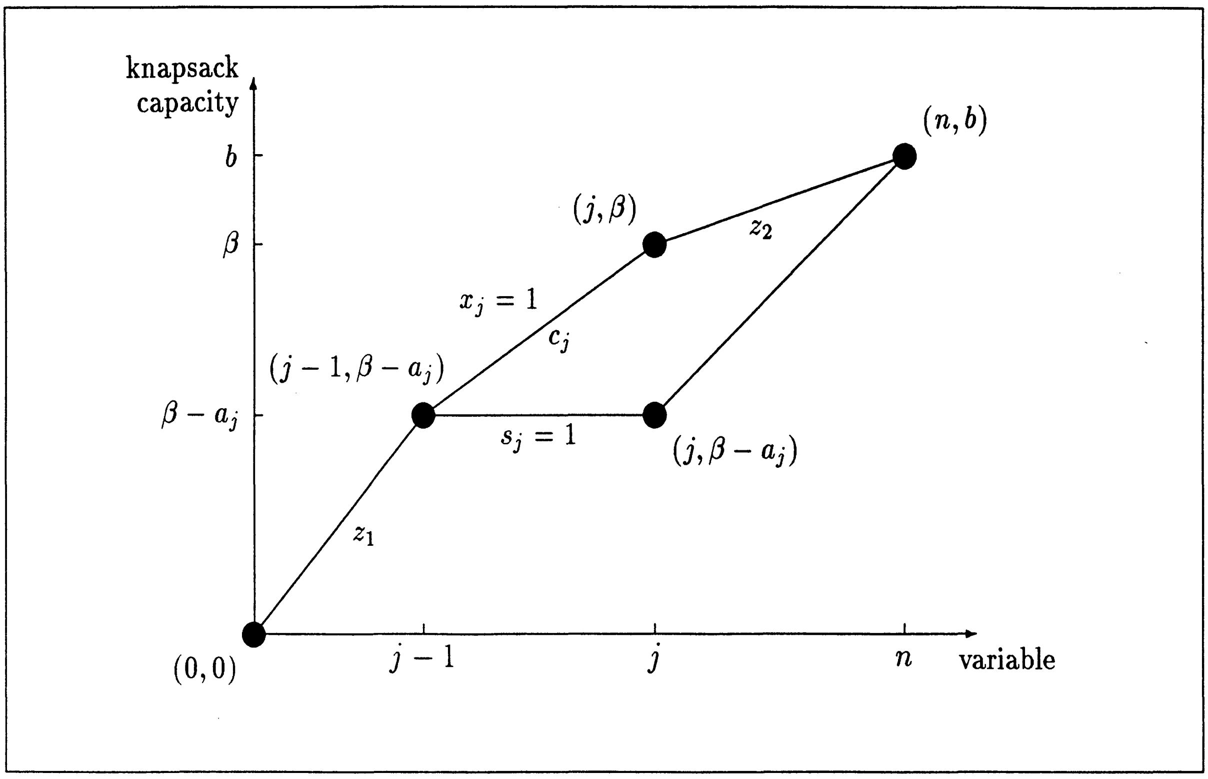

Figure 1: Illustration of the post-optimality analysis.

as given by equation (9). Consider the boundary conditions. For $j = 1, g_1(a_1) = \min g_1(\beta)$ over $a_1 \leq \beta \leq b$ and $z_1 = 0$; substitution gives equation (8). For $j = n$, $f_{n-1}(b - a_n) = \max f_{n-1}(\beta)$ over $0 \leq \beta \leq b - a_n$ and $z_2 = 0$; substitution gives equation (10). ∎

Corollary 1 *Suppose $x_j = 1$ in the current optimum solution. Assuming that all other objective coefficients remain unchanged, the minimum δ such that $c_j - \delta$ would cause $x_j = 0$ in an optimum solution is given by Δ_j, where*

$$\begin{aligned}
\Delta_1 &= g_1(0) \\
\Delta_j &= \min_{0 \leq \beta \leq b} \{g_j(\beta) - f_{j-1}(\beta)\} \qquad \text{for } j = 2, \ldots, n-1 \\
\Delta_n &= z - f_{n-1}(b).
\end{aligned}$$

Proof: If $x_j = 1$ then $s_j = 0$ in the current solution. The result directly follows by interchanging x_j and s_j in the theorem. ∎

It is interesting to note that the procedure implied by the above theorem and its corollary is similar to finding *total floats* in activity networks by the critical path method. The floats define Δ_j values. This idea is adopted from an algorithm for the capital constrained equipment replacement problem (see [10] and [11]). It is similar to analyses in Lee [13] and Van Hoesel and Wagelmans [19].

4 Improved MAM

The results of 0-1 knapsack post-optimality analyses combined with Proposition 1 enable us to improve the traditional MAM. The improved MAM is given by the following algorithm:

Step 0: Initialize multipliers by either $u^0 = \max2_i\{c_{ij}\}$ ([5]) or $u^0 = \max_i\{c_{ij}\}$ ([7]). Solve m knapsack problems to obtain Lagrangian solution, x^0. Perform post-optimality analysis for each knapsack to compute Δ^0_{ij} quantities. Set the iteration counter $k \leftarrow 0$.

Step 1: Define $p_j = \sum_i x^k_{ij}$, $i1(j) = \text{argmin}_i\{\Delta^k_{ij}\}$, $i2(j) = \text{argmin}2_i\{\Delta^k_{ij}\}$ for all j. Let $S = \{j \mid p_j \neq 1 \text{ and } \Delta^k_{i2(j),j} > 0\}$ and go to step 2.

Step 2: If $S = \phi$, stop; otherwise select index $j^* \in S$ such that

$$j^* = \underset{j\in S}{\text{argmax}} \left\{ \max_{\{j|p_j=0\}} \{\Delta^k_{i1(j),j}\}, \max_{\{j|p_j>1\}} \{(p_j-2)\Delta^k_{i2(j),j} + \Delta^k_{i1(j),j}\} \right\}.$$

Construct a descent direction $\xi^k = (\xi^k_1, \ldots, \xi^k_n)$, where

$$\xi^k_j = \begin{cases} +1, & \text{if } j = j^* \text{ and } p_{j^*} = 0 \\ -1, & \text{if } j = j^* \text{ and } p_{j^*} > 1 \\ 0, & \text{otherwise.} \end{cases}$$

Step 3: Letting the step length $\Delta = \Delta^k_{i2(j^*),j^*}$, adjust multipliers, $u^{k+1} \leftarrow u^k - \Delta\,\xi^k$. Solve knapsack problems followed by post-optimality analyses to compute respective x^{k+1}_{ij} and Δ^{k+1}_{ij} quantities. Set $k \leftarrow k+1$, and go to step 1.

In step 2, this method evaluates all possible descent directions and chooses the steepest for the next multiplier update. The extra computational effort introduced by post-optimality analyses is compensated by 1) eliminating all restricted knapsack solutions for determining step lengths, and 2) increasing the rate of bound reductions through a steepest descent improvement at each iteration.

5 Branch-and-Bound Enumeration

If the Lagrangian solution does not provide an optimal GAP solution, one or more multiple-choice constraints are violated. We branch from such solutions using a strategy suggested by Bean [1]. First, we select a violated multiple-choice constraint with the largest multiplier value. Since any feasible solution includes exactly one out of m variables from this constraint, we create m branches. At each branch, one variable is fixed at one, and the rest are automatically set to zero.

We also use the following heuristic in an attempt to attain feasibility given a Lagrangian solution x that violates (3):

Step 1: Store in a list all the variables x_{ij} such that $x_{ij} = 1$ and $p_j > 1$ in the current solution. Sort the list in ascending c_{ij}/a_{ij} order.

Step 2: Starting from the top of the list, set $x_{ij} = 0$ and increase the slack space of knapsack i by a_{ij}. Repeat until $p_j \leq 1$ for all j.

Step 3: Using only variables x_{ij} with $p_j = 0$, solve all knapsack problems optimally.

If the heuristic procedure stops with $p_j = 1$ for all j, the final solution is feasible and the incumbent is checked for improvement. Otherwise, the attempt to make the Lagrangian solution feasible fails. Computational experiments showed that the heuristic is more effective with Lagrangian solutions returned by a subgradient algorithm. Therefore, following the termination of the MAM, we perform a few subgradient iterations with multipliers initialized by the output of the MAM.

While subsequent subgradient iterations improve the MAM upper bound at the root node for certain types of problems, similar improvements were rarely observed for the descendent nodes where output multipliers of the MAM are very close to optimum. Therefore, the primary advantage of subgradient iterations is to tighten the lower bound by enhancing the above heuristic, and thus, to improve the overall branch-and-bound enumeration. A study of the quality of lower bounds from the heuristic and upper bounds from the MAM and subgradient iterations follows the experiment description.

Our branch-and-bound algorithm was programmed in C and compared against the Martello-Toth branch-and-bound algorithm coded in FORTRAN (program MTG in [15]). The programs were compiled using cc and xlf compilers, respectively, on the IBM RS/6000, with optimization switches turned on. Random problems were generated according to distributions suggested in [14] and [15]. The results are shown in Table 1.

The statistics were obtained over a sample of five problems. The letters A, B, C, and D refer to the four different problem generations:

A. Parameters a_{ij} and c_{ij} are integers uniformly random in [5, 25] and [1, 40], respectively. Let $\hat{x}$ be a solution of (1)–(4) with (2) replaced with $a_{ij}x_{ij} \leq b_i$, for all i, j. Then, $b_i = 9(n/m) + \max_i\{\sum_j a_{ij}\hat{x}_{ij}\}$.

B. Same as A, but b_i is set 70% of the value given in A.

C. Same as A, but $b_i = 0.8\sum_j a_{ij}/m$.

D1. Parameters a_{ij} uniformly random in [1, 100], $b_i = 0.8\sum_j a_{ij}/m$, and $c_{ij} = a_{ij} + K$, where K is uniformly random in [−10, 10].

D2. Same as D1, but c_{ij} is uniformly random in $[a_{ij}, a_{ij} + 20]$.

Class A problems, first suggested by Ross-Soland [16], usually admit many feasible solutions and are easier than the others. Problems of B, C, and D have increasingly tighter capacity constraints and become increasingly difficult to solve. In addition, class D problems have a correlation between objective values and knapsack weights (often observed in real-world applications) and are the most difficult.

The results in Table 1 show that our algorithm is comparable with Martello-Toth algorithm with easy problems of class A. However, our algorithm performs better with more difficult problems of B, C, and D. Note that performance will degrade if larger knapsacks are solved. See [5] for discussion of this issue.

Turning to the quality of the upper and lower bounds, we ran the same set of problems disabling branching and obtained root node bounds. Table 2 shows average deviation from optimal value of a) the lower bounding heuristic, b) the MAM upper bound, and c) the MAM followed by 50 subgradient iterations initialized by the final MAM multipliers. For all problem types and sizes, one can observe that final upper bounds are closer to the true optimal value than lower bounds. Moreover, the quality of MAM bound tends to deteriorate with increasing problem difficulty, resulting in greater improvements by additional subgradient iterations to reach the final node bound.

6 Summary

We describe a branch-and-bound algorithm featuring a more effective MAM than those of Fisher, Jaikumar, and Van Wassenhove [5] and Guignard and Rosenwein [7]. The MAM's effectiveness stems from a post-optimality analysis we developed for the 0-1 knapsack subproblems. This MAM is embedded in a branch-and-bound algorithm adapting the multiple choice branching strategy of Bean [1]. Computational tests indicate that the algorithm is capable of solving nontrivial problems with up to 500 variables in reasonable times. The code is available from the first author.

Acknowledgment

This research was supported in part by National Science Foundation grants PYI-G-DMC-8352346, DDM-9018515 and DDM-9202849 to the University of Michigan.

References

[1] Bean, J. C., "A Lagrangian Algorithm for the Multiple Choice Integer Program," *Operations Research*, 32 (1984), 1185-1193.

[2] Bean, J. C., C. E. Noon, S. M. Ryan, and G. J. Salton, "Selecting Tenants in a Shopping Mall," *Interfaces*, 18 (1988), 1-9.

[3] Chalmet, L. and Gelders L., "Lagrangean Relaxations for a Generalized Assignment-Type Problem," In M. Roubens (editor) *Advances in Operations Research*, North-Holland, Amsterdam (1977), 103-109.

[4] Fisher M. L. and R. Jaikumar, "A Generalized Assignment Heuristic for Vehicle Routing," *Networks*, 11 (1981), 109-124.

[5] Fisher M. L., R. Jaikumar, and L. N. Van Wassenhove, "A Multiplier Adjustment Method for the Generalized Assignment Problem," *Management Science*, 32 (1986), 1095-1103.

[6] Gilmore, P. C. and R. E. Gomory, "The Theory and Computation of Knapsack Functions," *Operations Research*, 14 (1966), 1045-1074.

[7] Guignard, M. and M. B. Rosenwein, "An Improved Dual Based Algorithm for the Generalized Assignment Problem," *Operations Research*, 37 (1989), 658-663.

[8] Guignard, M. and M. B. Rosenwein, "An Application-Oriented Guide for Designing Lagrangean Dual Ascent Algorithms," *European Journal of Operations Research*, 43 (1989), 197–205.

[9] Jörnsten, K. and M. Näsberg, "A New Lagrangian Relaxation Approach to the Generalized Assignment Problem," *European Journal of Operations Research*, 27 (1986), 313–323.

[10] Karabakal, N., "Capital Rationing Replacement Problem," Ph.D. Dissertation, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 1991.

[11] Karabakal, N., J. R. Lohmann, and J. C. Bean, "Parallel Replacement Under Capital Rationing Constraints." Forthcoming in *Management Science.*

[12] Karabakal, N., J. C. Bean and J. R. Lohmann, "A Steepest Descent Multiplier Adjustment Method for the Generalized Assignment Problem: Technical Report," Technical Report 92-11, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 1992.

[13] Lee, I.-S., "On Sensitivity Analysis for Shortest Path Dynamic Programming," Working Paper, University of California, Los Angeles, (1986).

[14] Martello, S. and P. Toth, "An Algorithm for the Generalized Assignment Problem," In J. P. Brans (editor) *Operational Research '81*, North-Holland, Amsterdam (1981), 589–603.

[15] Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, 1990.

[16] Ross, G. T. and R. M. Soland, "A Branch and Bound Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, 8 (1975), 91–103.

[17] Ross, G. T. and R. M. Soland, "Modeling Facility Location Problems as Generalized Assignment Problems," *Management Science*, 24 (1977), 345–357.

[18] Toth, P., "Dynamic Programming Algorithms for the Zero-One Knapsack Problem," *Computing*, 25 (1980), 29–45.

[19] Van Hoesel, C.P.M. and A.P.M. Wagelmans, "Sensitivity Analysis of the Economic Lot Sizing Problem," Report 9165/A of The Erasmus University Rotterdam, The Netherlands, (1991).

Table 1: Computational experience with random problems. Times are in CPU seconds on IBM RS/6000. Five problems for each line.

			Martello-Toth			Karabakal-Bean-Lohmann		
m	n	Type	Min	Median	Max	Min	Median	Max
5	20	A	0.03	0.04	0.06	0.10	0.14	0.17
		B	0.26	1.36	6.17	0.25	0.61	0.86
		C	0.19	1.35	5.48	0.25	0.68	1.11
		D1	4.62	7.84	12.74	6.26	12.79	16.42
		D2	88.44	452.21	515.81	0.87	4.33	5.10
5	30	A	0.04	0.06	0.09	0.15	0.27	1.31
		B	3.73	64.40	293.73	0.56	1.88	3.85
		C	8.51	141.51	367.59	0.53	4.14	8.11
		D1	45.78	569.40	780.23	19.17	69.01	95.41
		D2	TIME	TIME	TIME	30.18	65.35	95.25
5	40	A	0.04	0.07	0.53	0.26	0.39	3.41
		B	181.36	814.87	1000*	1.21	7.46	23.64
		C	226.25	1000*	1000*	4.49	6.55	57.58
		D1	973.40	1000*	1000*	287.96	1000*	1000*
		D2	—	—	—	203.43	462.26	642.53
5	50	A	0.05	0.08	30.61	0.27	0.81	5.39
		B	45.92	1000*	1000*	0.59	20.01	158.85
		C	TIME	TIME	TIME	16.24	30.77	42.92
		D1	TIME	TIME	TIME	474.50	999.14	1000*
		D2	—	—	—	543.59	1000*	1000*
10	20	A	0.03	0.07	0.10	0.16	0.49	1.82
		B	0.06	22.24	50.80	0.32	0.50	2.45
		C	0.38	7.63	29.27	0.12	0.62	1.32
		D1	12.29	65.40	153.91	3.80	7.85	12.10
		D2	TIME	TIME	TIME	3.19	6.11	14.19
10	30	A	0.07	0.10	11.57	0.26	0.30	6.12
		B	0.15	202.67	1000*	1.91	2.66	6.45
		C	TIME	TIME	TIME	3.43	12.25	20.55
		D1	TIME	TIME	TIME	94.57	156.75	437.52
		D2	—	—	—	76.91	125.61	510.90
10	40	A	0.07	0.10	0.11	0.48	1.92	4.40
		B	0.14	1000*	1000*	4.75	7.39	153.94
		C	—	—	—	9.16	29.31	41.92
		D1	—	—	—	987.25	1000*	1000*
		D2	—	—	—	TIME	TIME	TIME
10	50	A	0.09	0.95	4.46	0.55	1.74	12.63
		B	TIME	TIME	TIME	7.07	25.70	75.32
		C	—	—	—	23.69	70.43	355.64

TIME indicates that none of the problems in the sample could be solved within 1000 seconds.

* Problem exceeded 1000 second time limit.

Table 2: Average bound deviations from the optimum at the root node. Five problems for each line.

m	n	Problem Type	Average Deviation of Lower Bound (Heuristic) (%)	Average Deviation of Upper Bound (MAM) (%)	Average Deviation of Upper Bound (MAM + Subgradient) (%)
5	20	A	0.00	0.00	—
		B	0.93	1.68	0.69
		C	1.79	1.16	1.02
		D1	2.53	1.40	0.60
		D2	1.82	3.06	1.46
5	30	A	0.08	0.00	—
		B	1.83	0.87	0.39
		C	2.26	2.17	0.98
		D1	2.30	3.15	0.51
		D2	1.69	5.26	0.40
5	40	A	0.06	0.05	0.00
		B	2.95	0.93	0.47
		C	3.13	1.71	0.72
		D1	1.53	6.03	0.33
		D2	2.31	12.13	0.54
5	50	A	0.29	0.02	0.00
		B	2.79	1.41	0.37
		C	1.80	1.61	0.61
		D1	1.80	8.89	0.28
		D2	1.17	16.76	0.27
10	20	A	0.84	0.00	—
		B	1.61	1.06	0.51
		C	0.09	1.20	0.28
		D1	1.92	1.88	0.87
		D2	3.29	2.31	0.99
10	30	A	0.56	0.00	—
		B	1.47	0.38	0.12
		C	3.74	1.30	1.16
		D1	2.79	3.95	0.49
		D2	2.47	11.24	0.67
10	40	A	0.49	0.00	—
		B	1.30	0.57	0.29
		C	1.96	1.37	0.34
		D1	4.14	19.19	1.43
		D2	3.57	20.07	0.48
10	50	A	0.37	0.00	—
		B	0.70	0.58	0.10
		C	1.77	1.05	0.33