A Branch and Price algorithm for the multi-period single-sourcing problem

Richard Freling* H. Edwin Romeijn† Dolores Romero Morales‡
Albert P.M. Wagelmans§

October 1999

Econometric Institute Report EI 9941/A

Abstract

In this paper we propose a Branch and Price algorithm for solving multi-period single-sourcing problems. In particular, we generalize a Branch and Price algorithm that was developed for the Generalized Assignment Problem (GAP) to a class of convex assignment problems. We then identify an important subclass of problems, containing many variants of the multi-period single-sourcing problem (MPSSP), as well as variants of the GAP, for which we derive an efficient solution procedure for the pricing problem, a critical factor in the efficiency of the Branch and Price algorithm. We execute an extensive numerical comparison between the performances of the Branch and Price algorithm and the MIP solver of CPLEX for a particular variant of the MPSSP.

1 Introduction

Some of the most important problems in logistics faced by a supplier are the timing of production, the location of inventories, and the assignment of customers to warehouses. In this paper we will study a multi-period single-sourcing problem (MPSSP) that can be used to support the corresponding decisions. The model we propose is dynamic in nature, in contrast to many of the quantitative models proposed in the literature which assume a static environment. The fact that our model is dynamic enables us to handle a dynamic demand pattern of the customers, as well as to support inventory decisions

*Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands; e-mail: freling@few.eur.nl; and ORTEC Consultants B.V., P.O. Box 490, 2800 AL Gouda, The Netherlands; email: rfreling@ortec.nl.

†Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, P.O. Box 116595, Gainesville, Florida 32611-6595; email: romeijn@ise.ufl.edu.

‡Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands; e-mail: D.Romero@fbk.eur.nl.

§Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands; e-mail: wagelmans@few.eur.nl.

explicitly. Related literature, focusing on static models, can be found in Geoffrion and Graves [10], Benders et al. [2], and Fleischmann [9]. Duran [7] studies a dynamic model for the planning of production, bottling, and distribution of beer, but focuses on the production, instead of the distribution, process. Chan, Muriel and Simchi-Levi [4] study a dynamic, but uncapacitated, distribution problem.

The logistics network we are considering consists of a set of facilities (each of which could be interpreted as a plant with an associated warehouse), and a set of customers. The decisions that need to be made concern (i) the assignment of customers to facilities, and (ii) the location and size of inventories. These two types of decisions can be handled in a nested fashion, where we essentially decide on the assignment of customers to facilities only, and where the location and size of inventories are determined optimally as a function of the customer assignments. Viewed in this way, the multi-period single-sourcing problem is a generalized assignment problem with a convex objective function and possibly additional constraints, representing, for example, throughput or physical inventory capacities, or perishability constraints. To be able to deal with many variants of the multi-period single-sourcing problem using a single solution approach, we will introduce a general class of *convex assignment problems*, having the property that the objective function and feasible region are convex, and are both separable in the facilities. The class of convex assignment problems clearly contains the well-known Generalized Assignment Problem (GAP), and thus convex assignment problems are $\mathcal{NP}$*-Hard* as well. We will discuss one of the variants of the multi-period single-sourcing problem in detail in this paper. In this variant each plant has known, finite, and possibly time-varying, capacity, each customer needs to be served by (assigned to) a unique facility throughout the planning horizon, and the customer demands exhibit a seasonal pattern.

The outline of the paper is as follows. In Section 2 we will introduce a class of convex assignment problems, CAP, and propose an exact branch-and-price procedure for solving these problems based on a column generation approach for a set partitioning formulation of the problem. This approach generalizes a similar branch-and-price procedure for the GAP (see Savelsbergh [22]). In Section 3 we study the pricing problem for a particular subclass for which an efficient branch-and-bound procedure can be constructed. In Section 4 we formulate the variant of the multi-period single-sourcing problem mentioned above. In Section 5 we illustrate the performance of the Branch and Price scheme for this variant of the MPSSP. In Section 6 we end the paper with some concluding remarks.

2 Solving convex assignment problems

2.1 Convex assignment problems

Consider the following convex assignment problem:

$$\text{minimize} \sum_{i=1}^{m} g_i(x_{i\cdot})$$

subject to (CAP)

$$
\begin{array}{rcll}
\sum_{i=1}^{m} x_{ij} & = & 1 & j = 1, \ldots, n \\
x_{ij} & \in & \{0,1\} & i = 1, \ldots, m;\ j = 1, \ldots, n \\
x_{i\cdot} & \in & X_i & i = 1, \ldots, m
\end{array}
$$

where the functions g_i are convex, as are the sets X_i denoting any additional constraints. Ferland, Hertz and Lavoie [8] introduce an even more general class of assignment problems, and show the applicability of object-oriented programming by developing software containing several heuristics. As mentioned in the introduction, the GAP is an example of a convex assignment problem, where the cost function g_i and the additional constraints defined by the set X_i associated with agent i are linear in $x_{i\cdot}$. Variants of the MPSSP are examples of convex assignment problems as well, one of which will be discussed in detail in Section 4. In a more general context, all set partitioning models discussed by Barnhart et al. [1] with convex and separable objective function in the index i are examples of convex assignment problems. The CAP can be formulated as a set partitioning problem, in a similar way as was done for the GAP by Cattryse, Salomon, and Van Wassenhove [3]; and Savelsbergh [22]. In particular, a feasible solution for (CAP) can be seen as a partition of the set of *objects* $\{1, \ldots, n\}$ into m subsets. Each element of the partition is associated with one of the m *agents*.

Now let L_i be the number of subsets of objects that can feasibly be assigned to agent i ($i = 1, \ldots, m$). Let $\alpha_{i\cdot}^{\ell}$ denote the ℓ-th subset (for fixed i), i.e., $\alpha_{ij}^{\ell} = 1$ if object j is an element of subset ℓ for agent i, and $\alpha_{ij}^{\ell} = 0$ otherwise. We will call $\alpha_{i\cdot}^{\ell}$ the ℓ-th *column* for agent i. Then, the set partitioning problem can be formulated as follows:

$$\text{minimize } \sum_{i=1}^{m} \sum_{\ell=1}^{L_i} g_i(\alpha_{i\cdot}^{\ell})\, y_i^{\ell}$$

subject to (MP)

$$
\begin{array}{rcll}
\sum_{i=1}^{m} \sum_{\ell=1}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} & = & 1 & j = 1, \ldots, n \qquad (1) \\
\sum_{\ell=1}^{L_i} y_i^{\ell} & = & 1 & i = 1, \ldots, m \qquad (2) \\
y_i^{\ell} & \in & \{0,1\} & \ell = 1, \ldots, L_i;\ i = 1, \ldots, m
\end{array}
$$

where y_i^{ℓ} is equal to 1 if column ℓ is chosen for agent i, and 0 otherwise. As mentioned by Barnhart et al. [1], the convexity constraint (2) for agent i ($i = 1, \ldots, m$) can be written as

$$\sum_{\ell=1}^{L_i} y_i^{\ell} \le 1$$

if $\alpha_{ij} = 0$ for each $j = 1, \ldots, n$ is a feasible column for agent i with associated costs $g_i(\alpha_{i\cdot}) = 0$. One of the advantages of (MP) is that its linear relaxation LP(MP) gives a bound on the optimal solution value of (MP) that is at least as tight (and usually tighter) as the one obtained by relaxing the integrality constraints in (CAP), R(CAP). Hence, if we let v(R(CAP)) and v(LP(MP)) denote the optimal objective values of R(CAP) and LP(MP), respectively, then the following holds.

Proposition 2.1 *The following inequality holds:*

$$v(R(CAP)) \leq v(LP(MP)).$$

Proof: First of all, note that if LP(MP) is infeasible, the inequality follows directly since in that case $v(\text{LP(MP)}) = \infty$. In the more interesting case that LP(MP) is feasible, the desired inequality follows from the convexity of the objective function and the feasible region of (CAP). We may observe that both relaxations can be obtained by relaxing the integrality constraints to nonnegativity constraints. Each feasible solution to LP(MP) can be transformed to a feasible solution to R(CAP) as follows:

$$x_{ij} = \sum_{\ell=1}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \qquad i = 1, \ldots, m; \; j = 1, \ldots, n.$$

For each $i = 1, \ldots, m$, vector $x_{i\cdot}$ is a convex combination of vectors $\alpha_{i\cdot}^{\ell}$ for $\ell = 1, \ldots, L_i$. Since all constraints in (CAP) are convex x is a feasible solution for (CAP). Moreover, by convexity of the functions g_i we have that

$$\sum_{i=1}^{m} g_i(x_{i\cdot}) = \sum_{i=1}^{m} g_i \left(\sum_{\ell=1}^{L_i} \alpha_{i\cdot}^{\ell} y_i^{\ell} \right) \leq \sum_{i=1}^{m} \sum_{\ell=1}^{L_i} g_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell}.$$

Thus, the desired inequality follows. □

This result suggests that the formulation (MP) is more promising than (CAP) when solving the convex assignment problem by a Branch and Bound scheme.

2.2 A Branch and Price scheme for (MP)

2.2.1 Solving the convex assignment problem

The convex assignment problem is a (non-linear) Integer Programming Problem which can be solved to optimality by using, for example, a Branch and Bound algorithm. One of the factors determining the performance of this algorithm is the quality of the lower bounds used to fathom nodes. Proposition 2.1 shows that the lower bound given by relaxing the integrality constraints in (MP) is at least as good as the one obtained by relaxing the integrality constraints in (CAP). Thus, the set partitioning formulation for the convex assignment problem looks more attractive when choosing a Branch and Bound scheme. There are other reasons to opt for this formulation like the possibility of adding constraints that are difficult to express analytically.

A standard Branch and Bound scheme would require all the columns to be available, but (in the worst case) the number of columns (and thus the number of variables) of (MP) can be exponential in the size of the problem. This makes a standard Branch and Bound scheme quite unattractive for (MP). However, since the number of constraints in (MP) is relatively small with respect to the number of variables, only few variables will have strictly positive value in the optimal solution of LP(MP). Thus, only a very small subset of columns is relevant in the optimization of LP(MP). Basically, this is the philosophy behind Column Generation techniques (see Gilmore and Gomory [11]). Combining a Branch and Bound scheme with a column generation procedure yields a so-called Branch and Price algorithm. Barnhart et al. [1] have unified the literature on Branch and Price algorithms for large scale Mixed Integer Problems. They focus on branching rules and some computational issues relevant in the implementation of a Branch and Price scheme. We will concentrate mainly on the pricing problem. We will also present a greedy heuristic which generates an initial set of columns for (MP). A similar approach has been followed by Chen and Powell [5] for parallel machine scheduling problems when the objective function is additive in the jobs.

2.2.2 Column generation scheme

Usually, the number of columns associated with each agent will be extremely large, thus prohibiting the construction and solution of LP(MP) as formulated above. However, one may solve LP(MP) using only a subset (say N) of its columns (and refer to the corresponding reduced problem as LP(MP(N))). If it is then possible to check whether this solution is optimal for LP(MP), and to generate an additional column that will improve this solution if it is not, we can solve LP(MP) using a so-called *column generation* approach:

Column generation for LP(MP)

Step 0. Construct a set of columns, say $N_0 \subseteq \{(\ell, i) : \ell = 1, \ldots, L_i;\ i = 1, \ldots, m\}$, such that LP(MP($N_0$)) has a feasible solution. Set $N = N_0$.

Step 1. Solve LP(MP(N)), yielding $y^*(N)$.

Step 2. If $y^*(N)$, extended to a solution of LP(MP) by setting the remaining variables to zero, is optimal for LP(MP): STOP.

Step 3. Find a column (or a set of columns) so that the new objective value is at least as good as the objective value of $y^*(N)$ and add this column (or set of columns) to N. Go to Step 1.

Steps 2 and 3 verify that the optimal solution of LP(MP(N)) is also optimal for LP(MP) or find a new columns to add to LP(MP(N)) that may improve the current objective value. The information contained in the optimal dual multipliers of the constraints of LP(MP(N)) is used to perform those steps. In the following we will describe the steps of the algorithm in more detail.

Step 0: Initial columns

The column generation procedure calls for an initial set of columns N_0 to start with. For this purpose, a straightforward generalization of the class of greedy heuristics proposed by Martello and Toth [14] for the GAP can be used. An element of this class asymptotically yields a feasible and optimal solution with probability one for large numbers of customers (see Romeijn and Romero Morales [19]). The basic idea is that each possible assignment of an object to an agent is evaluated by a pseudo-cost function $f(i,j)$. The *desirability* of assigning an object is measured by the difference between the second smallest and the smallest values of $f(i,j)$. Assignments of objects are made in decreasing order of this difference. Along the way, some agents will not be able to handle some of the objects due to the constraints defined by the sets X_i, and consequently the values of the desirabilities will be updated taking into account that the two most desirable agents for each object should be feasible.

We will denote a partial solution for (CAP) by x^{G}. Let $\hat{\jmath}$ be an object which has not been assigned yet and $x^{\mathrm{G}} \cup \{(\hat{\imath},\hat{\jmath})\}$ the partial solution for (CAP) where the assignment of object $\hat{\jmath}$ to agent $\hat{\imath}$ is added to x^{G}. More formally,

$$(x^{\mathrm{G}} \cup \{(\hat{\imath},\hat{\jmath})\})_{ij} = \begin{cases} x^{\mathrm{G}}_{ij} & \text{if } j \neq \hat{\jmath};\ i = 1,\ldots,m \\ 1 & \text{if } (i,j) = (\hat{\imath},\hat{\jmath}) \\ 0 & \text{otherwise.} \end{cases}$$

This greedy heuristic can formally be written as follows:

Greedy heuristic for (CAP)

Step 0. Set $L = \{1,\ldots,n\}$, $\mathcal{NA} = \emptyset$ and $x^{\mathrm{G}}_{ij} = 0$ for each $i = 1,\ldots,m$ and $j = 1,\ldots,n$.

Step 1. Let

$$\mathcal{F}_j = \{i = 1,\ldots,m : x^{\mathrm{G}} \cup \{(i,j)\} \in X_i\} \qquad \text{for } j \in L.$$

If $\mathcal{F}_j = \emptyset$ for some $j \in L$, the algorithm cannot assign object j; then set $L = L \setminus \{j\}$, $\mathcal{NA} = \mathcal{NA} \cup \{j\}$ and repeat Step 1. Otherwise, let

$$\begin{aligned} i_j &\in \arg\min_{i \in \mathcal{F}_j} f(i,j) && \text{for } j \in L \\ \rho_j &= \min_{\substack{s \in \mathcal{F}_j \\ s \neq i_j}} f(s,j) - f(i_j,j) && \text{for } j \in L. \end{aligned}$$

Step 2. Let $\hat{\jmath} \in \arg\max_{j \in L} \rho_j$. Set

$$\begin{aligned} x^{\mathrm{G}}_{i_{\hat{\jmath}}\hat{\jmath}} &= 1 \\ L &= L \setminus \{\hat{\jmath}\}. \end{aligned}$$

Step 3. If $L = \emptyset$: STOP. If $\mathcal{NA} = \emptyset$, x^{G} is a feasible assignment for (CAP), otherwise x^{G} is a partial feasible assignment for (CAP). Otherwise, go to Step 1.

The challenge is to specify a pseudo-cost function that will yield a good (or at least a feasible) solution to (CAP). See Martello and Toth [14] for suggested pseudo-cost functions for the GAP; and Romeijn and Romero Morales [19, 21] for pseudo-cost functions for the GAP and a variant of the multi-period single-sourcing problem for which the greedy heuristic is asymptotically feasible and optimal in a probabilistic sense.

The output of this heuristic is a vector of feasible assignments x^{G}, which is (at least) a partial solution to (CAP), and thus yields a set of columns for (MP). As mentioned in the previous section, the optimal dual vector of $\mathrm{LP}(\mathrm{MP}(N_0))$ is required to perform Steps 2 and 3. Thus, when the solution is only a partial one, $\mathrm{LP}(\mathrm{MP}(N_0))$ is infeasible and we cannot start the column generation procedure. Moreover, it could also be the case that (MP) is infeasible. To overcome those two situations we have added a dummy variable $s_j \geq 0$ to the j-th constraint (1) with a high cost, for each $j = 1, \ldots, n$. This ensures that $\mathrm{LP}(\mathrm{MP}(N_0))$ always has a feasible solution, and infeasibility of this LP-problem is characterized by the positiveness of some of the dummy variables.

Steps 2 and 3: The Pricing Problem

A major issue in the success of the column generation approach is of course the viability of Steps 2 and 3. The usual approach is to consider the dual problem D(MP) to LP(MP):

$$\text{maximize } \sum_{j=1}^{n} u_j - \sum_{i=1}^{m} \delta_i$$

subject to $\qquad$ D(MP)

$$\begin{array}{rcll} \displaystyle\sum_{j=1}^{n} \alpha_{ij}^{\ell} u_j - \delta_i & \leq & g_i(\alpha_{i\cdot}^{\ell}) & \ell = 1, \ldots, L_i;\ i = 1, \ldots, m \\ u_j & & \text{free} & j = 1, \ldots, n \\ \delta_i & & \text{free} & i = 1, \ldots, m. \end{array}$$

Now note that the optimal dual solution corresponding to $y^*(N)$, say $(u^*(N), \delta^*(N))$, satisfies all dual constraints in D(MP) corresponding to elements $(\ell, i) \in N$. Moreover, if it satisfies *all* dual constraints in D(MP), then $y^*(N)$ (extended with zeroes) is the optimal solution to LP(MP). The challenge is thus to check feasibility of the dual solution $(u^*(N), \delta^*(N))$. This can, for example, be achieved by solving, for each $i = 1, \ldots, m$, the following optimization problem

$$\text{minimize } g_i(z) - \sum_{j=1}^{n} u_j^*(N) z_j + \delta_i^*(N)$$

subject to

$$\begin{array}{rcll} z_j & \in & \{0, 1\} & j = 1, \ldots, n \\ z & \in & X_i & \end{array}$$

thereby finding the minimum slack in all dual constraints. If all these optimization problems yield a nonnegative value, then all dual constraints are satisfied. Otherwise, feasible solutions with positive objective function value correspond to columns that would enter the basis if added to LP(MP(N)) (starting from $y^*(N)$).

The success of the column generation procedure depends on the ability to solve this subproblem efficiently, thus, its structure is crucial. For example, Savelsbergh [22] shows that this subproblem turns out to be a Knapsack Problem for the case of the GAP.

2.2.3 Branching

If the optimal solution of the LP-relaxation of (MP) is not integer we need to branch to obtain an optimal integer solution. Since the LP-relaxation of (MP) has been solved by column generation, it is unlikely that all columns are present in the final reduced linear programming problem. Thus, by using a Branch and Bound scheme using only the columns thus generated, we will in the best case end up with only a feasible solution for (MP). This approach thus yields a heuristic for solving the convex assignment problem.

If we want a certificate of optimality new columns (when needed) should be generated when branching. The choice of the branching rule is crucial since it can destroy the structure of the pricing problem. The straightforward choice would be to branch on the variables y_i^ℓ. Fixing one of those variables to zero is equivalent to prohibiting the generation of that column again. As Savelsbergh [22] pointed out for the GAP, with this branching rule we may need to find not only the optimal solution of the pricing problem but also the second optimal solution for it. Usually we cannot incorporate this additional information into the pricing problem directly, thereby prohibiting an efficient algorithm for the pricing problem. However, the proof of Proposition 2.1 shows that each feasible solution y for (MP) has a corresponding feasible solution x for (CAP). Moreover, it is easy to see that if y is fractional then x is fractional as well. Thus, we can branch on the fractional variables x_{ij}. We may observe that the subproblems obtained by branching on the x_{ij} variables are again convex assignment problems. Thus, the column generation procedure in each node of the tree is the same as in the root node.

3 A special case

3.1 Introduction

In this section we will consider a class of convex assignment problems for which the pricing problem exhibits an attractive property. In the disciplines of logistics and scheduling, the situation where objects require some resources which are available at the agents appears frequently (for example, see Mazzola and Neebe [16]). The limited availability of those resources can often be modeled by means of knapsack constraints and the costs as linear functions.

In the remainder of this section we will analyze the class of convex assignment problems where for each agent i the set X_i is defined by a knapsack constraint and the costs g_i are equal to the sum of a linear function in $x_{i\cdot}$ and a convex penalization of the use of the resource of agent i. More precisely, we will choose

$$X_i = \left\{ z \in [0,1]^n \,:\, \sum_{j=1}^{n} \omega_{ij} z_j \leq \Omega_i \right\}$$
$$g_i(z) = \sum_{j=1}^{n} \nu_{ij} z_j + G_i \left(\sum_{j=1}^{n} \omega_{ij} z_j \right) \qquad \text{for each } z \in \mathbb{R}^n.$$

We may notice that the GAP is still a member of this class with $G_i = 0$ for each $i = 1, \ldots, m$. Some extensions of the GAP are also included. The convex penalty function could be seen as a way of modeling a situation where the resource capacities are not rigid, and where they are allowed to be exceeded at some cost (see Srinivasan and Thompson [23]). Another example could be that a convex penalty is used to model the fact that it is undesirable to plan the use of resources to full capacity, due to possible deviations from the predicted requirements when a solution is implemented. As we will see in Section 4, a variant of the MPSSP is another relevant example of a member of this class.

Since this is a subclass of the class of convex assignment problems, we can use the Branch and Price scheme described in Section 2.2. As mentioned above, the success of this procedure depends on how efficiently we can solve the pricing problem. After rearranging terms and transforming it into a maximization problem, the pricing problem for this class associated with agent i reads

$$\text{maximize} \sum_{j=1}^{n} \left(u_j^*(N) - \nu_{ij} \right) z_j - G_i \left(\sum_{j=1}^{n} \omega_{ij} z_j \right) - \delta_i^*(N)$$

subject to

$$\sum_{j=1}^{n} \omega_{ij} z_j \leq \Omega_i$$
$$z_j \in \{0,1\} \qquad j = 1, \ldots, n.$$

Without loss of optimality we can leave out the constant term $\delta_i^*(N)$. The feasible region of this problem is described by a knapsack constraint. As in the Knapsack Problem, items which are added to the knapsack yield a profit $u_j^*(N) - \nu_{ij}$. However, in contrast with the traditional Knapsack Problem, the utilization of the knapsack is penalized by the convex function G_i. We will call this problem the Penalized Knapsack Problem (PKP) and it will be analyzed in the next section.

3.2 The Penalized Knapsack Problem

3.2.1 Definition of the problem

Consider a knapsack with a certain capacity and a set of items which make use of this capacity. When adding an item to the knapsack a profit is obtained. However, the total use of the knapsack will be penalized by a convex function. The PKP is the problem of choosing items in such a way that the capacity constraint is not violated when we add those items to the knapsack and the total profit minus the penalization on the use of the knapsack is maximal.

Let n denote the number of items. The required space of item j is given by $\omega_j \geq 0$, and the profit associated with adding item j to the knapsack is equal to $p_j \geq 0$. Let Ω be the capacity of the knapsack, and let $G(u)$ denote the penalization of using u units of capacity of the knapsack, where G is a convex function. The PKP can then be formulated as follows:

$$\text{maximize} \sum_{j=1}^{n} p_j z_j - G\left(\sum_{j=1}^{n} \omega_j z_j\right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} \omega_j z_j &\leq \Omega \\ z_j &\in \{0,1\} \qquad\qquad j = 1, \ldots, n. \end{aligned}$$

Since all items with zero space requirement will definitely be added to the knapsack, we can without loss of generality assume that $\omega_j > 0$ for each $j = 1, \ldots, n$. However, contrary to the ordinary knapsack problem, items with $p_j = 0$ cannot a priori be excluded from the knapsack, since the penalty function is not required to be nondecreasing, and thus adding such an item to the knapsack can be profitable. If the penalization on the use of the knapsack is nonpositive, i.e. $G(u) \leq 0$ for each $u \in [0, \Omega]$, we know that the optimal solution of the problem is maximal in the sense that no additional items can be added to the knapsack without violating the capacity constraint, see Martello and Toth [15]. However, in the general case, it may occur that the profit associated with adding an item to the knapsack is not enough to compensate for the penalization of the capacity used to add this item to the knapsack. The same follows for the relaxation of the PKP, say R(PKP), where the integer constraints are relaxed.

Consider the case where some items have been already added to the knapsack. Let u be the used capacity by those items. We will say that item j not yet in the knapsack is a *feasible* item for the knapsack if

$$u + \omega_j \leq \Omega,$$

and that it is *profitable* if it is feasible and

$$p_j - G(u + \omega_j) = \max_{\gamma \in [0,1]} \{p_j \gamma - G(u + \omega_j \gamma)\}.$$

3.2.2 Example

Consider the following example of the PKP where there is only one item to be added to the knapsack ($n = 1$) with profit $p_1 = 10$ and required space $\omega_1 = 20$. Moreover, let the capacity of the knapsack be equal to $\Omega = 25$ and the penalization equal to $G(u) = 15u^2$. This particular instance of the PKP reads

$$\text{maximize}\, 10z_1 - 15z_1^2$$

subject to

$$
\begin{aligned}
20z_1 &\leq 25 \\
z_1 &\in \{0,1\}.
\end{aligned}
$$

The item is feasible since the required space (20) is below the capacity (25). The objective value of not adding the item to the knapsack ($z_1 = 0$) is equal to 0, and the cost of adding it to the knapsack completely is equal to -5. Thus, the item is not profitable and the optimal solution of the PKP is equal to 0. Figure 1 plots the value of its relaxation R(PKP). We may observe that the maximum of this function is attained at $z_1^* = \frac{1}{3}$ even though, as we have seen before, the item can be feasibly added to the knapsack.

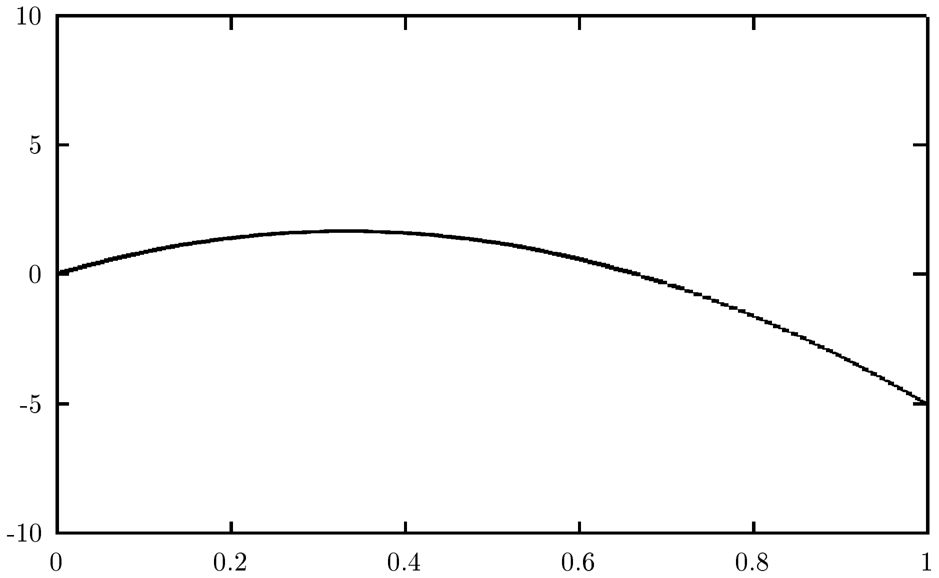

Figure 1: Value of the relaxation R(PKP) of PKP

3.2.3 The relaxation

One of the properties of the PKP is that the optimal solution of its relaxation R(PKP) has the same structure as the optimal solution of the LP-relaxation of the standard Knapsack Problem (see Martello and Toth [15]), and can be solved explicitly as well.

Assume that the items are ordered according to non-increasing ratio p_j/ω_j. Assume that items 1 till item $\ell - 1$ can be feasibly added to the knapsack. Let $P^\ell(\gamma)$, where $\gamma \in [0,1]$, be the objective value of R(PKP) associated with the solution $z_j = 1$ for each $j = 1, \ldots, \ell - 1$, $z_j = 0$ for each $j = \ell + 1, \ldots, n$, and $z_\ell = \gamma$. The next lemma shows the behaviour of this function.

Lemma 3.1 *$P^\ell(\cdot)$ is a concave function.*

Proof: Let u be the capacity used by items 1 till $\ell - 1$, i.e., $u = \sum_{j=1}^{\ell-1} \omega_j$. The domain of the function P^ℓ is the segment $\left[0, \min\{1, \frac{\Omega - u}{\omega_\ell}\}\right]$. Thus, we have that

$$P^\ell(\gamma) = \sum_{j=1}^{\ell-1} \omega_j + p_\ell \gamma - G(u + \omega_\ell \gamma)$$

which a concave function in γ. $\square$

Given that items 1 till $\ell - 1$ have been added to the knapsack, item ℓ is profitable if it is feasible and the maximum of the function $P^\ell(\cdot)$ is reached at $\gamma = 1$. This can be characterized by the condition $(P^\ell)'_-(1) \geq 0$, where $(P^\ell)'_-(\gamma)$ denotes the left derivative of the function P^ℓ in γ. Define items k_1 and k_2 as

$$\begin{aligned} k_1 &= \min\{\ell = 1, \ldots, n : \sum_{j=1}^{\ell} \omega_j > \Omega\} \\ k_2 &= \min\{\ell = 1, \ldots, n : (P^\ell)'_-(1) < 0\}. \end{aligned}$$

By definition, k_1 is the first item which cannot be added completely to the knapsack due to the capacity constraint, and item k_2 is the is the first item which will not be added completely to the knapsack due to the penalization of the capacity utilization. Now define item s as

$$s = \min(k_1, k_2),$$

i.e., the first item which should not be completely added to the knapsack due to either the capacity constraint or because it is not profitable. In the next proposition we show that the optimal solution for R(PKP) just adds to the knapsack items $1, \ldots, s-1$ and the feasible and profitable fraction γ^* of item s, i.e.,

$$\gamma^* = \min\{\gamma_1^*, \gamma_2^*\}$$

where

$$\gamma_1^* = \frac{\Omega - \sum_{j=1}^{s-1} \omega_j}{\omega_s}$$

and γ_2^* is the maximizer of the function $P^s(\cdot)$.

Proposition 3.2 *The vector $\bar{z} \in \mathbb{R}^n$ defined by*

$$\bar{z}_j = \begin{cases} 1 & \text{if } j < s \\ \gamma^* & \text{if } j = s \\ 0 & \text{if } j > s \end{cases}$$

is an optimal solution for R(PKP).

Proof: Let z be a feasible solution for R(PKP). The idea is to show that there exists a feasible solution $\hat{z}$ at least as good as z so that $\hat{z}_j = 1$ for each $j = 1, \ldots, s-1$ and $\hat{z}_j = 0$

for each $j = s+1, \dots, n$. By the definition of item s and fraction γ^*, $\bar{z}$ is at least as good as $\hat{z}$. Thus, the desired result follows.

Suppose that there exists an item $r = 1, \dots, s-1$ so that $z_r < 1$. If $z_q = 0$ for each $q = s, \dots, n$ we can construct a better solution by increasing z_r to 1 because the first $s-1$ items are feasible and profitable. Thus, assume that there exists $q = s, \dots, n$ so that $z_q > 0$. By increasing z_r by $\varepsilon > 0$ and decreasing z_q by $\varepsilon\omega_r/\omega_q$ the used capacity remains unchanged which implies that the penalization remains the same. Moreover, the profit associated with the new solution is at least as good as the profit in z since

$$p_r\varepsilon - p_q\varepsilon\,\omega_r/\omega_q = \varepsilon\,\omega_r(p_r/\omega_r - p_q/\omega_q) \geq 0$$

because $r < q$. Hence, we can assume that $z_j = 1$ for each $j = 1, \dots, s-1$.

Now we will prove that $z_j = 0$ for each $j = s+1, \dots, n$. Suppose that there exists an item $r = s+1, \dots, n$ so that $z_r > 0$. Then, $z_s < \gamma_1^*$ since $z_j = 1$ for each $j = 1, \dots, s-1$. In this case, by increasing z_s by $\varepsilon > 0$ and decreasing z_r by $\varepsilon\omega_r/\omega_q$ it follows in a similar way as above that the new solution is at least as good as z. □

Lemma 3.1 and Proposition 3.2 suggest a procedure to solve R(PKP) explicitly. We will denote the optimal solution for R(PKP) by z^{R} and a feasible solution for the PKP by z^{IP}. We will add items to the knapsack while there is enough space and the objective function does not decrease, i.e., $P^\ell(1) \geq P^\ell(0)$. Let r be the last item added to the knapsack. If we stop due to infeasibility, then the critical item is $s = r+1$. Otherwise, the objective function decreases if item $r+1$ is completely added to the knapsack. Then, there are two possible cases. In the first case, the function P^r is an increasing function, thus the item $s = r+1$ is the first item which is not profitable. Otherwise, the maximum of the function P^r is attained at $\gamma \in (0,1)$, so this is the critical item, i.e. $s = r$. However, we only realize that when we try to add item $r+1$. More precisely, if $(P^r)'_-(1) \geq 0$ then $s = r+1$, otherwise $s = r$. Finally, it remains to evaluate the optimal fraction γ_s^* which can be found efficiently since it is the maximizer of a concave function (see Hiriart-Urruty and Lemaréchal [13]). We may observe that as a by-product we obtain a feasible solution z^{IP} for the PKP. We can set $z_j^{\mathrm{IP}} = 1$ for each $j = 1, \dots, r$, and $z_j^{\mathrm{IP}} = 0$ otherwise.

Recall that the items have been renumbered so that if $j < k$ then $\frac{p_j}{\omega_j} \geq \frac{p_k}{\omega_k}$.

Solving R(PKP)

Step 0. Set $J = \{1, \dots, n\}$. Set $z_j^{\mathrm{R}} = 0$, and $z_j^{\mathrm{IP}} = 0$ for each $j = 1, \dots, n$.

Step 1. Set $\hat{\jmath} = \arg\min\{j \in J\}$ and $J = J \setminus \{\hat{\jmath}\}$. If $\hat{\jmath}$ is not feasible then set

$$\begin{aligned} s &= \hat{\jmath} \\ z_s^{\mathrm{R}} &= \frac{\Omega - \sum_{j=1}^{s-1}\omega_j}{\omega_s}, \end{aligned}$$

and STOP.

Step 2. If $P^{\hat{\jmath}}(1) \geq P^{\hat{\jmath}}(0)$, set

$$\begin{aligned} z_{\hat{\jmath}}^{\mathrm{IP}} &= 1 \\ z_{\hat{\jmath}}^{\mathrm{R}} &= 1 \end{aligned}$$

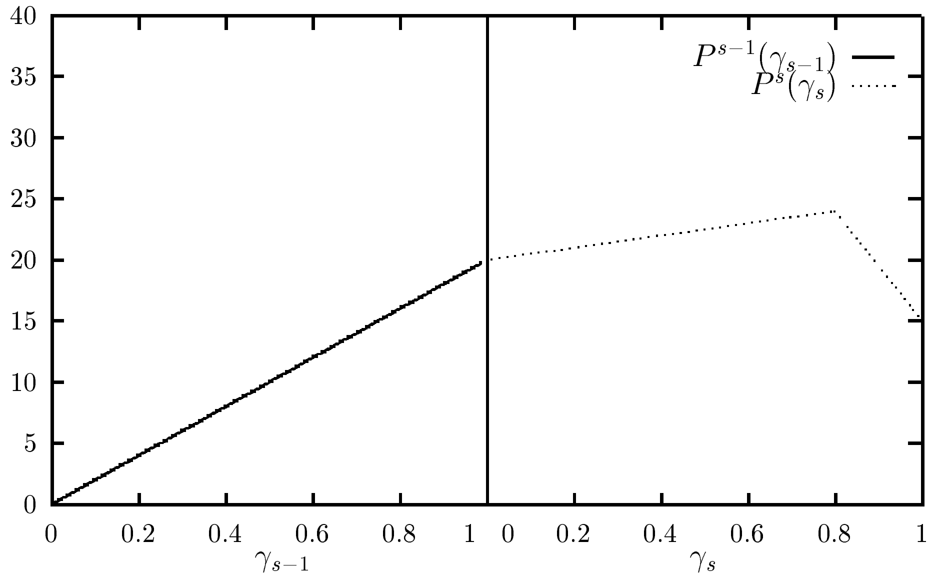

Figure 2: When $s = r + 1$

and go to Step 1. Else, if $(P^{\hat{\jmath}-1})'_-(1) \geq 0$ set $s = \hat{\jmath}$, else set $s = \hat{\jmath} - 1$. Set

$$z^{\mathrm{R}}_{\hat{\jmath}} = \arg\max_{\gamma \in [0,1]} P^s(\gamma),$$

and STOP.

Step 2 is illustrated by Figures 2 and 3. In the first case (see Figure 2), item $s-1$ was added completely since P^{s-1} is a strictly increasing function. However, the objective function drops from 20 to 15 by adding item s to the knapsack. Thus, this is the first item which is not profitable. However, in the second case (see Figure 3) the objective function increases from 20 to 25 by adding item s to the knapsack. Nevertheless, the maximum of the objective function is attained at $\gamma_s = 0.8$, so this is the critical item. However, we only realize that after we try to add item $s+1$.

4 The Multi-Period Single-Sourcing Problem

In this section we will introduce the notation of the MPSSP and we will show that it is a member of the class of convex assignment problems presented in Section 3.1.

Let n denote the number of customers, m the number of facilities, and T the planning horizon. The total demand of customer j throughout the planning horizon is given by d_j. The demand patterns over time of the customers are assumed to exhibit a common seasonality, represented by nonnegative seasonal factors σ_t for each $t = 1, \ldots, T$, satisfying $\sum_{t=1}^{T} \sigma_t = 1$. Thus, the demand of customer j in period t is equal to $\sigma_t d_j$. Let b_{it} denote the production capacity at facility i in period t. The costs of supplying customer j by facility i in period t are equal to c_{ijt}. The unit inventory holding costs at facility i in period t are given by h_{it}. (All parameters are nonnegative by definition.) For convenience, we

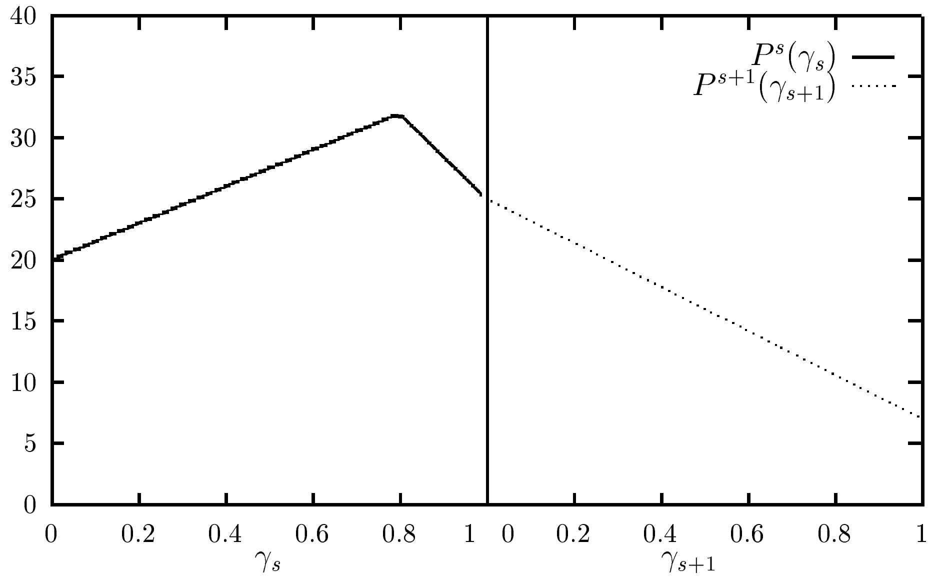

Figure 3: When $s = r$

assume that each warehouse has essentially unlimited physical and throughput capacity. In other words, we assume that its physical capacity is sufficient to be able to store the cumulative excess production of its corresponding plant, even if this plant produces to full capacity in each period. In addition, the throughput capacity is large enough for the warehouse to be able to supply any combination of customers assigned to it. However, these two types of capacity constraints can be easily added at little expense to the algorithm.

The MPSSP can be formulated as follows:

$$\text{minimize} \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijt}x_{ij} + \sum_{t=1}^{T}\sum_{i=1}^{m} h_{it}I_{it}$$

subject to $\qquad\qquad$ (P$_0$)

$$\begin{aligned}
\sigma_t \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{it} &\leq b_{it} + I_{i,t-1} && i=1,\ldots,m;\ t=1,\ldots,T && (3)\\
\sum_{i=1}^{m} x_{ij} &= 1 && j=1,\ldots,n\\
x_{ij} &\in \{0,1\} && i=1,\ldots,m;\ j=1,\ldots,n\\
I_{i0} &= 0 && i=1,\ldots,m\\
I_{it} &\geq 0 && i=1,\ldots,m;\ t=1,\ldots,T
\end{aligned}$$

where x_{ij} is equal to 1 if customer j is assigned to facility i and zero otherwise, and I_{it} represents the amount of product in storage at facility i at the end of period t. Hereafter $x \in \mathbb{R}^{mn}$ will denote the vector with components x_{ij} and similarly for $I \in \mathbb{R}^{mT}$.

Romeijn and Romero Morales [21] have shown for a variant of the MPSSP that the inventory variables can be eliminated, at the expense of introducing convexity in the objective function, i.e., an equivalent formulation with a convex objective function exists. In our case, this reformulation of the MPSSP yields a Single-Sourcing Problem (hereafter SSP) with convex objective function.

Proposition 4.1 *(P_0) can be equivalently reformulated as:*

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{t=1}^{T} c_{ijt}\right) x_{ij} + \sum_{i=1}^{m} H_i\left(\sum_{j=1}^{n} d_j x_{ij}\right)$$

subject to *(P)*

$$\begin{array}{rcll} \sum_{j=1}^{n} d_j x_{ij} & \leq & \min_{t=1,\ldots,T}\left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t}\sigma_\tau}\right) & i=1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} & = & 1 & j=1,\ldots,n \\ x_{ij} & \in & \{0,1\} & i=1,\ldots,m;\ j=1,\ldots,n \end{array}$$

where $H_i(u)$ is the convex function given by the optimal value of the following problem

$$\text{minimize} \sum_{t=1}^{T} h_{it} I_t$$

subject to

$$\begin{array}{rcll} I_t - I_{t-1} & \leq & b_{it} - \sigma_t u & t=1,\ldots,T \\ I_0 & = & 0 & \\ I_t & \geq & 0 & t=1,\ldots,T. \end{array}$$

Proof: Let F be the feasible region of (P_0). By decomposing (P_0), we obtain the following equality

$$\begin{aligned} & \min_{(x,I)\in F}\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijt}x_{ij} + \sum_{t=1}^{T}\sum_{i=1}^{m} h_{it}I_{it}\right) = \\ = \quad & \min_{x:\exists I'\,(x,I')\in F}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{t=1}^{T} c_{ijt}\right)x_{ij} + \min_{I:(x,I)\in F}\sum_{t=1}^{T}\sum_{i=1}^{m} h_{it}I_{it}\right) \\ = \quad & \min_{x:\exists I'\,(x,I')\in F}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{t=1}^{T} c_{ijt}\right)x_{ij} + H(x)\right) \end{aligned}$$

where $H(x)$ is equal to

$$\text{minimize} \sum_{i=1}^{m}\sum_{t=1}^{T} h_{it} I_t$$

subject to

$$
\begin{array}{rcll}
I_{it} - I_{i,t-1} & \leq & b_{it} - \sigma_t \cdot \sum_{j=1}^{n} d_j x_{ij} & i = 1, \ldots, m; t = 1, \ldots, T \\
I_{i0} & = & 0 & i = 1, \ldots, m \\
I_{it} & \geq & 0 & i = 1, \ldots, m; t = 1, \ldots, T.
\end{array}
$$

This problem is separable in i, and moreover for each $i = 1, \ldots, m$ it only depends on $\sum_{j=1}^{n} d_j x_{ij}$. Thus, $H(x) = \sum_{i=1}^{m} H_i \left(\sum_{j=1}^{n} d_j x_{ij} \right)$. Now we will show that the feasible region of the decomposed problem is equal to the feasible region of (P). Consider some x so that there exists a feasible solution (x, I) for (P_0). For each facility i, we aggregate the capacity constraints over all the periods. Then, we obtain

$$
\begin{array}{rcl}
\sum_{t=1}^{T} \left(\sigma_t \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{it} \right) & \leq & \sum_{t=1}^{T} (b_{it} + I_{i,t-1}) \\
\left(\sum_{t=1}^{T} \sigma_t \right) \cdot \sum_{j=1}^{n} d_j x_{ij} + \sum_{t=1}^{T} I_{it} & \leq & \sum_{t=1}^{T} b_{it} + \sum_{t=1}^{T} I_{i,t-1} \\
\left(\sum_{t=1}^{T} \sigma_t \right) \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{iT} & \leq & \sum_{t=1}^{T} b_{it} + I_{i0}
\end{array}
$$

which is equivalent to

$$
\left(\sum_{t=1}^{T} \sigma_t \right) \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{iT} \leq \sum_{t=1}^{T} b_{it}
$$

and this implies

$$
\left(\sum_{t=1}^{T} \sigma_t \right) \cdot \sum_{j=1}^{n} d_j x_{ij} \leq \sum_{t=1}^{T} b_{it}.
$$

The previous inequality shows that x is feasible for (P). Now, consider a feasible solution x to (P). Then, we know there exists a vector $y \in \mathbb{R}^{mT}$ so that

$$
y_{it} \leq b_{it} \qquad i = 1, \ldots, m; t = 1, \ldots, T
$$

and

$$
\sum_{t=1}^{T} y_{it} = \sum_{t=1}^{T} \sigma_t \sum_{j=1}^{n} d_j x_{ij} \qquad i = 1, \ldots, m
$$

(Note that y can be interpreted as a set of feasible production levels corresponding to (x, I) in the original three-level formulation of (P_0).) Now, define I_{it} as

$$
I_{it} = \sum_{\tau=1}^{t} y_{i\tau} - \left(\sum_{\tau=1}^{t} \sigma_\tau \right) \cdot \sum_{j=1}^{n} d_j x_{ij}
$$

for each $i = 1, \ldots, m$ and $t = 1, \ldots, T$. It is easy to see that I_{it} is nonnegative, and $(x, I) \in F$. This means that x is a feasible solution for the decomposed problem.

With respect to function $H_i(u)$ it is easy to see that has a finite value and thus by strong LP-duality we obtain

$$\begin{aligned} H_i(u) &= \min\left\{\sum_{t=1}^{T} h_{it} I_t : \; I_t - I_{t-1} \leq b_{it} - \sigma_t u, \; I_0 = 0, \; I_t \geq 0, \; t = 1, \ldots, T\right\} \\ &= \max\left\{\sum_{t=1}^{T} (\sigma_t u - b_{it}) w_t : \; w \in W_i\right\} \end{aligned}$$

where

$$W_i = \{w \in \mathbb{R}^T : -w_t + w_{t+1} \geq h_{it}, \, t = 1, \ldots, T-1; \; w_t \geq 0, \, t = 1, \ldots, T\}.$$

Now let $\mu \in [0, 1]$ and fix $u, u' \in \mathbb{R}$. Then

$$\begin{aligned} &\max\left\{\sum_{t=1}^{T} ((\mu u + (1-\mu)u')\sigma_t - b_{it}) w_t : \; w \in W_i\right\} \\ &= \max\left\{\mu \sum_{t=1}^{T} (\sigma_t u - b_{it}) w_t + (1-\mu) \sum_{t=1}^{T} (\sigma_t u' - b_{it}) w_t : \; w \in W_i\right\} \\ &\leq \mu \max\left\{\sum_{t=1}^{T} (\sigma_t u - b_{it}) w_t : \; w \in W_i\right\} + \\ &\quad (1-\mu) \max\left\{\sum_{t=1}^{T} (\sigma_t u' - b_{it}) w_t : \; w \in W_i\right\} \end{aligned}$$

which shows the convexity of $H_i(u)$. □

The function H_i calculates the minimal inventory costs at facility i needed to be able to supply the customers assigned to it. We may observe that the value of the inventory costs at each facility only depends on the total demand required by the customers assigned to it. The previous proposition tells us that the MPSSP belongs to the class of convex assignment problems introduced in Section 3.1 by choosing

$$\begin{aligned} g_i(z) &= \sum_{j=1}^{n} \left(\sum_{t=1}^{T} c_{ijt}\right) z_j + H_i\left(\sum_{j=1}^{n} d_j z_j\right) \qquad \text{for each } z \in \mathbb{R}^n \\ X_i &= \left\{z \in [0,1]^n \; : \; \sum_{j=1}^{n} d_j z_j \leq \min_{t=1,\ldots,T} \left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau}\right)\right\}. \end{aligned}$$

We know that function H_i is convex. In fact, it is easy to show that this function is also piecewise linear. This is illustrated by an example, where we will suppress the index i for convenience. Consider $n = 1$, $T = 3$, and

$$\begin{aligned}\sigma &= (1,1,1)^\top \\ h &= (2,2,2)^\top \\ d_1 &= 25 \\ b &= (50,20,10)^\top.\end{aligned}$$

In that case, we have that $H(z_1)$ is equal to the optimal value of

$$\text{minimize}\, 2(I_1 + I_2 + I_3)$$

subject to

$$\begin{aligned}I_1 - I_0 &\leq 50 - 25z_1 \\ I_2 - I_1 &\leq 20 - 25z_1 \\ I_3 - I_2 &\leq 10 - 25z_1 \\ I_0 &= 0 \\ I_t &\geq 0 \qquad\qquad t = 1,2,3.\end{aligned}$$

Figure 4 plots the optimal objective function value of its LP-relaxation as a function of the fraction z_1 of the item added to the knapsack. Thus, we observe that it is a piecewise linear function in the fraction z_1 added to the knapsack. Note that each breakpoint corresponds to a new inventory variable becoming positive. In this particular case, all inventory variables are equal to zero if the fraction of the demand supplied is below 0.4, i.e., $z_1 \in [0, 0.4]$. If $z_1 \in (0.4, 0.6]$, I_2 becomes positive. Finally, if $z_1 \in (0.6, 1]$, I_1 also becomes positive.

5 Testing the Branch and Price algorithm

5.1 Experimental design

The MPSSP is an $\mathcal{NP}$-*Hard* problem since for $T = 1$ we obtain the SSP, which is $\mathcal{NP}$-*Hard* (see Martello and Toth [15]). Moreover, the decision problem associated with the feasibility of the MPSSP is an $\mathcal{NP}$-*Complete* problem. Therefore, even to test whether a problem instance has at least one feasible solution is computationally hard. In this section we propose a stochastic model for the MPSSP where the tightness of the random instances can be controlled. The numerical results generated in Section 5.2 illustrate the hardness of those instances.

When testing a procedure the data generation schemes may introduce biases into the computational results, see Hall and Posner [12]. For the particular case of the GAP, Romeijn and Romero Morales [20] have proposed and studied a stochastic model. In this section we describe a general stochastic model for the MPSSP based on the stochastic model for the GAP. Consider the following probabilistic model for the parameters of the MPSSP. For each $j = 1, \ldots, n$, let (D_j, C_j) be i.i.d. random vectors in $[\underline{D}, \overline{D}] \times [\underline{C}, \overline{C}]^{mT}$ where $C_j = (C_{ijt})_{i=1,\ldots,m;\, t=1,\ldots,T}$. We assume that the vector (D_j, C_j) is absolutely

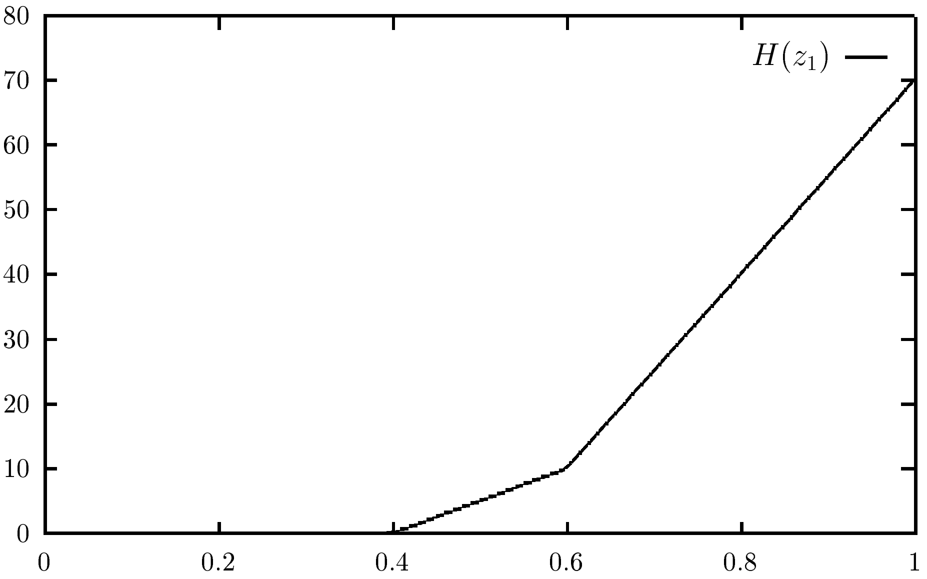

Figure 4: The inventory costs

continuous for each $j = 1, \ldots, n$. Furthermore, let b_{it} depend linearly on n, i.e., $b_{it} = \beta_{it} n$, for positive constants β_{it}. Observe that m and T are fixed, thus the size of the MPSSP only depends on the number of customers n.

In Proposition 4.1 we have shown that the MPSSP can be reformulated as a SSP with convex objective function. The SSP is a particular case of the GAP, and its feasibility is clearly not affected by the convexity of the objective function. We can thus apply the results found for the standard GAP by Romeijn and Piersma [18]. They observe that feasibility of the problem instances is not guaranteed under the above stochastic model, even for the LP-relaxation of the GAP, and find an implicit condition on the parameters of the GAP to ensure feasibility with probability one as the number of agents m grows to infinity. The following assumption ensures feasibility of the MPSSP with probability one as n goes to infinity.

Theorem 5.1 (cf. Romeijn and Piersma [18]) *As $n \longrightarrow \infty$, the MPSSP is feasible with probability one if*

$$\sum_{i=1}^{m} \min_{t=1,\ldots,T} \left(\frac{\sum_{\tau=1}^{t} \beta_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau} \right) > \mathcal{E}(D_1),$$

and infeasible with probability one if the inequality is reversed.

5.2 Computational results

The Branch and Price algorithm for the MPSSP has been tested on instances generated according to the stochastic model proposed in Section 5.1. Uniform coordinates in the square $[0, 10]^2$ were generated for facilities and customers. The total demand d_j was generated uniformly in $[5, 25]$. The seasonal factors σ_t were chosen as in Table 1. The costs of supplying customer j by facility i in period t, c_{ijt}, were calculated as the product of the Euclidean distance from customer j to facility i times the demand of customer j in period t. The unit inventory holding costs h_{it} were uniformly generated in $[10, 30]$. Finally, capacities were assumed to be equal for each facility and each period. According to Theorem 5.1, condition

$$\beta > \beta_{\min} \equiv \frac{15}{m} \max_{\tau=1,\ldots,T} \frac{1}{\tau} \sum_{t=1}^{\tau} \sigma_t$$

ensures feasibility of the instances with probability one when the number of customers goes to infinity. We have chosen $\beta = 1.1 \times \beta_{\min}$.

t	1	2	3	4	5	6
σ_t	$\frac{1}{9}$	$\frac{1}{6}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{1}{6}$	$\frac{1}{9}$

Table 1: Values of σ_t

We have run the greedy heuristic for (CAP) given in Section 2.2.2 to get an initial set of columns in the root node. We have chosen $f(i,j) = \sum_{t=1}^{T} c_{ijt} + \sum_{t=1}^{T} \lambda_{it}^{*} \sigma_t d_j$ where $\lambda_{it}^{*} \geq 0$ is the optimal dual multiplier of the capacity constraint (3) in the LP-relaxation of the MPSSP (where constraints (3) are rewritten in $\geq$-form to ensure that $\lambda \geq 0$). Romeijn and Romero Morales [21] have shown that this pseudo-cost function yields a heuristic that is asymptotically feasible and optimal with probability one for a variant of the MPSSP, and we conjecture that the same result holds for this greedy heuristic when applied to the variant of the MPSSP under consideration in this paper. We have already remarked that the greedy heuristic for (CAP) does not guarantee a feasible solution for the assignment constraints. A neighborhood search heuristic was implemented in a similar way as Romeijn and Romero Morales [21] for a variant of the MPSSP to try to assign the unassigned customers. A second neighborhood search heuristic was used to improve the objective value of the solution obtained by the greedy heuristic. The same procedure has been applied in each node of the tree with depth at most 10 to improve the best integer solution found by the Branch and Price algorithm.

When, for a given set of columns N, we do not have a certificate of optimality of the reduced problem LP(MP(N)) we search for columns pricing out for each facility. The procedure we follow for each facility is as follows. We first run a greedy heuristic for the PKP similar to the one proposed by Rinnooy Kan, Stougie, and Vercellis [17] for the Multi-Knapsack Problem. Let $\mu \in \mathbb{R}_{+}^{T}$, and let $p_j - \sum_{t=1}^{T} \mu_t \sigma_t d_j$ be a weight function measuring the value of adding item j to the knapsack. We order the set of items according to non-increasing value of the weight function. Each time an item is added to the knapsack, we calculate the remaining capacity. It could happen that some of the remaining items cannot (or should not) be added anymore because there is not enough capacity, or they are not profitable because the payment for using extra capacity is larger than the benefit of adding them to the knapsack. For all those items j which cannot be added the variables z_j are forced to 0. In the current implementation of the Branch and Price we have, for reasons of computational efficiency, simply chosen $\mu_t = 0$ for each $t = 1, \ldots, T$. When the obtained column does not price out, we use a Branch and Bound procedure for the PKP with depth-first search. We have branched on the variable equal to one from the optimal solution of the relaxation of the PKP (see Martello and Toth [15]). The relaxation of the PKP was solved explicitly as shown in Section 3.2.3. Without extra computational effort we were able to add more than one column pricing out. More precisely, we have added all columns pricing out from the sequence of improving solutions found in the tree.

With respect to the branching rule for (MP), we have chosen the variable x_{ij} which is closest to 0.5. Preliminary tests have indicated that this is a good choice.

To avoid a large number of columns in the model, we have included two types of deletions of columns. The first one concerns the new columns added to the model in each iteration of the column generation procedure. Each time that the number of columns added to the model is larger than η^1, we eliminate a fraction υ^1 of the ones with reduced costs larger than ζ_1. The second deletion affects all the columns in the model and works in a similar way. It is applied when the number of columns is larger than η^2_1, $\eta^2_2, \ldots$. In this paper we have chosen, $\eta^1 = 10$, $\eta^2_1 = 1000$, $\eta^2_2 = 2000$, $\ldots$, $\upsilon^1 = \upsilon^2 = 0.9$ and $\zeta_1 = \zeta_2 = 0.99$.

All the runs were performed on a PC with a 350 MHz Pentium II processor and 128 MB RAM. All LP-relaxations were solved using CPLEX 6.5 [6]. In contrast with most literature on column generation, we have compared the performance of our Branch and Price algorithm with the performance of the MIP solver from CPLEX applied to the standard formulation of the MPSSP. The objective value of the solution given by the greedy heuristic for (CAP) was given to CPLEX as an upper bound. Our computational experiences have shown us that both procedures find most of the times the optimal solution in an early stage, however to prove optimality can be very time consuming for some instances. Thus, our Branch and Price algorithm and CPLEX as a MIP solver, were stopped when the relative upper bound on the error of the best integer solution found was below 1%.

We have generated 50 random problem instances for each size of the problem. For all of them the number of periods T was fixed to 6. We have generated two classes of instances. In the first class we fix the ratio between the number of customers and the number of warehouses, and in the second one we fix the number of warehouses. Table 2 shows results of the performance of our Branch and Price algorithm and CPLEX as a MIP solver for $n/m = 5$, and similarly, Table 3 for $n/m = 10$, Table 4 for $m = 5$ and Table 5 for $m = 10$.

In the tables we have used the following notation. Column I indicates the size of the problem, in the format $m.n$, and column fI indicates the number of these instances that are feasible. Next, column f(h) tells us the number of times that the heuristic applied to the MPSSP could find a feasible solution, column f(r) is the number of times that we have a feasible solution in the root node, column f is the number of times that the Branch and Price algorithm could find a feasible solution for the problem, and column s is the number of instances that were solved successfully, i.e., either a solution with guaranteed error less than 1% was found, or the instance was shown to be infeasible. The following two columns give average results on the quality of the initial solutions: column er(h) is the average upper bound on the error of the initial solution given by the heuristic, and column er(r) gives the upper bound on the error of the solution obtained in the root node.

		B&P												CPLEX			
I	fI	f(h)	f(r)	f	s	er(h)	er(r)	#c	#n	nt	t(h)	t	t_r	f(c)	s(c)	t(c)	t_r(c)
2.10	41	39	41	41	50	1.29%	0.00%	44.78	0.74	50	0.01	0.06	0.05	41	50	0.06	0.05
3.15	41	40	40	41	50	2.46%	0.01%	110.46	0.82	49	0.02	0.11	0.10	41	50	0.14	0.12
4.20	43	41	43	43	50	3.97%	0.79%	211.36	1.16	44	0.04	0.26	0.23	43	50	3.49	0.37
5.25	38	37	37	38	50	1.93%	0.61%	237.14	1.26	39	0.10	0.56	0.45	38	50	8.44	1.00
6.30	46	43	44	46	50	2.45%	1.01%	395.36	3.10	34	0.08	1.37	0.87	46	47	240.20	6.99
7.35	49	44	44	49	50	2.40%	1.40%	498.12	4.42	24	0.11	2.18	1.60	49	46	356.65	30.10
8.40	48	48	48	48	50	3.53%	2.56%	591.50	4.44	21	0.13	2.94	2.44	48	48	448.01	52.09
9.45	49	46	47	49	50	3.67%	3.33%	872.26	23.46	9	0.18	16.23	6.58	48	41	776.39	478.71

Table 2: $n/m = 5$

		B&P												CPLEX			
I	fI	f(h)	f(r)	f	s	er(h)	er(r)	#c	#n	nt	t(h)	t	t_r	f(c)	s(c)	t(c)	t_r(c)
2.20	41	41	41	41	50	0.61%	0.00%	95.70	0.34	50	0.04	0.22	0.14	41	50	0.13	0.08
3.30	45	44	44	45	50	1.06%	0.13%	457.36	0.98	45	0.05	1.56	1.18	45	50	0.36	0.27
4.40	48	48	48	48	50	1.41%	0.31%	818.08	1.18	43	0.07	3.77	3.29	48	50	0.84	0.75
5.50	50	48	48	50	50	1.54%	1.00%	979.56	3.96	30	0.11	13.25	9.25	50	49	61.58	3.15
6.60	50	48	48	50	50	1.21%	0.93%	998.36	3.00	30	0.16	17.90	14.35	50	50	22.95	8.74
7.70	49	49	49	49	50	1.45%	1.15%	1274.70	5.32	25	0.23	35.04	28.39	49	50	37.78	15.99
8.80	50	50	50	50	50	1.86%	1.46%	1708.92	6.30	20	0.30	63.19	54.34	50	45	297.49	77.29
9.90	50	49	49	50	50	1.40%	1.16%	1924.38	6.60	25	2.15	99.98	74.96	50	45	359.16	194.94

Table 3: $n/m = 10$

		B&P												CPLEX			
I	fI	f(h)	f(r)	f	s	er(h)	er(r)	#c	#n	nt	t(h)	t	t_r	f(c)	s(c)	t(c)	t_r(c)
5.25	38	37	37	38	50	1.93%	0.61%	237.14	1.26	39	0.10	0.56	0.45	38	50	8.44	1.00
5.30	41	39	40	41	50	2.35%	0.83%	405.14	1.52	37	0.09	1.15	0.93	41	50	14.76	1.26
5.35	44	42	42	44	50	2.03%	1.09%	588.76	2.92	34	0.08	2.27	1.64	44	50	14.45	1.61
5.40	47	47	47	47	50	1.36%	0.57%	641.22	1.84	41	0.09	3.08	2.32	47	50	3.32	2.07
5.45	48	48	48	48	50	1.08%	0.34%	885.88	0.98	45	0.10	4.46	3.75	48	50	3.33	1.17
5.50	50	49	49	50	50	1.08%	0.53%	954.18	1.52	39	0.11	8.43	7.08	50	50	6.41	2.32
5.55	49	46	46	49	50	1.35%	0.66%	1334.34	4.28	31	0.13	22.25	16.12	49	50	90.93	3.81
5.60	50	50	50	50	50	0.61%	0.38%	1245.00	1.06	42	0.13	16.67	14.32	50	50	4.40	2.39
5.65	49	49	49	49	50	0.92%	0.51%	1308.68	1.26	40	0.15	26.90	21.77	49	50	2.66	2.32

Table 4: $m = 5$

		B&P												CPLEX			
I	fI	f(h)	f(r)	f	s	er(h)	er(r)	#c	#n	nt	t(h)	t	t_r	f(c)	s(c)	t(c)	t_r(c)
10.25	38	27	29	38	50	5.91%	1.75%	191.06	2.22	33	0.30	0.91	0.74	37	34	2345.88	1594.68
10.30	43	27	31	43	50	4.33%	1.77%	265.24	4.40	25	0.25	1.73	1.20	39	30	2819.48	1709.18
10.35	48	41	43	48	50	5.99%	4.24%	432.74	9.30	14	0.17	3.59	2.34	46	34	2233.45	1494.59
10.40	44	36	36	44	50	5.70%	3.85%	499.02	11.04	17	0.22	5.29	3.36	41	32	1621.04	1193.54
10.45	49	45	45	49	50	3.85%	3.38%	740.48	22.18	9	0.21	12.73	6.34	48	27	1696.40	1384.96
10.50	48	46	46	48	50	3.17%	2.67%	852.20	13.66	12	0.23	11.00	7.45	48	28	1844.87	1459.34
10.55	46	45	45	46	50	3.34%	3.09%	934.30	20.80	12	0.26	17.74	10.23	46	36	1125.14	842.64
10.60	47	45	45	47	50	2.82%	2.28%	1294.58	32.24	13	0.30	37.52	18.90	46	26	1420.94	1277.64
10.65	50	48	48	50	50	2.66%	2.57%	1201.68	25.96	6	0.30	38.69	25.53	49	34	891.78	743.60

Table 5: $m = 10$

The latter two averages have been calculated only taking into account the instances where a feasible solution was found. The following group of columns give information on the Branch and Price phase of the algorithm. Column #c is the average number of columns in the model at the end of the Branch and Price procedure, column #n is the average number of nodes inspected, and column nt shows us how many times the optimal solution of the MPSSP was found in the root node. The final columns pertaining to the Branch and Price algorithm deal with computation times. Column t(h) is the average time used by the heuristic for the MPSSP applied in the root node, and t is the average total time used by the Branch and Price procedure. To illustrate the stability of this average, we have also calculated the average time of the 45 fastest instances, see column t_r which eliminates the few instances that have an extreme effect on the average running time. Finally, the last results show the behaviour of CPLEX as a MIP solver. Column f(c) indicates the number of times that CPLEX could find a feasible solution, and column s(c) shows the number of times that CPLEX was successful (similar to column s above). Column t(c) is the average total time employed by CPLEX and column $t_r(c)$ is the average of the 45 fastest instances.

The main conclusion that we can draw from Tables 2-5 is that the Branch and Price algorithm is very well suited for solving the MPSSP problem studied in this paper, especially when the ratio between the number of customers and the number of warehouses is not too large. For large ratios the MIP solver in CPLEX is the more efficient solution approach to this MPSSP problem. The breakpoint lies somewhere between the ratios 5 and 10. In fact, CPLEX tends to become more efficient, even in an absolute sense, as the number of customers grows for a fixed number of warehouses. A possible explanation for this fact is that CPLEX, as well as the heuristic, seem to be able to take advantage of the fact that, with an increase in the number of customers, the number of feasible options for choosing which sets of customers to assign to a given warehouse also increases – not only due to the increasing number of customers, but also due to an increased flexibility in switching customers between warehouses. On the other hand, for the Branch and Price algorithm this increasing number of feasible assignments translates to an increase in the number of columns in the set partitioning problem (MP), and thus in the number of columns that may need to be generated in the column generation phase.

The second conclusion that can be drawn from the tables is that the Branch and Price algorithm is much more successful in solving the problems than CPLEX. In fact, the Branch and Price algorithm succeeded in finding a solution with an error of at most 1% or giving a certificate of infeasibility of the instance for all of the instances generated, while CPLEX often failed (due to a lack of memory) to solve the problem satisfactorily – especially for the larger instances, with failure rates up to 48% for instances with 10 warehouses.

Thirdly, the Branch and Price algorithm shows more stability in the computation times, caused by fewer and/or less extreme outliers.

Finally, Tables 4 and 5 support our conjecture that the heuristic is asymptotically feasible and optimal as the number of customers increases by showing an increasing number of feasible instances, and a decreasing error exhibited by the heuristic solution.

6 Concluding remarks

In this paper we have generalized a Branch and Price algorithm that was developed for the Generalized Assignment Problem (GAP) to a much richer class of problems, which we have called CAP (Convex Assignment Problems). The viability of this approach depends critically on the possibility of solving the pricing problem efficiently. We have identified an important subclass of problems, containing many variants of the multi-period single-sourcing problem (MPSSP), as well as some variants of the GAP, for which this is the case. We have applied the method to a particular variant of the MPSSP, and have shown that the Branch and Price algorithm is very useful for solving problems for which the ratio between the number of customers and the number of warehouses is at most 10. For these problems the Branch and Price algorithm is more successful in finding the optimal solution, the computation times are superior (or comparable for large ratios between the number of customers and the number of warehouses) to the computation times obtained using the MIP solver of CPLEX, and show greater stability, i.e., fewer and less extreme outliers are observed.

References

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[2] J.F. Benders, J.A. Keulemans, J.A.E.E. van Nunen, and G. Stolk. A decision support program for planning locations and allocations with the aid of linear programming. In C.B. Tilanus, O.B. de Gaus, and J.K. Lenstra, editors, *Quantitative Methods in Management: cases studies of failures and successes*, chapter 4, pages 29–34. John Wiley & Sons, Chichester, 1986.

[3] D.G. Cattrysse, M. Salomon, and L.N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, 72:167–174, 1994.

[4] L.M.A. Chan, A. Muriel, and D. Simchi-Levi. Supply-Chain Management: Integrating inventory and transportation. Research Report, Department of Industrial Engeneering and Management Sciences, Northwestern University, Evanston, Illinois, 1998.

[5] Z.-L. Chen and W.B. Powell. Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1):78–94, 1999.

[6] CPLEX Reference Manual. *ILOG CPLEX 6.5.* ILOG, Inc., Incline Village, Nevada, 1999.

[7] F. Duran. A large mixed integer production and distribution program. *European Journal of Operational Research*, 28:207–217, 1987.

[8] J.A. Ferland, A. Hertz, and A. Lavoie. An object-oriented methodology for solving assignment-type problems with neighborhood search techniques. *Operations Research*, 44(2):347–359, 1996.

[9] B. Fleischmann. Designing distribution systems with transport economies of scale. *European Journal of Operational Research*, 70:31–42, 1993.

[10] A. Geoffrion and G.W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, 20(5):822–844, 1974.

[11] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[12] N.G. Hall and M.E. Posner. Generating experimental data for scheduling problems. Technical report, The Ohio State University, 1996.

[13] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms: Fundamentals*, volume 1. Springer-Verlag, Berlin, 1993.

[14] S. Martello and P. Toth. An algorithm for the generalized assignment problem. In J.P. Brans, editor, *Operational Research*, pages 590–603, Amsterdam, 1981. IFORS, North–Holland.

[15] S. Martello and P. Toth. *Knapsack problems, algorithms and computer implementations.* John Wiley & Sons, New York, 1990.

[16] J.B. Mazzola and A.W. Neebe. Resource-constrained assignment scheduling. *Operations Research*, 34(4):560–572, 1986.

[17] A.H.G. Rinnooy Kan, L. Stougie, and C. Vercellis. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Applied Mathematics*, 42:279–290, 1993.

[18] H.E. Romeijn and N. Piersma. A probabilistic feasibility and value analysis of the generalized assignment problem. ERASM Management Report Series no. 293, Rotterdam School of Management, Erasmus University Rotterdam, 1996.

[19] H.E. Romeijn and D. Romero Morales. A class of greedy algorithms for the generalized assignment problem. ERASM Management Report Series no. 40(13), Rotterdam School of Management, Erasmus University Rotterdam, 1997. Forthcoming in *Discrete Applied Mathematics.*

[20] H.E. Romeijn and D. Romero Morales. Generating experimental data for the generalized assignment problem. ERASM Management Report Series no. 1998-9, Rotterdam School of Management, Erasmus University Rotterdam, 1998.

[21] H.E. Romeijn and D. Romero Morales. An asymptotically optimal greedy heuristic for the multi-period single-sourcing problem: the cyclic case. ERASM Management Report Series no. 20-1999, Rotterdam School of Management, Erasmus University Rotterdam, 1999.

[22] M.P.W. Savelsbergh. A branch–and–price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

[23] V. Srinivasan and G.L. Thompson. An algorithm for assigning uses to sources in a special class of transportation problems. *Operations Research*, 21:284–295, 1972.