A BRANCH-AND-PRICE ALGORITHM FOR THE MULTIPERIOD SINGLE-SOURCING PROBLEM

RICHARD FRELING

(deceased)

Formerly with Econometric Institute, Erasmus University Rotterdam, Rotterdam, The Netherlands, and ORTEC Consultants B.V., Gouda, The Netherlands

H. EDWIN ROMEIJN

Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611-6595, romeijn@ise.ufl.edu

DOLORES ROMERO MORALES

Saïd Business School, University of Oxford, Oxford OX 1HP, United Kingdom, dolores.romero-morales@sbs.ox.ac.uk

ALBERT P. M. WAGELMANS

Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands, wagelmans@few.eur.nl

In this paper, we propose a multiperiod single-sourcing problem (MPSSP), which takes both transportation and inventory into consideration, suitable for evaluating the performance of a logistics distribution network in a dynamic environment. We reformulate the MPSSP as a Generalized Assignment Problem (GAP) with a convex objective function. We then extend a branch-and-price algorithm that was developed for the GAP to this problem. The pricing problem is a so-called Penalized Knapsack Problem (PKP), which is a knapsack problem where the objective function includes an additional convex term penalizing the total use of capacity of the knapsack. The optimal solution of the relaxation of the integrality constraints in the PKP shows a similar structure to the optimal solution of the knapsack problem, that allows for an efficient solution procedure for the pricing problem. We perform an extensive numerical study of the branch-and-price algorithm.

(*Production/distribution: integrating production distribution and inventory. Integer programming: branch-and-price algorithm and penalized knapsack problem.*)

Received November 1999; revisions received December 2000, February 2002; accepted October 2002.

Area of review: Optimization.

1. INTRODUCTION

Some of the most important logistics problems faced by a supplier are the timing of production, the location of inventories, and the allocation of customers to warehouses. These decisions must not only be made separately, but must also be coordinated for the entire supply chain to perform efficiently. This coordination is especially required in dynamic environments where the settings of the logistics distribution network change significantly over the planning horizon. In this paper, we will study a dynamic model that can be used to support decisions concerning the allocation of demand and the location/allocation of inventories. This research was motivated by the problem faced by a large beer brewery that typically has to deal with a strongly seasonal demand. The demand for beer is heavily influenced by the weather, leading to a much higher request in warmer periods. Moreover, due to the strong variation of demand over time, the production capacity is usually not sufficient in periods of high demand, leading to the necessity of holding inventories. Thus, strong seasonal differences in demand require the supplier to use a dynamic approach to managing both production and inventory. In general, the model that we propose is also applicable to other producers of soft drinks and beers, or other products whose demand shows a strong seasonal component.

Most of the existing models in the literature for the allocation of demand are static (single period) in nature. This means that they are not able to explicitly model inventory decisions. Moreover, the implicit assumption in these models is that the environment, including all problem data, is constant over time. For instance, demand patterns are assumed to be constant, even though this is often an unrealistic assumption. Hence, the adequacy of those models is limited to situations where the demand pattern exhibits no remarkable changes throughout the planning horizon. In practice, it means that all (by nature dynamic) input parameters to the model are approximated by static ones like average parameters. Related literature focusing on static models can be found in Geoffrion and Graves (1974), Benders et al. (1986), and Fleischmann (1993). There are notable exceptions where the planning horizon is discretized. Duran (1987) plans the production, bottling, and distribution to agencies of different types of beer, with an emphasis on the production process. A one-year planning horizon is considered, but in contrast to most of the literature, the model is dynamic, with 12 monthly periods. Chan et al. (1998) study a dynamic, but uncapacitated, distribution problem in an operational setting.

The logistics distribution network we are considering consists of a set of facilities, each of which could be interpreted as a production plant with an associated warehouse and a set of customers. Each production plant has known, finite, and possibly time-varying capacity, and each customer needs to be served by (assigned to) a unique facility throughout the planning horizon, while the customer demands exhibit a seasonal pattern. We assume that each warehouse has essentially unlimited physical and throughput capacity; that is, 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. We assume that products can only be stored at the facilities; i.e., no storage is allowed at the customers. This is a realistic assumption for customers like small supermarkets or restaurants who have a very limited storage place available. The decisions that need to be made concern (i) the assignment of customers to facilities, and (ii) the location and size of inventories.

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 of the customers over time 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.)

The multiperiod single-sourcing problem (MPSSP) can be formulated as follows:

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

subject to (P$_0$)

$$\sigma_t \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{it} \leqslant b_{it} + I_{i,t-1}, \qquad i = 1, \ldots, m;\ t = 1, \ldots, T, \quad (1)$$

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

$$x_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, m;\ j = 1, \ldots, n,$$

$$I_{i0} = 0, \qquad i = 1, \ldots, m,$$

$$I_{it} \geqslant 0, \qquad i = 1, \ldots, m;\ t = 1, \ldots, T,$$

where x_{ij} is equal to one 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 $I = (I_{it}) \in \mathbb{R}^{mT}$.

The assignment and the inventory 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 MPSSP is a kind of Generalized Assignment Problem (GAP) with a nonlinear objective function. The GAP has been studied extensively in the literature from an algorithmic point of view. See Cattrysse and Van Wassenhove (1992) and Osman (1995) for reviews on solution methods for the GAP. In particular, Savelsbergh (1997) proposes a successful branch-and-price algorithm for the GAP. In this paper, we use a similar branch-and-price algorithm to solve the MPSSP. The structure of the pricing problem is a major issue in the success of a column generation procedure, and therefore of a branch-and-price algorithm. For the MPSSP, we show that the pricing problem is a so-called Penalized Knapsack Problem (PKP), which is a knapsack problem where the objective function includes an additional convex term penalizing the total use of capacity of the knapsack. The optimal solution of the relaxation of the integrality constraints in the PKP shows a similar structure to the optimal solution of the knapsack problem that allows for an efficient solution procedure for the pricing problem.

The outline of this paper is as follows. In §2 we reformulate the MPSSP as a GAP with a convex objective function. In §3 we propose an exact branch-and-price procedure for solving the MPSSP based on a column generation approach for a set-partitioning formulation of the problem. This approach generalizes the branch-and-price procedure for the GAP developed by Savelsbergh (1997). In §4 we present an efficient solution procedure for the relaxation of the PKP as well as a class of greedy heuristics. In §5 we show that this branch-and-price procedure can solve a broader class of problems. In §6 we illustrate the performance of this branch-and-price scheme. In §7 we end the paper with some concluding remarks.

2. A NONLINEAR GAP

Romeijn and Romero Morales (2003) have shown for another 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 GAP with a convex objective function.

PROPOSITION 2.1. (P$_0$) *is equivalent to*

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

subject to (P)

$$\sum_{j=1}^{n} d_j x_{ij} \leqslant \min_{t=1,\ldots,T}\left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau}\right), \qquad i = 1, \ldots, m,$$

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

$$x_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, m;\ j = 1, \ldots, n,$$

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

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{T} h_{it} I_t \\ \text{subject to} \quad & I_t - I_{t-1} \leqslant b_{it} - \sigma_t u, \qquad t = 1, \ldots, T, \\ & I_0 = 0, \\ & I_t \geqslant 0, \qquad t = 1, \ldots, T. \end{aligned}$$

PROOF. See the Appendix. □

The function H_i calculates the minimal inventory costs needed at facility i to be able to supply the customers assigned to it. We may observe that the value of the inventory costs at each facility depends only on the total demand required by the customers assigned to it. Proposition 2.1 now tells us that the MPSSP can be formulated as a GAP with a convex objective function. In this formulation, we have m agents and n tasks; the cost function associated with agent i is equal to

$$g_i(z) = \sum_{j=1}^{n} \left(\sum_{t=1}^{T} c_{ijt} \right) z_j + H_i \left(\sum_{j=1}^{n} d_j z_j \right) \qquad \text{for each } z \in \mathbb{R}^n$$

and its capacity constraint is equal to

$$\sum_{j=1}^{n} d_j x_{ij} \leqslant \min_{t=1,\ldots,T} \left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau} \right).$$

We may observe that the requirements of the tasks d_j are agent independent.

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 in which we will suppress the index i for convenience. Consider $n = 1$, $T = 3$, and

$$\sigma = (1, 1, 1)^\top,$$

$$h = (2, 2, 2)^\top,$$

$$d_1 = 25,$$

$$b = (50, 20, 10)^\top.$$

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

$$\begin{aligned} \text{minimize} \quad & 2(I_1 + I_2 + I_3) \\ \text{subject to} \quad & I_1 - I_0 \leqslant 50 - 25 z_1, \\ & I_2 - I_1 \leqslant 20 - 25 z_1, \\ & I_3 - I_2 \leqslant 10 - 25 z_1, \\ & I_0 = 0, \\ & I_t \geqslant 0, \qquad t = 1, 2, 3. \end{aligned}$$

Figure 1. The inventory costs.

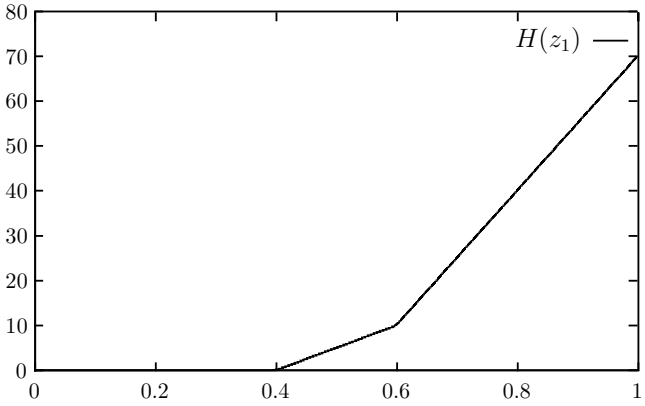


Figure 1 plots the optimal objective function value of its LP relaxation as a function of the fraction z_1 of the item added to the knapsack. Thus, we observe that it is a piecewise linear function in the fraction z_1 added to the knapsack. Note that each breakpoint corresponds to a new inventory variable 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.

3. A BRANCH-AND-PRICE ALGORITHM FOR THE MPSSP

3.1. A Set-Partitioning Formulation

In a similar way as was done for the GAP by Cattryse et al. (1994) and Savelsbergh (1997), (P) can be formulated as a set-partitioning problem. In particular, a feasible solution for (P) can be seen as a partition of the set of tasks $\{1, \ldots, n\}$ 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 tasks that can feasibly be assigned to agent i ($i = 1, \ldots, m$). Let $\alpha_{i\cdot}^l$ denote the lth subset (for fixed i); i.e., $\alpha_{ij}^l = 1$ if task j is an element of subset l for agent i, and $\alpha_{ij}^l = 0$ otherwise. We will call $\alpha_{i\cdot}^l$ the lth *column* for agent i. Then, the set-partitioning problem can be formulated as follows:

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

subject to (MP)

$$\sum_{i=1}^{m} \sum_{l=1}^{L_i} \alpha_{ij}^l y_i^l = 1, \qquad j = 1, \ldots, n, \tag{2}$$

$$\sum_{l=1}^{L_i} y_i^l = 1, \qquad i = 1, \ldots, m, \tag{3}$$

$$y_i^l \in \{0, 1\}, \qquad l = 1, \ldots, L_i;\ i = 1, \ldots, m,$$

where y_i^l is equal to one if column l is chosen for agent i, and zero otherwise. As mentioned by Barnhart et al. (1998),

the convexity constraint (3) for agent i, $i = 1, \ldots, m$, can be written as

$$\sum_{l=1}^{L_i} y_i^l \leqslant 1,$$

because $\alpha_{ij} = 0$ 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 programming 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 (P), R(P). A similar result was proved for the linear GAP by Savelsbergh (1997). Hence, if we let $v(\text{R(P)})$ and $v(\text{LP(MP)})$ denote the optimal objective values of R(P) and LP(MP), respectively, then the following holds.

PROPOSITION 3.1. *The following inequality holds*:

$$v(\text{R(P)}) \leqslant v(\text{LP(MP)}).$$

PROOF. Similar to the proof of this result for the linear GAP; see Savelsbergh (1997). □

This result suggests that the formulation (MP) is more attractive than (P) when solving the MPSSP by a branch-and-bound scheme based on lower bounds obtained by relaxing the integrality constraints. However, the number of columns associated with each agent in (MP) can be extremely large, thus prohibiting the explicit construction and solution of LP(MP). Therefore, this formulation is suitable for applying a column generation approach where new columns are generated only when needed; see Gilmore and Gomory (1961). When this procedure to calculate LP bounds is embedded in a branch-and-bound scheme, we obtain the so-called branch-and-price algorithm which solves to optimality the integer programming problem (MP).

3.2. Designing the Branch-and-Price Scheme

3.2.1. Introduction. As mentioned above, the tightness of LP(MP) makes a branch-and-price algorithm a promising optimization method to solve the MPSSP. There are several key features in the design of a branch-and-price scheme. First, when solving the LP bounds by a column generation procedure, a good starting set of columns can accelerate the convergence of this procedure. Second, the structure of the pricing problem is a major issue in the success of the column generation procedure. Typically, many columns need to be generated until optimality can be proven with the current set of columns. Therefore, the success of this procedure depends on how efficiently the pricing problem can be solved. Finally, it is known that standard branching rules are usually not appropriate for branch-and-price schemes; see, e.g., Vanderbeck (2000). They may destroy the structure of the pricing problem. In addition, they produce unbalanced branch-and-bound trees. In the following, we discuss these three features in more detail. Recall that the MPSSP has been reformulated as a GAP with a convex objective function, so the branch-and-price procedure described below resembles the one proposed by Savelsbergh (1997) for the linear GAP.

3.2.2. Initial Set of Columns. The column generation procedure calls for an initial set of columns to start with. For this purpose, we make use of the class of greedy heuristics proposed for Romeijn and Romero Morales (1999) for a larger class of multiperiod single-sourcing problems. These greedy heuristics are a generalization of the ones proposed by Martello and Toth (1981) for the GAP. The basic idea is that each possible assignment of a task to an agent is evaluated by a pseudo-cost function $f(i, j)$. The *desirability* of assigning a task is measured by the difference between the second smallest and the smallest values of $f(i, j)$. Assignments of tasks are made in decreasing order of this difference. Along the way, some agents will not be able to handle some of the tasks due to their capacity constraints, and consequently the values of the desirabilities will be updated, taking into account that the two most desirable agents for each task should be feasible.

The challenge is to specify a pseudo-cost function that will yield a good, or at least a feasible, solution to (P). Romeijn and Romero Morales (1999) propose a pseudo-cost function for which the greedy heuristic is asymptotically feasible and optimal in the probabilistic sense.

To be able to prove infeasibility in any of the nodes of the branch-and-price tree, we have added a dummy variable $s_j \geqslant 0$ to the jth constraint (2) with a high cost, for each $j = 1, \ldots, n$. This ensures that any LP relaxation always has a feasible solution, and infeasibility of this LP problem is characterized by the positiveness of some of the dummy variables in its optimal solution.

3.2.3. The Pricing Problem. A major issue in the success of the column generation approach is, of course, the viability to solve the pricing problem. In this section, we examine the structure of this problem. We suppose that N is the current set of columns in the reduced LP relaxation. Let LP(MP(N)) be the corresponding reduced LP relaxation, and $(u^*(N), \delta^*(N))$ an optimal dual vector to LP(MP(N)). When needed, columns are searched for each agent. The pricing problem for agent i, $i = 1, \ldots, m$, is equal to

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{n}\left(\sum_{t=1}^{T} c_{ijt}\right) z_j + H_i\left(\sum_{j=1}^{n} d_j z_j\right) \\
& \qquad - \sum_{j=1}^{n} u_j^*(N) z_j + \delta_i^*(N) \\
\text{subject to} \quad & \sum_{j=1}^{n} d_j z_j \leqslant \min_{t=1,\ldots,T}\left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau}\right) \\
& z_j \in \{0, 1\}, \qquad j = 1, \ldots, n.
\end{aligned}$$

Without loss of optimality, we can leave out the constant term $\delta_i^*(N)$. After rearranging terms and transforming it

into a maximization problem, the pricing problem associated with agent i reads

$$\begin{aligned}\text{maximize} \quad & \sum_{j=1}^{n}\left(u_j^*(N)-\sum_{t=1}^{T}c_{ijt}\right)z_j - H_i\left(\sum_{j=1}^{n} d_j z_j\right)\\ \text{subject to} \quad & \sum_{j=1}^{n} d_j z_j \leqslant \min_{t=1,\ldots,T}\left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t}\sigma_\tau}\right)\\ & z_j \in \{0,1\}, \qquad j=1,\ldots,n.\end{aligned}$$

The feasible region of this problem is described by a knapsack constraint. Similarly to the knapsack problem, items which are added to the knapsack yield a profit $u_j^*(N)-\sum_{t=1}^{T}c_{ijt}$. However, in contrast with the ordinary problem, the utilization of the knapsack is penalized by the convex function H_i. We will call this problem the PKP, and it will be analyzed in §4.

3.2.4. Branching Rule. If the optimal solution of the LP relaxation of (MP) is not integer, we need to branch to obtain an optimal integer solution. Because the LP relaxation of (MP) has been solved by a column generation procedure, it is unlikely that all columns are present in the final reduced LP problem. Thus, by using a branch-and-bound scheme with only the columns generated, we will, in the best case, end up with only a feasible solution for (MP). This approach yields a heuristic for solving the MPSSP.

If we want a certificate of optimality, new columns (when needed) should be generated when branching. The choice of the branching rule is crucial because 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 (1997) 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, it can be proved that each feasible solution y for (MP) has a corresponding feasible solution x for (P). 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 MPSSPs. Thus, the column generation procedure in each node of the tree is the same as in the root node.

4. THE PENALIZED KNAPSACK PROBLEM

4.1. 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 $\omega_j \geqslant 0$, and the profit associated with adding item j to the knapsack is equal to $p_j \geqslant 0$. 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:

$$\begin{aligned}\text{maximize} \quad & \sum_{j=1}^{n} p_j z_j - G\left(\sum_{j=1}^{n}\omega_j z_j\right)\\ \text{subject to} \quad & \sum_{j=1}^{n}\omega_j z_j \leqslant \Omega,\\ & z_j \in \{0,1\}, \qquad j=1,\ldots,n.\end{aligned}$$

Because all items with zero space requirement will definitely be added to the knapsack and $p_j \geqslant 0$, 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 because 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., $G(u) \leqslant 0$ 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 (1990). 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 deleted. This is illustrated with an example in §4.2.

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

$$u+\omega_j \leqslant \Omega,$$

and that it is *profitable* if

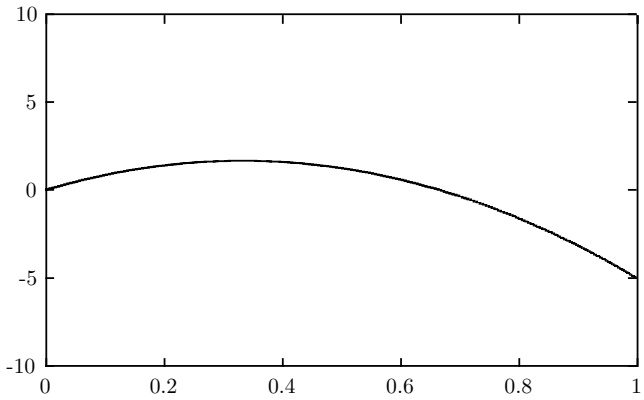
$$p_j - G(u+\omega_j) = \max_{\gamma\in[0,1]}\{p_j\gamma - G(u+\omega_j\gamma)\}.$$

4.2. 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 problem instance of the PKP reads

$$\begin{aligned}\text{maximize} \quad & 10z_1 - 15z_1^2\\ \text{subject to} \quad & 20z_1 \leqslant 25,\\ & z_1 \in \{0,1\}.\end{aligned}$$

Figure 2. Value of the relaxation R(PKP).



The item is feasible because 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 2 plots the value of its relaxation R(PKP). We may observe that the maximum of this function is attained at $z_1^* = 1/3$ even though, as we have seen before, the item can feasibly be added to the knapsack.

4.3. 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 linear knapsack problem (see Martello and Toth 1990), and can be solved explicitly as well.

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

LEMMA 4.1. *$P^l(\cdot)$ is a concave function.*

PROOF. The result follows by observing that, for each $\gamma \in [0, 1]$, we have that

$$P^l(\gamma) = \sum_{j=1}^{l-1} p_j + p_l\gamma - G\left(\sum_{j=1}^{l-1} \omega_j + \omega_l\gamma\right),$$

which is a concave function in γ. □

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

$$k_1 = \min\left\{l = 1, \ldots, n\colon \sum_{j=1}^{l} \omega_j > \Omega\right\},$$

$$k_2 = \min\left\{l = 1, \ldots, n\colon (P^l)'_-(1) < 0\right\}.$$

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 first item which will not be added completely to the knapsack due to the penalization of the capacity utilization. Now define item k as

$$k = \min\{k_1, k_2\},$$

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 $1, \ldots, k-1$ and the feasible and profitable fraction γ^* of item k, i.e.,

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

where

$$\gamma_1^* = \frac{\Omega - \sum_{j=1}^{k-1} \omega_j}{\omega_k},$$

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

PROPOSITION 4.2. *The vector $\bar{z} \in \mathbb{R}^n$ defined by*

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

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 $j = 1, \ldots, k-1$ and $\hat{z}_j = 0$ for each $j = k+1, \ldots, n$. By the definition of item k and fraction γ^*, $\bar{z}$ is at least as good as $\hat{z}$. Thus, the desired result follows.

Suppose that there exists an item $r = 1, \ldots, k-1$ so that $z_r < 1$. If $z_q = 0$ for each $q = k, \ldots, n$, we can construct a better solution by increasing z_r to 1 because the first $k-1$ items are feasible and profitable. Thus, assume that there exists $q = k, \ldots, n$ 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 because

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

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

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

Lemma 4.1 and Proposition 4.2 suggest a procedure to solve R(PKP) explicitly. We denote the optimal solution for

R(PKP) by z^{R}. We will add items to the knapsack while there is enough space and the objective function does not decrease; i.e., $P^l(1) \geqslant P^l(0)$. Let r be the last item added to the knapsack. If we stop due to infeasibility, then the critical item is $k = r+1$. 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 $k = r+1$ 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., $k = r$. However, we only realize that when we try to add item $r+1$. More precisely, if $(P^r)'_-(1) \geqslant 0$, then $k = r+1$, otherwise $k = r$. Finally, it remains to evaluate the optimal fraction γ^*, which can be found efficiently because it is the maximizer of a concave function (see Hiriart-Urruty and Lemaréchal 1993). 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 $j = 1, \ldots, r$, and $z_j^{\mathrm{IP}} = 0$ otherwise.

Recall that the items have been renumbered so that if $j < l$, then, $p_j/\omega_j \geqslant p_l/\omega_l$.

Solving R(PKP)

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

Step 1. Set $\hat{\jmath} = \arg\min\{j \in J\}$ and $J = J \setminus \{\hat{\jmath}\}$. If $\hat{\jmath}$ is not feasible, then set $k = \hat{\jmath}$ and go to Step 3.

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

$$z_{\hat{\jmath}}^{\mathrm{R}} = 1$$

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

Step 3. Set

$$z_k^{\mathrm{R}} = \min\left\{\frac{\Omega - \sum_{j=1}^{k-1} \omega_j}{\omega_k}, \arg\max_{\gamma \in [0,1]} P^k(\gamma)\right\}.$$

Step 2 is illustrated by Figures 3 and 4. In the first case (see Figure 3), item $k-1$ was added completely because P^{k-1} is a strictly increasing function. However, the objective function drops from 8 to 7.81 by adding item k to the knapsack. Thus, this is the first item which is not profitable.

Figure 3. When $k = r+1$.

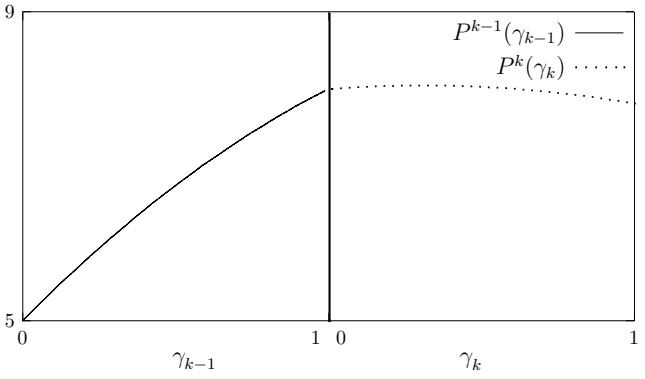
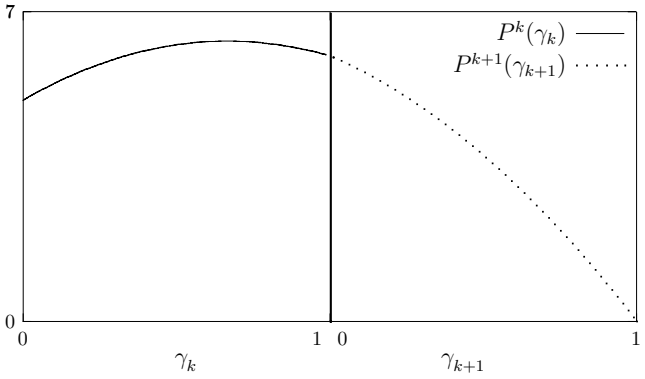


Figure 4. When $k = r$.



However, in the second case (see Figure 4), the objective function increases from 5 to 6 by adding item k to the knapsack. Nevertheless, the maximum of the objective function is attained at $\gamma_k = 2/3$, so this is the critical item. However, we only realize that after we try to add item $k+1$.

4.4. A Class of Greedy Heuristics

In this section, we describe a class of greedy heuristics for the PKP similar to the class proposed by Rinnooy Kan et al. (1993) for the multiknapsack problem. Let $\pi(j)$ be a weight function measuring the value of adding item j to the knapsack. We order the set of items according to nonincreasing 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. The variables z_j corresponding to those items are set to zero.

Greedy Algorithm for the PKP

Step 0. Set $\mathcal{L} = \{j\colon j = 1, \ldots, n\}$ and $z_j^{\mathrm{G}} = 0$ for each $j = 1, \ldots, n$.

Step 1. If $\mathcal{L} = \varnothing$: STOP, z^{G} is a feasible solution for the PKP. Otherwise, if there exists $j \in \mathcal{L}$ so that j is not feasible or profitable $\mathcal{L} = \mathcal{L} \setminus \{j\}$ and repeat Step 1.

Step 2. Let $\hat{\jmath} \in \arg\min_{j \in \mathcal{L}} \pi(j)$. Set

$$z_{\hat{\jmath}}^{\mathrm{G}} = 1,$$
$$\mathcal{L} = \mathcal{L} \setminus \{\hat{\jmath}\},$$

and go to Step 1.

5. A BROADER CLASS OF PROBLEMS

The branch-and-price algorithm presented above can solve a larger class of problems. We may observe that the fact that the requirements of the tasks in (P) are agent independent has not been explicitly used. Therefore, this solution procedure is still applicable to any GAP with a convex objective function. This includes some extensions of the

GAP like the GAP with fixed-charge costs (see Neebe and Rao 1983), where each agent processing at least one task incurs a fixed cost, and the GAP with flexible capacities (see Srinivasan and Thompson 1972), where additional capacity can be used at a cost which is linear in the amount of capacity added. Moreover, any variant of the MPSSP that can be reformulated as a GAP with a convex objective function can also be solved using this branch-and-price scheme. Romero Morales (2000) has studied the addition of several relevant constraints when evaluating the performance of a logistics distribution network to the MPSSP. In particular, she analyzes the addition of throughput capacity, physical capacity, and perishability constraints. The last type of constraints is suitable to model a limited shelf life of the product. This can be caused by the fact that the product exhibits a physical perishability or it might be affected by an economic perishability (obsolescence). In both cases, the storage duration of the product should be limited. For the MPSSP with these additional constraints, she obtains a similar reformulation to the one given in Proposition 2.1 for the MPSSP. Therefore, these variants of the MPSSP can be rewritten as a GAP with a convex objective function.

6. TESTING THE BRANCH-AND-PRICE ALGORITHM

6.1. Experimental Design

The MPSSP is an $\mathcal{NP}$-*Hard* problem because for $T = 1$ we obtain the GAP with agent-independent requirements, which is $\mathcal{NP}$-*Hard*; see Martello and Toth (1990). 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 problem instances can be controlled. The numerical results generated in §6.3 illustrate the hardness of those problem instances.

When testing a procedure the choice of data generation scheme may introduce a bias into the computational results; see Hall and Posner (2001). For the particular case of the GAP, Romeijn and Romero Morales (2001) 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 $C_j = (C_{ijt})_{i=1,\ldots,m;\, t=1,\ldots,T}$. 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, and therefore the size of the MPSSP only depends on the number of customers n.

In Proposition 2.1 we have shown that the MPSSP can be reformulated as a GAP with agent-independent requirements and a convex objective function. Its feasibility is clearly not affected by the convexity of the objective function. Therefore, we can apply the results found for the linear GAP by Romeijn and Piersma (2000). They observe that the 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. When the requirements are agent independent they derive an easy-to-check feasibility condition. The following assumption ensures feasibility of the MPSSP with probability one as n goes to infinity.

In the following theorem, $\mathcal{E}(\cdot)$ denotes expectation.

THEOREM 6.1 (CF. ROMEIJN AND PIERSMA 2000). *As $n \to \infty$, 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.

Therefore, the following assumption ensures feasibility of the MPSSP with probability one as n goes to infinity.

ASSUMPTION 6.2. *Assume that*

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

Now consider the greedy heuristic for the MPSSP given in §3.2.2 with the pseudo-cost function $f^*(i, j) = \sum_{t=1}^{T} c_{ijt} + \sum_{t=1}^{T} \lambda_{it}^* \sigma_t d_j$, where $\lambda^* \in \mathbb{R}_+^{mT}$ is the optimal dual subvector corresponding to the capacity constraints (1) in the LP relaxation of the MPSSP. If the random vectors (D_j, C_j), $j = 1, \ldots, n$, are distributed according to an absolutely continuous probability distribution and $\underline{D} > 0$, this greedy heuristic is asymptotically feasible and optimal with probability one.

THEOREM 6.3 (CF. ROMEIJN AND ROMERO MORALES 1999). *Under Assumption* 6.2*, the greedy heuristic given in §*3.2.2 *with the pseudo-cost function f^* is asymptotically feasible and optimal with probability one.*

6.2. Description of the Implementation

We have run the greedy heuristic for the MPSSP given in §3.2.2 with the pseudo-cost function f^* defined in the previous section to get an initial set of columns in the root node. The greedy heuristic for the MPSSP does not guarantee a feasible solution for the assignment constraints. A neighborhood search heuristic was implemented in a way similar to Romeijn and Romero Morales (2003) 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 of 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 the greedy heuristic for the PKP proposed in §4.4. Recall that the value of adding item j to the knapsack is measured by a weight function $\pi(j)$. In the current implementation of the branch-and-price algorithm we have, for reasons of computational efficiency, simply chosen $\pi(j) = p_j$. 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 (1990). The relaxation of the PKP was solved explicitly as shown in §4.3. 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_1^2, $\eta_2^2, \ldots$. In this paper, we have chosen $\eta^1 = 10$, $\eta_1^2 = 1{,}000$, $\eta_2^2 = 2{,}000, \ldots$, $\upsilon^1 = \upsilon^2 = 0.9$ and $\zeta_1 = \zeta_2 = 0.99$.

All the runs were performed on a PC with a 550 MHz Pentium III processor and 256 MB RAM. All LP relaxations were solved using CPLEX 6.6 (CPLEX Reference Manual 1999).

6.3. Computational Results

6.3.1. Introduction. The branch-and-price algorithm for the MPSSP has been tested on problem instances generated according to the stochastic model proposed in §6.1. We have investigated the behavior of this algorithm on four types of problem instances depending on how the locations of the facilities and the customers are generated, and the choice made for the seasonal factors. In §6.3.2, these locations are uniformly distributed. In §6.3.3, the locations of the facilities are still uniformly distributed, but the customers are clustered around the facilities. The locations of the customers in the cluster associated with facility i are generated from a truncated bivariate normal distribution centered at this facility. For both types of problem instances we propose two vectors of seasonal factors, which differ with respect to the timing of the peak and nonpeak seasons. We have chosen the first vector equal to

$$\sigma^{(1)} = \left(\tfrac{1}{2}, \tfrac{3}{4}, 1, 1, \tfrac{3}{4}, \tfrac{1}{2}\right)^\top$$

and the second one to

$$\sigma^{(2)} = \left(1, \tfrac{3}{4}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{3}{4}, 1\right)^\top.$$

For all problem instances, the total demand d_j was generated uniformly in [5, 25]. 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 6.1, condition

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

then ensures feasibility of the problem instances with probability one when the number of customers goes to infinity. To allow for a reasonable number of feasible instances for finitely many customers, we have chosen $\beta = 1.1 \cdot \beta_{\min}$. For the demand pattern given by $\sigma^{(1)}$, this yields, for $i = 1, \ldots, m$,

$$\beta_{it} = \beta_{\min}^{(1)} = 1.1 \cdot \frac{15}{m} \cdot \frac{13}{16} \qquad \text{for } t = 1, \ldots, T,$$

which means that the aggregate capacity is about 89% of the peak aggregate expected demand. In contrast, for the demand pattern given by $\sigma^{(2)}$, the peak demand is in the first period, which would yield an aggregate capacity level exceeding the peak expected demand by 10%, and therefore give very loosely capacitated problem instances. For the GAP, this has been shown to make the problem instances (much) easier (see Romeijn and Romero Morales 2001). The cause of this unexpected behavior of the required capacities is the assumption that initial (as well as ending) inventories are equal to zero, which seems unrealistic from a practical point of view for this demand pattern. Therefore, we have assumed that initial inventory levels for $\sigma^{(2)}$ allow for an aggregate capacity level that is equal to the capacity level that was obtained for $\sigma^{(1)}$. The two problem classes indeed differ only in the *timing* of the optimization model with respect to the demand pattern, as was intended. Note that an initial inventory level is in fact equivalent to an increased capacity level in period 1, which we then accompany by a similarly decreased capacity level in period T (corresponding to a ending inventory level equal to the initial inventory level, thereby preparing for a new demand cycle). Using Theorem 6.1, we then obtain, for $i = 1, \ldots, m$,

$$\beta_{i1}^{(2)} = \frac{15}{m} \cdot \left(1.1 \cdot \frac{13}{16} + \frac{3}{16}\right),$$

$$\beta_{it}^{(2)} = 1.1 \cdot \frac{15}{m} \cdot \frac{13}{16}, \qquad \text{for } t = 2, \ldots, T-1.$$

$$\beta_{iT}^{(2)} = \frac{15}{m} \cdot \left(1.1 \cdot \frac{13}{16} - \frac{3}{16}\right).$$

Figure 5. Aggregate capacities for the two vectors of expected demands.

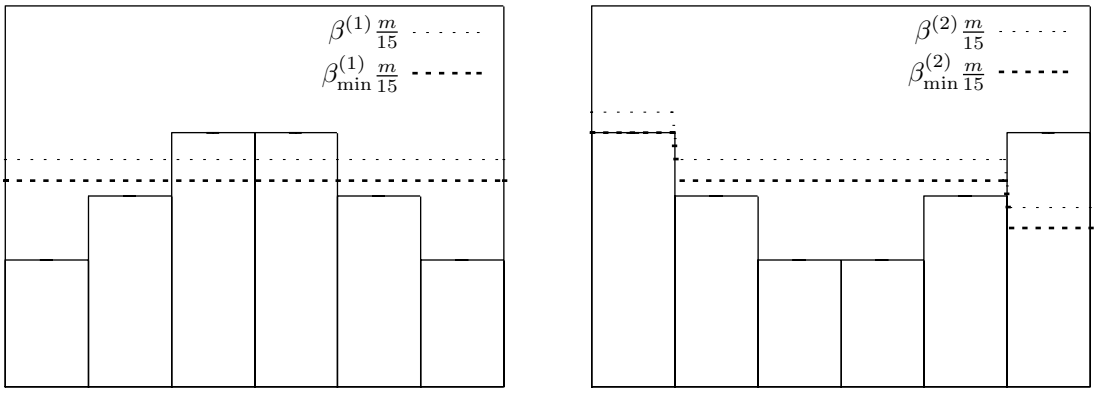


Figure 5 illustrates the two demand patterns with the corresponding (effective) capacities, as well as the minimal capacities that ensure asymptotic feasibility of the problem instances. Note that for $\sigma^{(1)}$ the capacities seem to become important earlier in the planning period than for $\sigma^{(2)}$. We will discuss the computational implications of this later in this section.

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 six, which is, for example, adequate for a yearly planning environment. We have generated two types of problem instances. In the first type we fix the ratio between the number of customers and the number of facilities, and in the second one we fix the number of facilities. In a tactical setting like ours, the problem instances encountered in practice are of small or medium size. Therefore, we have chosen $n/m = 5$, $n/m = 10$, $m = 5$, and $m = 10$.

6.3.2. Uniform Locations. Uniform coordinates in the square $[0, 10]^2$ were generated for facilities and customers. We have compared the performance of our branch-and-price algorithm with the performance of the MIP solver of CPLEX applied to the standard formulation of the MPSSP, (P_0). When the greedy heuristic for the MPSSP found a feasible solution, its objective value was given to the MIP solver of CPLEX as an upper bound. Our computational experiments show that both procedures are usually able to find the optimal solution at an early stage. However, the proof of optimality can be very time consuming for some problem instances. We therefore decided to stop our branch-and-price algorithm and the MIP solver of CPLEX as soon as the relative error was guaranteed to be below 1%.

Table 1 shows results of the performance of our branch-and-price algorithm and the MIP solver of CPLEX for $n/m = 5$ and

$$\sigma^{(1)} = \left(\tfrac{1}{2}, \tfrac{3}{4}, 1, 1, \tfrac{3}{4}, \tfrac{1}{2}\right)^{\top},$$

and similarly, Table 2 for $n/m = 10$, Table 3 for $m = 5$, and Table 4 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 problem 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 problem instances that were solved successfully; i.e., either a solution with guaranteed error less than 1% was found, or the problem 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

Table 1. $n/m = 5$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	41	39	41	50	1.28	0.00	44.18	0.74	50	0.03	0.04	0.02	41	50	0.03	0.03
3.15	41	40	40	41	50	2.46	0.01	111.22	0.82	49	0.01	0.07	0.06	41	50	0.08	0.07
4.20	43	43	41	43	50	3.97	0.79	220.86	1.12	44	0.02	0.17	0.14	43	50	2.33	0.22
5.25	38	37	37	38	50	1.93	0.61	244.46	1.30	39	0.04	0.28	0.24	38	50	4.86	0.58
6.30	47	44	43	47	50	2.45	1.01	432.10	3.98	33	0.05	1.19	0.60	46	48	442.38	4.91
7.35	49	44	44	49	50	2.40	1.40	540.54	4.78	24	0.07	1.53	1.16	49	48	504.33	25.07
8.40	48	48	48	48	50	3.53	2.56	643.18	4.84	21	0.09	2.10	1.73	48	49	331.47	34.31
9.45	49	47	46	49	50	3.67	3.33	897.40	16.66	9	0.12	7.96	4.92	48	42	1,676.78	832.56

Table 2. $n/m = 10$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	93.94	0.34	50	0.03	0.10	0.06	41	50	0.04	0.03
3.30	45	44	44	45	50	1.06	0.13	488.42	0.98	45	0.02	0.89	0.66	45	50	0.19	0.13
4.40	48	48	48	48	50	1.41	0.31	803.70	1.22	43	0.04	2.22	1.85	48	50	0.45	0.39
5.50	50	48	48	50	50	1.54	1.00	951.88	3.08	30	0.07	7.43	5.48	50	50	14.48	1.38
6.60	50	48	48	50	50	1.21	0.93	1,055.52	3.32	30	0.11	12.31	8.99	50	50	24.42	3.54
7.70	49	49	49	49	50	1.45	1.15	1,405.22	5.68	25	0.15	25.68	21.11	49	50	16.43	8.58
8.80	50	50	50	50	50	1.86	1.46	1,833.54	7.62	20	0.20	43.92	36.41	50	47	593.36	64.08
9.90	50	49	49	50	50	1.40	1.16	2,042.56	5.04	25	0.27	58.04	50.05	50	48	429.11	116.70

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. The latter two averages have been calculated taking into account only the problem instances where a feasible solution was found. The following group of columns gives 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 a solution of the MPSSP with an error of at most 1% 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 problem instances (see column t_r), which eliminates the few problem instances that have an extreme effect on the average running time. Finally, the last results show the behavior of the MIP solver of CPLEX. Column $f(c)$ indicates the number of times that the MIP solver of CPLEX could find a feasible solution, and column $s(c)$ shows the number of times that the MIP solver of CPLEX was successful (similar to column s above). Column $t(c)$ is the average total time employed by the MIP solver of CPLEX, and column $t_r(c)$ is the average of the 45 fastest problem instances.

The main conclusion that we can draw from Tables 1–4 is that the branch-and-price algorithm is very well suited for solving the MPSSP studied in this paper, especially when the ratio between the number of customers and the number of facilities is not too large. Looking at Tables 3 and 4, we see that for large ratios (the breakpoint lying somewhere between 5 and 10) the MIP solver of CPLEX is the more efficient solution approach to the MPSSP. Note, however, that we use the greedy heuristic for the MPSSP which is a part of our branch-and-price algorithm to provide CPLEX with a rather good upper bound on the optimal solution value to the problem. Without this bound, CPLEX is not competitive at all with the branch-and-price algorithm. The MIP solver of CPLEX tends to become more efficient, even in an absolute sense, as the number of customers grows for a fixed number of facilities. A possible explanation for this observation is that the MIP solver of 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 facility also increases—not only due to the increasing number of customers, but also due to an increased flexibility in switching customers between facilities. 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 the MIP solver of

Table 3. $m = 5$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	244.46	1.30	39	0.04	0.28	0.24	38	50	4.86	0.58
5.30	41	40	39	41	50	2.35	0.83	417.96	1.72	37	0.04	0.65	0.50	41	50	9.51	0.75
5.35	44	42	42	44	50	2.03	1.09	610.66	3.44	34	0.05	1.45	0.98	44	50	8.23	0.86
5.40	47	47	47	47	50	1.36	0.57	688.52	2.32	41	0.05	2.14	1.47	47	50	1.71	0.95
5.45	48	48	48	48	50	1.08	0.34	918.68	0.98	45	0.06	2.61	2.31	48	50	0.85	0.59
5.50	50	49	49	50	50	1.08	0.53	1,026.00	1.44	39	0.07	5.28	4.40	50	50	3.25	1.07
5.55	49	46	46	49	50	1.35	0.66	1,392.94	5.28	31	0.08	14.99	10.37	49	50	13.69	2.57
5.60	50	50	50	50	50	0.61	0.38	1,265.96	1.14	42	0.08	10.94	9.25	50	50	1.42	0.84
5.65	49	49	49	49	50	0.92	0.51	1,351.04	1.26	40	0.10	18.13	14.80	49	50	1.04	0.89

Table 4. $m = 10$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	39	31	27	39	50	5.91	1.63	194.38	2.22	34	0.11	0.44	0.35	37	36	3,854.65	2,028.20
10.30	44	31	27	44	50	4.33	1.77	280.68	4.36	24	0.15	1.29	0.72	39	31	3,217.24	2,018.45
10.35	49	43	41	49	50	5.99	4.24	461.70	12.34	13	0.11	3.30	1.68	46	38	2,624.63	1,535.20
10.40	44	36	36	44	50	5.70	3.85	541.16	12.72	17	0.15	3.82	2.14	42	35	2,746.86	1,813.68
10.45	49	45	45	49	50	3.85	3.38	822.30	29.26	9	0.13	9.46	5.06	48	37	2,934.15	2,181.72
10.50	49	46	46	49	50	3.17	2.67	915.24	17.74	11	0.15	21.08	5.54	48	41	2,118.18	1,458.52
10.55	46	45	45	46	50	3.34	3.09	909.74	18.32	12	0.17	10.94	7.11	46	41	2,046.76	1,282.02
10.60	48	45	45	48	50	2.82	2.28	1,331.78	50.84	12	0.20	58.44	12.64	46	38	2,277.30	1,549.03
10.65	50	48	48	50	50	2.66	2.57	1,179.12	25.92	6	0.20	26.42	18.57	49	40	1,949.97	1,281.23

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 problem instances generated, while the MIP solver of CPLEX often failed (due to a lack of memory) to solve the problem satisfactorily—especially for the larger problem instances, with failure rates up to 38% for problem instances with 10 facilities.

Third, the branch-and-price algorithm shows more stability in the computation times, caused by fewer and/or less extreme outliers.

Finally, Tables 3 and 4 illustrate the asymptotic feasibility and optimality of the greedy heuristic for the MPSSP with the pseudo-cost function f^* by showing, as the number of customers increases, an increasing number of feasible problem instances and a decreasing error exhibited by the heuristic solution.

Tables 5–8 show the corresponding results for

$$\sigma^{(2)} = \left(1, \tfrac{3}{4}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{3}{4}, 1\right)^{\top}.$$

We can draw similar conclusions about the performance of our branch-and-price algorithm and the MIP solver of CPLEX for this vector of seasonal factors. The branch-and-price algorithm outperforms the MIP solver of CPLEX when the ratio n/m is not too large. We may observe that the breakpoint is smaller, and closer to five. However, as for $\sigma^{(1)}$, the branch-and-price algorithm is more successful (in terms of being able to find a solution with an error of at most 1% or give a certificate of infeasibility) in solving the MPSSP than the MIP solver of CPLEX. In addition, our procedure is again more stable than the MIP solver of CPLEX. From the tables, we can see that the number of outliers with respect to the computation times is smaller, but they have become even more extreme.

The computation times of the branch-and-price algorithm and the MIP solver of CPLEX are closer to each other for $\sigma^{(2)}$ than for $\sigma^{(1)}$. A possible explanation for this can be the shape of the demand pattern. As we noted above, we can see from Figure 5 that, for $\sigma^{(2)}$, the first period where the expected demand is below the aggregate capacity is the last one, while for $\sigma^{(1)}$ this already happens in the third and fourth periods. Therefore, the need for holding inventory arises later for $\sigma^{(2)}$ than for $\sigma^{(1)}$. This implies that the number of feasible options for choosing a set of customers to assign to a given facility increases due to the lesser importance of the capacity constraints. As discussed above, for the branch-and-price algorithm this increased number of feasible assignments may translate into higher computational times. We conclude that the branch-and-price approach becomes more attractive as the capacity constraints become more important.

A remarkable observation from the results is that the solution obtained by the greedy heuristic is of much higher quality for the seasonal factors $\sigma^{(2)}$ than it is for $\sigma^{(1)}$. The average errors are smaller by a factor of up to two, which, for instance, for $n/m = 10$ corresponds to an average upper bound on the error in the greedy solution that is below 1%. This clearly leads to an improvement in the performance of both our branch-and-price algorithm and the MIP solver

Table 5. $n/m = 5$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	37	37	36	37	50	0.87	0.01	32.80	0.56	50	0.01	0.03	0.02	37	50	0.04	0.03
3.15	41	41	40	41	50	0.66	0.19	98.98	0.92	46	0.01	0.06	0.05	41	50	0.07	0.06
4.20	41	41	37	41	50	2.12	1.01	176.20	1.32	40	0.03	0.15	0.12	41	50	0.17	0.13
5.25	37	36	35	37	50	1.19	0.48	207.02	1.10	42	0.05	0.21	0.17	37	50	0.24	0.20
6.30	44	41	37	44	50	1.92	0.94	368.22	3.30	34	0.05	0.66	0.42	44	50	1.61	0.84
7.35	46	43	43	46	50	1.86	1.27	501.12	3.40	25	0.07	1.13	0.76	45	49	422.61	1.93
8.40	47	44	44	47	50	1.82	1.46	606.58	4.86	24	0.09	1.72	1.20	47	50	140.87	4.75
9.45	47	44	42	47	50	1.79	1.37	757.38	7.18	24	0.13	3.17	1.69	47	49	473.02	8.39

Table 6. $n/m = 10$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	40	40	40	40	50	0.28	0.00	73.76	0.32	50	0.01	0.08	0.04	40	50	0.05	0.05
3.30	40	40	40	40	50	0.39	0.05	156.28	0.32	49	0.03	0.25	0.12	40	50	0.13	0.10
4.40	47	45	45	47	50	0.53	0.26	520.22	0.92	44	0.05	1.12	0.86	47	50	0.42	0.28
5.50	46	46	45	46	50	0.80	0.55	961.24	1.66	39	0.07	4.42	3.70	46	50	0.99	0.81
6.60	43	41	41	43	50	0.61	0.46	1,213.42	2.96	42	0.12	11.75	5.16	43	50	23.75	1.59
7.70	48	48	47	48	50	0.92	0.89	1,491.94	3.52	29	0.15	15.08	12.52	48	50	8.41	3.40
8.80	48	46	46	48	50	0.77	0.77	1,655.80	4.28	33	0.20	22.70	17.87	48	50	17.36	8.48
9.90	49	48	48	49	50	0.96	0.90	2,258.70	7.80	28	0.26	59.29	41.42	48	49	491.26	19.93

of CPLEX in terms of computation times and success in solving the MPSSP, and shows the power of the greedy heuristic as a potential solution approach in itself and as a generator of high-quality initial feasible solutions for an exact solution method.

6.3.3. Clustered Locations. In this section, we test the branch-and-price algorithm and the MIP solver of CPLEX on a set of clustered problem instances. All the data is generated as in §6.3.2 except for the location of the customers. In this type of problem instances, we associate a cluster of customers with each facility so that the sizes of the clusters are (almost) the same. The locations of the customers in the cluster associated with facility i are generated from a bivariate normal distribution centered at this facility, with standard deviation equal to two, and truncated to the square $[0, 1]^2$. Other values for the standard deviation yield a similar behavior of the branch-and-price algorithm.

We first tested the problem instances with the vector of seasonal factors $\sigma^{(1)}$. The results, shown in Tables 9–12, suggest that the performance of the branch-and-price algorithm is quite robust with respect to the distribution of the customers and facilities in the plane. The computation times are of the same order; a solution with an error of at most 1% or a certificate of infeasibility is still obtained for all of the problem instances, and, finally, there are no remarkable outliers with respect to the computation times. One difference is that the quality of the solutions given by the heuristic and in the root node are worse for clustered instances, in particular for smaller problem instances.

The behavior of the MIP solver of CPLEX is also similar for both clustered and uniform problem instances. Therefore, as for the uniform problem instances, the branch-and-price algorithm outperforms the MIP solver of CPLEX when the ratio n/m is not too large. Moreover, our procedure has a higher success rate in solving the MPSSP and is more stable in terms of computation times. For $m = 10$, it seems that for small values of n, the success of the MIP solver of CPLEX in solving the MPSSP is larger for clustered instances than for uniform instances.

Finally, Tables 13–16 illustrate the results for the vector of seasonal factors $\sigma^{(2)}$. The conclusions are similar to the conclusions for uniform instances. Our branch-and-price algorithm is more stable, but the improvement with respect to the MIP solver of CPLEX is less pronounced than for $\sigma^{(1)}$. The performance of both our procedure and the MIP solver of CPLEX improves because of the better upper bound given by the greedy heuristic.

7. CONCLUDING REMARKS

In this paper, we have generalized a branch-and-price algorithm that was developed for the Generalized Assignment Problem (GAP) to a multiperiod single-sourcing problem (MPSSP). The viability of this approach depends critically on the possibility of solving the pricing problem efficiently, which is the case for the MPSSP. We 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 facilities is at most 10. For these

Table 7. $m = 5$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	37	36	35	37	50	1.19	0.48	207.02	1.10	42	0.05	0.21	0.17	37	50	0.24	0.20
5.30	38	37	36	38	50	1.13	0.49	301.84	1.00	46	0.06	0.32	0.26	38	50	0.35	0.29
5.35	43	42	42	43	50	0.93	0.47	538.02	2.06	40	0.05	0.82	0.64	43	50	0.67	0.42
5.40	47	46	44	47	50	0.66	0.33	543.14	1.32	45	0.06	1.18	0.82	47	50	0.84	0.42
5.45	48	48	46	48	50	0.50	0.26	739.74	0.82	47	0.06	1.70	1.49	48	50	1.01	0.54
5.50	47	47	47	47	50	0.68	0.44	814.52	1.06	41	0.07	3.13	2.69	47	50	0.80	0.62
5.55	46	46	45	46	50	0.68	0.46	1,145.60	1.60	42	0.08	6.95	5.16	46	50	5.31	1.04
5.60	50	50	50	50	50	0.51	0.21	1,226.86	0.66	48	0.08	8.29	6.22	50	50	1.14	0.91
5.65	49	49	49	49	50	0.23	0.23	725.76	0.44	47	0.09	6.39	3.51	49	50	0.96	0.86

Table 8. $m = 10$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	36	31	25	36	50	4.43	1.66	181.90	3.96	37	0.13	0.56	0.31	36	48	6,569.57	4.42
10.30	39	29	24	39	50	3.20	1.85	246.82	3.84	27	0.16	1.01	0.58	39	48	3,963.89	12.91
10.35	46	41	39	46	50	3.53	2.46	405.16	7.28	19	0.12	1.45	0.94	46	50	214.40	27.42
10.40	40	33	32	40	50	2.14	1.55	449.28	3.98	27	0.16	3.23	1.05	40	48	1,890.62	12.10
10.45	48	42	40	48	50	1.98	1.75	687.32	10.84	19	0.15	11.73	1.96	46	48	1,395.52	26.58
10.50	47	44	44	47	50	2.12	1.76	860.10	10.14	17	0.15	5.00	3.56	47	49	1,561.11	21.82
10.55	45	41	40	45	50	1.87	1.75	922.06	8.22	18	0.19	5.42	4.02	45	50	612.20	30.69
10.60	45	44	44	45	50	1.42	1.29	876.60	3.54	28	0.19	4.46	3.73	45	50	57.75	17.84
10.65	48	47	47	48	50	1.42	1.28	1,099.98	10.76	23	0.20	12.97	6.67	48	50	74.38	14.32

Table 9. Clustered problem instances with $n/m = 5$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	35	35	34	35	50	1.42	0.00	34.28	0.58	50	0.01	0.03	0.02	35	50	0.04	0.03
3.15	39	38	36	39	50	2.55	0.03	111.80	0.96	49	0.02	0.08	0.06	39	50	0.16	0.10
4.20	40	40	40	40	50	4.50	1.59	205.06	1.60	41	0.03	0.18	0.14	40	50	0.24	0.19
5.25	40	39	38	40	50	3.40	1.05	287.10	1.60	38	0.04	0.33	0.26	40	50	3.93	0.48
6.30	46	45	44	46	50	3.24	1.57	412.24	3.00	31	0.05	0.72	0.54	46	50	18.88	3.84
7.35	48	45	44	48	50	6.50	4.54	587.86	5.72	15	0.07	1.64	1.27	48	50	70.45	20.64
8.40	48	44	44	48	50	4.66	2.56	757.66	29.12	22	0.09	7.00	1.64	47	47	352.63	46.91
9.45	48	44	44	48	50	4.22	3.76	834.34	18.96	11	0.12	6.61	3.80	47	44	1,037.57	300.23

Table 10. Clustered problem instances with $n/m = 10$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	1.35	0.00	137.92	0.50	50	0.01	0.12	0.09	41	50	0.05	0.04
3.30	45	45	44	45	50	1.10	0.12	492.34	0.82	48	0.03	0.90	0.65	45	50	0.17	0.18
4.40	45	44	43	45	50	1.49	0.30	910.62	1.42	43	0.05	2.85	1.99	45	49	49.22	0.32
5.50	49	49	48	49	50	2.44	0.78	1,197.84	2.38	32	0.07	6.90	5.98	49	50	4.06	1.43
6.60	47	45	45	47	50	1.74	1.41	1,312.06	4.46	23	0.11	14.64	11.74	47	50	25.04	4.09
7.70	50	49	49	50	50	1.82	1.49	1,792.42	16.48	20	0.14	36.53	20.01	50	50	45.12	17.36
8.80	50	49	49	50	50	1.87	1.58	2,029.74	13.44	12	0.20	55.08	39.57	49	49	237.53	68.40
9.90	50	50	50	50	50	2.17	1.78	2,263.60	11.76	15	0.25	78.56	62.03	50	48	822.93	250.88

Table 11. Clustered problem instances with $m = 5$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	40	39	38	40	50	3.40	1.05	287.10	1.60	38	0.04	0.33	0.26	40	50	3.93	0.48
5.30	44	44	44	44	50	3.31	0.96	472.08	3.98	39	0.04	0.90	0.51	44	50	4.47	0.82
5.35	49	48	48	49	50	1.79	0.74	633.14	1.68	35	0.05	1.14	1.01	49	50	16.87	0.74
5.40	48	48	48	48	50	2.42	1.37	919.46	3.94	30	0.05	2.78	2.02	48	50	1.91	0.92
5.45	47	46	44	47	50	2.32	0.98	1,058.28	4.70	33	0.07	4.59	3.15	47	50	2.40	1.02
5.50	48	48	48	48	50	2.23	1.16	1,170.84	2.76	29	0.07	7.81	6.35	48	50	4.82	1.95
5.55	50	49	49	50	50	1.49	0.72	1,554.62	2.66	33	0.08	12.06	9.96	50	50	2.70	1.50
5.60	50	50	50	50	50	1.65	0.83	1,811.54	2.18	33	0.08	18.84	16.20	50	50	2.24	1.57
5.65	48	48	48	48	50	1.65	1.14	2,375.66	8.48	27	0.10	47.15	31.10	48	50	6.41	3.33

Table 12. Clustered problem instances with $m = 10$ and $\sigma = \sigma^{(1)}$.

		Branch-and-Price												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	42	38	31	42	50	10.96	5.09	188.82	2.18	33	0.10	0.40	0.32	41	43	2,539.15	648.30
10.30	47	37	34	47	50	6.70	4.35	306.88	6.14	18	0.11	1.09	0.73	46	41	3,387.31	1,131.31
10.35	47	40	38	47	50	7.01	3.97	410.36	13.66	22	0.12	6.33	1.03	46	47	788.03	369.11
10.40	48	43	42	48	50	5.39	3.52	542.64	7.88	20	0.12	2.49	1.75	46	43	1,901.77	873.49
10.45	47	41	40	47	50	4.32	3.62	709.82	11.58	10	0.15	5.33	3.40	44	40	2,509.11	1,344.43
10.50	48	46	45	48	50	4.61	3.53	929.18	23.00	12	0.15	9.97	5.60	46	38	3,502.91	2,131.26
10.55	49	44	44	49	50	4.84	3.75	1,072.54	19.86	8	0.17	12.83	9.21	48	27	6,158.84	4,771.04
10.60	48	45	45	48	50	4.04	3.59	1,128.30	32.24	8	0.19	21.55	12.21	46	33	3,807.03	2,680.91
10.65	50	48	48	50	50	4.26	3.45	1,246.96	24.84	8	0.19	24.86	15.79	50	40	3,138.35	2,044.33

Table 13. Clustered problem instances with $n/m = 5$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	33	33	32	33	50	1.72	0.00	29.36	0.54	50	0.02	0.02	0.02	33	50	0.03	0.03
3.15	35	35	32	35	50	1.90	0.30	90.34	0.82	47	0.02	0.08	0.06	35	50	0.13	0.07
4.20	40	39	38	40	50	2.01	0.80	164.94	1.54	43	0.03	0.21	0.10	40	50	1.56	0.11
5.25	38	36	34	38	50	1.78	0.44	227.62	1.32	45	0.05	0.26	0.15	38	50	1.61	0.22
6.30	44	41	40	44	50	2.18	1.20	349.34	1.96	32	0.05	0.49	0.37	44	50	1.81	0.70
7.35	44	40	40	44	50	2.54	1.73	505.02	4.04	24	0.07	1.03	0.74	44	50	196.09	2.54
8.40	45	42	42	45	50	1.87	1.26	532.50	2.78	30	0.09	1.05	0.80	45	50	5.56	2.18
9.45	44	40	40	44	50	2.15	1.56	648.56	9.40	22	0.13	3.81	1.46	43	48	1,339.99	5.69

Table 14. Clustered problem instances with $n/m = 10$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	39	39	39	39	50	0.40	0.00	58.32	0.24	50	0.01	0.05	0.03	39	50	0.05	0.05
3.30	42	42	42	42	50	0.33	0.11	183.64	0.42	49	0.02	0.29	0.17	42	50	0.12	0.10
4.40	42	42	42	42	50	1.20	0.33	724.12	1.04	45	0.05	1.49	1.29	42	50	0.33	0.29
5.50	44	42	42	44	50	0.57	0.22	663.64	0.82	45	0.08	1.97	1.61	44	50	0.70	0.50
6.60	44	43	43	44	50	0.93	0.70	1,120.50	2.14	36	0.11	7.72	5.42	44	50	106.93	1.16
7.70	48	48	48	48	50	1.16	0.95	1,617.96	2.26	31	0.14	14.00	12.53	48	50	4.20	2.69
8.80	49	48	48	49	50	0.59	0.52	1,272.14	1.78	41	0.19	12.76	9.72	49	47	10.36	3.32
9.90	50	49	49	50	50	1.28	1.16	2,492.78	13.10	24	0.24	62.02	38.42	50	48	51.43	20.33

Table 15. Clustered problem instances with $m = 5$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	36	34	38	50	1.78	0.44	227.62	1.32	45	0.05	0.26	0.15	38	50	1.61	0.22
5.30	43	42	40	43	50	1.76	1.34	385.50	2.34	34	0.05	0.57	0.38	43	50	2.82	0.38
5.35	48	46	45	48	50	1.63	0.93	576.60	1.86	34	0.05	1.07	0.79	48	50	57.36	0.47
5.40	47	44	43	47	50	1.27	0.54	722.60	1.98	39	0.06	1.64	1.09	47	50	5.37	0.57
5.45	43	42	41	43	50	1.26	0.81	805.98	2.28	37	0.07	5.90	1.85	42	49	222.06	0.52
5.50	47	47	47	47	50	1.15	0.80	1,036.78	2.14	32	0.07	4.40	3.98	47	50	1.34	0.81
5.55	49	48	47	49	50	0.94	0.60	1,263.78	1.36	38	0.08	6.80	5.75	49	50	7.63	0.68
5.60	47	47	47	47	50	0.60	0.33	1,415.88	1.00	45	0.09	8.28	7.08	47	50	0.94	0.76
5.65	47	47	47	47	50	0.85	0.60	1,639.02	2.00	36	0.09	17.93	15.18	47	50	1.32	0.88

Table 16. Clustered problem instances with $m = 10$ and $\sigma = \sigma^{(2)}$.

		Branch-and-Price												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	40	32	25	40	50	4.86	2.56	178.86	2.12	33	0.12	0.43	0.48	40	49	536.19	4.15
10.30	46	32	29	46	50	4.33	1.91	280.06	4.96	25	0.13	0.98	1.09	46	47	4,087.92	11.17
10.35	45	34	33	45	50	3.66	2.38	368.90	6.90	20	0.13	1.50	1.67	45	50	325.62	6.70
10.40	46	41	39	46	50	3.26	2.37	531.14	13.08	19	0.13	3.70	4.11	45	48	1,376.77	33.37
10.45	43	38	38	43	50	3.14	2.70	598.72	8.90	16	0.15	3.10	3.45	43	50	222.91	21.59
10.50	46	41	41	46	50	2.41	1.77	791.50	16.32	19	0.16	5.88	6.53	46	50	446.38	17.31
10.55	47	44	44	47	50	1.85	1.60	1,090.88	53.74	23	0.17	51.46	57.18	45	48	2,726.09	24.60
10.60	46	42	42	46	50	1.64	1.44	918.70	6.60	27	0.19	5.41	6.01	45	49	671.94	13.32
10.65	50	48	47	50	50	2.02	1.64	1,197.14	32.44	15	0.19	21.96	24.40	49	48	1,576.24	49.73

problems, the branch-and-price algorithm is more successful in finding a solution with an error of at most 1%, the computation times are superior (or comparable for large ratios between the number of customers and the number of facilities) to the computation times obtained using the MIP solver of CPLEX, and show greater stability, i.e., fewer and less extreme outliers are observed.

APPENDIX

PROPOSITION 2.1. (P_0) *is equivalent to*

$$\begin{aligned}
&\text{minimize} && \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{t=1}^{T} c_{ijt}\right)x_{ij} + \sum_{i=1}^{m} H_i\left(\sum_{j=1}^{n} d_j x_{ij}\right) \\
&\text{subject to} && \qquad\qquad (P)\\
&\sum_{j=1}^{n} d_j x_{ij} \leqslant \min_{t=1,\ldots,T}\left(\frac{\sum_{\tau=1}^{t} b_{i\tau}}{\sum_{\tau=1}^{t}\sigma_\tau}\right), && i = 1,\ldots,m,\\
&\sum_{i=1}^{m} x_{ij} = 1, && j = 1,\ldots,n,\\
&x_{ij} \in \{0,1\}, && i = 1,\ldots,m;\ j = 1,\ldots,n,
\end{aligned}$$

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

$$\begin{aligned}
&\text{minimize} && \sum_{t=1}^{T} h_{it} I_t\\
&\text{subject to} && I_t - I_{t-1} \leqslant b_{it} - \sigma_t u, && t = 1,\ldots,T,\\
& && I_0 = 0,\\
& && I_t \geqslant 0, && t = 1,\ldots,T.
\end{aligned}$$

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

$$\begin{aligned}
&\text{minimize} && \sum_{i=1}^{m}\sum_{t=1}^{T} h_{it} I_t\\
&\text{subject to} && I_{it} - I_{i,t-1} \leqslant b_{it} - \sigma_t \cdot \sum_{j=1}^{n} d_j x_{ij},\\
& && \qquad\qquad i = 1,\ldots,m;\ t = 1,\ldots,T,\\
& && I_{i0} = 0, \qquad i = 1,\ldots,m,\\
& && I_{it} \geqslant 0, \qquad i = 1,\ldots,m;\ t = 1,\ldots,T.
\end{aligned}$$

This problem is separable in the index i, and moreover for each $i = 1,\ldots,m$, it depends only on $\sum_{j=1}^{n} d_j x_{ij}$. Thus, $H(x) = \sum_{i=1}^{m} H_i(\sum_{j=1}^{n} d_j x_{ij})$. 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 first t capacity constraints, for each $t = 1,\ldots,T$. Then, we obtain

$$\begin{aligned}
&\sum_{\tau=1}^{t}\left(\sigma_\tau \cdot \sum_{j=1}^{n} d_j x_{ij} + I_{i\tau}\right) \leqslant \sum_{\tau=1}^{t}(b_{i\tau} + I_{i,\tau-1}),\\
&\left(\sum_{\tau=1}^{t}\sigma_\tau\right)\cdot\sum_{j=1}^{n} d_j x_{ij} + \sum_{\tau=1}^{t} I_{i\tau} \leqslant \sum_{\tau=1}^{t} b_{i\tau} + \sum_{\tau=1}^{t} I_{i,\tau-1},\\
&\left(\sum_{\tau=1}^{t}\sigma_\tau\right)\cdot\sum_{j=1}^{n} d_j x_{ij} + I_{it} \leqslant \sum_{\tau=1}^{t} b_{i\tau} + I_{i0},
\end{aligned}$$

which is equivalent to

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

and this implies

$$\left(\sum_{\tau=1}^{t}\sigma_\tau\right)\cdot\sum_{j=1}^{n} d_j x_{ij} \leqslant \sum_{\tau=1}^{t} b_{i\tau}.$$

The previous inequality holds for each $t = 1,\ldots,T$, and therefore this 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

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

and

$$\sum_{\tau=1}^{t} y_{i\tau} \geqslant \sum_{\tau=1}^{t} \sigma_\tau \sum_{j=1}^{n} d_j x_{ij}, \qquad i = 1, \ldots, m; t = 1, \ldots, T.$$

(Note that y can be interpreted as a set of feasible production levels corresponding to (x, I) in a 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 it has a finite value, and thus by strong LP duality we obtain

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

where

$$\begin{aligned} W_i = \big\{ & w \in \mathbb{R}^T \colon -w_t + w_{t+1} \geqslant h_{it}, \\ & t = 1, \ldots, T-1; w_t \geqslant 0, t = 1, \ldots, T\big\}. \end{aligned}$$

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

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

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

ACKNOWLEDGMENTS

The work of the second author was supported in part by the National Science Foundation under grant no. DMI-0085682. The work of the third author was supported in part by the Netherlands Organization for Scientific Research (NWO) and the National Science Foundation under grant no. DMI-0085682.

REFERENCES

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46**(3) 316–329.

Benders, J. F., W. K. M. Keulemans, J. A. E. E. van Nunen, G. Stolk. 1986. A decision support program for planning locations and allocations with the aid of linear programming. C. B. Tilanus, O. B. de Gaus, J. K. Lenstra, eds. *Quantitative Methods in Management: Case Studies of Failures and Successes*, Ch. 4. John Wiley and Sons, Chichester, U.K., 29–34.

Cattrysse, D. G., L. N. Van Wassenhove. 1992. A survey of algorithms for the generalized assignment problem. *Eur. J. Oper. Res.* **60** 260–272.

——, M. Salomon, L. N. Van Wassenhove. 1994. A set partitioning heuristic for the generalized assignment problem. *Eur. J. Oper. Res.* **72** 167–174.

Chan, L. M. A., A. Muriel, D. Simchi-Levi. 1998. Production/distribution planning problems with piece-wise linear and concave transportation costs. Research report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

CPLEX Reference Manual. 1999. *ILOG CPLEX 6.6*. ILOG, Inc., Incline Village, NV.

Duran, F. 1987. A large mixed integer production and distribution program. *Eur. J. Oper. Res.* **28** 207–217.

Fleischmann, B. 1993. Designing distribution systems with transport economies of scale. *Eur. J. Oper. Res.* **70** 31–42.

Geoffrion, A. M., G. W. Graves. 1974. Multicommodity distribution system design by Benders decomposition. *Management Sci.* **20**(5) 822–844.

Gilmore, P. C., R. E. Gomory. 1961. A linear programming approach to the cutting-stock problem. *Oper. Res.* **9** 849–859.

Hall, N. G., M. E. Posner. 2001. Generating experimental data for scheduling problems. *Oper. Res.* **49**(6) 854–865.

Hiriart-Urruty, J. B., C. Lemaréchal. 1993. *Convex Analysis and Minimization Algorithms: Fundamentals*, Vol. 1. Springer-Verlag, Berlin, Germany.

Martello, S., P. Toth. 1981. An algorithm for the generalized assignment problem. J. P. Brans, ed. *Operational Research '81*. IFORS, North-Holland, Amsterdam, The Netherlands, 589–603.

——, ——. 1990. *Knapsack Problems, Algorithms and Computer Implementations*. John Wiley and Sons, New York.

Neebe, A. W., M. R. Rao. 1983. An algorithm for the fixed-charge assigning users to sources problem. *J. Oper. Res. Soc.* **34**(11) 1107–1113.

Osman, I. H. 1995. Heuristics for the generalised assignment problem: Simulated annealing and tabu search approaches. *OR Spektrum* **17** 211–225.

Rinnooy Kan, A. H. G., L. Stougie, C. Vercellis. 1993. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Appl. Math.* **42** 279–290.

Romeijn, H. E., N. Piersma. 2000. A probabilistic feasibility and value analysis of the generalized assignment problem. *J. Combin. Optim.* **4**(3) 325–355.

——, D. Romero Morales. 1999. Asymptotic analysis of a greedy heuristic for the multi-period single-sourcing problem: The acyclic case. Research report 99-13, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL.

——, ——. 2001. Generating experimental data for the generalized assignment problem. *Oper. Res.* **49**(6) 866–878.

——, ——. 2003. An asymptotically optimal greedy heuristic for the multi-period single-sourcing problem: The cyclic case. *Naval Res. Logist.* **50**(5) 412–437.

Romero Morales, D. 2000. Optimization problems in supply chain management. Ph.D. thesis, TRAIL Thesis Series no. 2000/4 & ERIM Ph.D. Series Research in Management, No. 3. Rotterdam, The Netherlands.

Savelsbergh, M. W. P. 1997. A branch-and-price algorithm for the generalized assignment problem. *Oper. Res.* **45**(6) 831–841.

Srinivasan, V., G. L. Thompson. 1972. An algorithm for assigning uses to sources in a special class of transportation problems. *Oper. Res.* **21** 284–295.

Vanderbeck, F. 2000. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.* **48**(1) 111–128.