ELSEVIER

European Journal of Operational Research 115 (1999) 285–299

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Lagrangian-relaxation-based solution procedures for a multiproduct capacitated facility location problem with choice of facility type

Joseph B. Mazzola [a,*], Alan W. Neebe [b]

[a] *Fuqua School of Business, Duke University, Durham, NC 27708-0120, USA*
[b] *Kenan-Flagler Business School, University of North Carolina, CB #3490, Chapel Hill, NC 27599, USA*

Received 10 October 1996; accepted 1 May 1998

Abstract

This paper presents exact and heuristic solution procedures for a multiproduct capacitated facility location (MPCFL) problem in which the demand for a number of different product families must be supplied from a set of facility sites, and each site offers a choice of facility types exhibiting different capacities. MPCFL generalizes both the uncapacitated (or simple) facility location (UFL) problem and the pure-integer capacitated facility location problem. We define a branch-and-bound algorithm for MPCFL that utilizes bounds formed by a Lagrangian relaxation of MPCFL which decomposes the problem into UFL subproblems and easily solvable 0-1 knapsack subproblems. The UFL subproblems are solved by the dual-based procedure of Erlenkotter. We also present a subgradient optimization-Lagrangian relaxation-based heuristic for MPCFL. Computational experience with the algorithm and heuristic are reported. The MPCFL heuristic is seen to be extremely effective, generating solutions to the test problems that are on average within 2% of optimality, and the branch-and-bound algorithm is successful in solving all of the test problems to optimality.

Keywords: Capacitated facility location; Integer programming; Branch-and-bound algorithm; Heuristic; Lagrangian relaxation

1. Introduction

We consider a multiproduct capacitated facility location problem (MPCFL) in which the demand for a number of different product families is to be supplied from a specified set of facility sites, each offering a set of possible facility types, with each facility type exhibiting a finite capacity. This problem, referred to as MPCFL, was recently presented in Lee (1991, 1993). MPCFL is closely related to the well-known capacitated facility location (CFL) problem (see, for example, Geoffrion and Graves (1974), Akinc and Khumawala (1977), Geoffrion and McBride (1978) or Guignard and

* Corresponding author. Tel.: +1 919 660 7856; fax: +1 919 681 6245; e-mail: jmazzola@mail.duke.edu

0377-2217/99/$ – see front matter
PII: S0377-2217(98)00303-8

Opaswongkarn (1990), a review of the CFL problem is provided in Sridharan (1995)); MPCFL generalizes the pure-integer CFL in which each customer's demand must be met from a single facility (see, for example, Neebe and Rao (1983), Barcelo and Casanovas (1984), Klincewicz and Luss (1986), Barcelo et al. (1990)).

To provide a more precise description of MPCFL, let F be the (index) set of product families, with product (family) $f \in F$, and let $n_F = |F|$. The set of demand points (i.e., customers) is specified by I, and the demand d_{if} of customer $i \in I$ for product $f \in F$ is given. Let $n_I = |I|$. The total demand for product $f \in F$ is given by $D_f = \sum_{i \in I} d_{if}$. We assume $D_f > 0$, for each $f \in F$. The set of possible facility sites is represented by J; let $n_J = |J|$. There is a set K of different facility types that can be opened at each facility site $j \in J$. Each facility type $k \in K$ has a finite capacity $S_k > 0$, where capacity is measured in a unit that is common across all product families (e.g., labor hours or production hours). Let $n_K = |K|$.

There are costs associated with producing the product families at the facilities as well as with the transporting of product from the facilities to the demand points. Let C_{ijf} be the cost of shipping all of customer i's demand for product f from a facility located at site j. These costs are variable in that the cost incurred is proportional to the amount of customer i's demand that is shipped. The corresponding variable production cost at facility site j can also be included in this amount. MPCFL also takes into account fixed production costs. In particular, let $E_{jf} \geqslant 0$ be the fixed cost incurred to equip (or prepare) a facility located at site $j \in J$ to produce product f; this cost is incurred if any amount of product f is supplied from a facility located at site j. Also, let $G_{jk} \geqslant 0$ be the fixed cost associated with establishing a facility of type $k \in K$ at site j. We assume that for each $j \in J$ the set K of facility types is ordered such that $S_{k_1} > S_{k_2}$ and $G_{jk_1} > G_{jk_2}$, for $k_1 > k_2$ (since any facility providing less capacity but having a greater fixed cost is dominated and can thus be removed from consideration).

MPCFL requires the use of three sets of decision variables. For $i \in I, j \in J, f \in F$, and $k \in K$, let x_{ijf} be the fraction of customer i's demand for product f that is shipped from a facility located at site j; also, let

$$y_{jf} = \begin{cases} 1 & \text{if the facility at site } j \text{ is equipped} \\ & \text{to supply product } f, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$z_{jk} = \begin{cases} 1 & \text{if a facility of type } k \text{ is opened at site } j, \\ 0 & \text{otherwise.} \end{cases}$$

MPCFL can then be formulated as the following 0-1 mixed integer linear programming problem:

MPCFL :

$$\min \quad \sum_{f \in F} \left(\sum_{i \in I} \sum_{j \in J} C_{ijf} x_{ijf} + \sum_{j \in J} E_{jf} y_{jf} \right) + \sum_{j \in J} \sum_{k \in K} G_{jk} z_{jk}, \tag{1}$$

s.t.

$$\sum_{j \in J} x_{ijf} = 1, \quad i \in I, f \in F, \tag{2}$$

$$x_{ijf} \leqslant y_{jf}, \quad i \in I, j \in J, f \in F, \tag{3}$$

$$\sum_{f \in F} D_f y_{jf} \leqslant \sum_{k \in K} S_k z_{jk}, \quad j \in J, \tag{4}$$

$$\sum_{k \in K} z_{jk} \leqslant 1, \quad j \in J, \tag{5}$$

System configuration constraints on the y variables (6)

$$x_{ijf} \geqslant 0, \quad i \in I, j \in J, f \in F, \tag{7}$$

$$y_{jf} = 0 \text{ or } 1, \quad j \in J, f \in F, \tag{8}$$

$$z_{jk} = 0 \text{ or } 1, \quad j \in J, k \in K. \tag{9}$$

All data are assumed to be integers.

The objective (1) is to minimize the sum of variable (production and shipping) costs and fixed costs. Constraints (2) ensure that all of customer i's demand for each product is met. Constraints (3) require that each facility site j first be equipped to produce product f in order for any of product f to be shipped from it. Constraints (4) establish the finite capacity of the facility types at each site. Specifically, it is assumed that once a facility at location j is equipped to supply product f, then all of the demand, D_f, for that product can be supplied from the site. Constraints (4) then ensure that

the total demand required to produce the corresponding families at each site j does not exceed the total capacity of the facility types opened at that site. Constraints (5) provide for the selection of at most one facility type from among the set K of facility types at each potential site.

Constraints (6) allow for the inclusion of additional system configuration constraints involving the y variables. For example, these constraints might impose a policy restriction on the maximum number of sites, $n_{\max}$, that can supply each product family (Lee, 1991, 1993). This set of constraints would then assume the form

$$\sum_{j\in J} y_{jf} \leqslant n_{\max}, \quad f \in F. \tag{10}$$

Finally, constraints (7)–(9) provide for the nonnegativity and integrality of the decision variables. We can assume without loss of generality that $x_{ijf} = 0$ or 1 in an optimal solution to MPCFL, since constraints (4) allow for all of the demand for a product to be supplied from each site that is equipped to supply the product. Thus, each customer's demand for a product will be supplied from a single site. MPCFL therefore generalizes the pure-integer (or single sourcing) CFL. Note, however, that the model allows for the same product to be supplied to different customers from different sites; that is, it may be optimal to equip more than one site to supply any product.

Observe that if $S_k = \sum_{f\in F} D_f$ and if $n_K = 1$ with $G_{j1} = 0$, for all $j \in J$, then MPCFL decomposes into a set of independent uncapacitated (or simple) facility location problems. (For a discussion of the uncapacitated facility location problem (UFL), see, for example, Efroymson and Ray (1966); Khumawala (1972); Cornuéjols et al. (1977); Erlenkotter (1978); Krarup and Prozan (1983); or Klincewicz and Luss (1987).) Since UFL is known to be NP-hard (see, for instance, Nemhauser and Wolsey, 1988), it follows that MPCFL is NP-hard.

Lee (1991) presents an algorithm for MPCFL that is based on Benders' decomposition. Lee (1993) later defines a cross-decomposition algorithm for MPCFL, which combines Benders' decomposition and a Lagrangian decomposition of MPCFL in which constraints (3) are dualized. We explore the decomposition of MPCFL based on the relaxation of constraints (4). This decomposition enables the problem to separate into n_F independent UFL subproblems and n_J easily solvable 0-1 knapsack problems (with a special structure). The extremely effective dual-based procedure (DUALOC) for UFL defined in Erlenkotter (1978) provides the impetus for exploring this approach to MPCFL problem solution. Barcelo et al. (1990) apply this bounding approach to the pure-integer CFL, utilizing it in conjunction with cutting planes derived from valid inequalities of polytopes corresponding to knapsack inequalities implied by the capacity constraints.

In Section 2 we define a Lagrangian relaxation of MPCFL in which constraints (4) are dualized. A subgradient-optimization procedure that utilizes Erlenkotter's DUALOC algorithm to solve the Lagrangian relaxations is then presented. The resulting bounding procedure provides a lower bound on the optimal objective value of MPCFL. In Section 3 we define a subgradient optimization-Lagrangian relaxation-based heuristic for MPCFL. The heuristic employs a solution-improving subprocedure which strives systematically to improve the solution by shifting production of products among facilities. In Section 4 we define a polychotomous, breadth-first tree-search strategy for solving MPCFL. We also present a number of logical tests which are designed to facilitate the search. In conjunction with the lower bounding procedure of Section 2, the branching strategy and logical tests form a branch-and-bound algorithm for MPCFL. Computational results with the MPCFL branch-and-bound algorithm and heuristic are reported in Section 5, and conclusions are offered in Section 6.

2. Lagrangian relaxation-based lower bounds

We begin this section by discussing a Lagrangian relaxation of MPCFL which can be used to provide lower bounds on the optimal objective value to MPCFL. (See Geoffrion (1974) or Fisher (1981, 1985) for a discussion of Lagrangian relaxation.) Let z_{MPCFL} be the optimal objective value to MPCFL. For any $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{n_J}) \geqslant 0$, a lower bound on z_{MPCFL} can be obtained by

dualizing constraints (4), relaxing the system configuration constraints (6), and solving the following Lagrangian relaxation (LR(λ)):

$$\begin{aligned}\mathrm{LR}(\lambda): \quad & \theta(\lambda) \\ = \min \quad & \sum_{f\in F}\left(\sum_{i\in I}\sum_{j\in J} C_{ijf}x_{ijf} + \sum_{j\in J}(E_{jf}+\lambda_j D_f)y_{jf}\right) \\ & + \sum_{j\in J}\sum_{k\in K}(G_{jk}-\lambda_j S_k)z_{jk}\end{aligned}$$

s.t. (2), (3), (5), (7)–(9).

The best such lower bound is then obtained by solving

$$\theta = \max_{\lambda \geqslant 0}\theta(\lambda).$$

For fixed λ the optimal values of the x and y variables in the Lagrangian problem LR(λ) can be established independently of the optimal values of the z variables. Moreover, LR(λ) can be solved by independently solving two sets of problems; namely, a set of *facility-location* problems involving the x and y variables, and a set of *facility-type* problems involving the z variables. This can be done as follows.

Observe first that the Lagrangian problem decomposes on the index f into a set of n_F uncapacitated facility location problems:

$$\theta_1(\lambda, f) = \min \quad \sum_{i\in I}\sum_{j\in J} c_{ijf}x_{ijf} + \sum_{j\in J}(E_{jf}+\lambda_j D_f)y_{jf} \tag{11}$$

s.t.

$$\sum_{j\in J} x_{ijf} = 1, \quad i \in I, \tag{12}$$

$$x_{ijf} \leqslant y_{jf}, \quad i \in I, j \in J, \tag{13}$$

$$x_{ijf} \geqslant 0, \quad i \in I, j \in J, \tag{14}$$

$$y_{jf} = 0 \text{ or } 1, \quad j \in J. \tag{15}$$

The UFL problem (11)–(15) has been discussed extensively in the literature and can be solved using any one of a number of algorithms. Note that Eqs. (11)–(15) constitute the strong formulation of the UFL, as opposed to the weak formulation in which constraints (13) are replaced by

$$\sum_{j\in J} x_{ijf} \leqslant n_J y_{jf}, \quad i \in I. \tag{16}$$

The dual-based algorithm presented in Erlenkotter (1978), which is based on the strong formulation, has been demonstrated to be extremely effective in solving UFL and is thus a key component of the bounding procedure for MPCFL presented here.

Next, observe that the Lagrangian problem decomposes on the index j into a series of n_J facility type problems:

$$\theta_2(\lambda, j) = \min \quad \sum_{k\in K}(G_{jk}-\lambda_j S_k)z_{jk} \tag{17}$$

s.t.

$$\sum_{k\in K} z_{jk} \leqslant 1, \tag{18}$$

$$z_{jk} = 0 \text{ or } 1, \quad k \in K. \tag{19}$$

The 0-1 knapsack problem defined by Eqs. (17)–(19) can be solved easily by setting

$$\theta_2(\lambda, j) = \min\left\{\min_{k\in K}\{G_{jk}-\lambda_j S_k\}, 0\right\}.$$

Clearly $\theta(\lambda) = \sum_{f\in F}\theta_1(\lambda, f) + \sum_{j\in J}\theta_2(\lambda, j)$.

For fixed λ, let $(x^*_{ijf}, y^*_{jf}, z^*_{jk})$ be an optimal solution to LR(λ). Since we propose solving MPCFL using branch and bound, the objective is to find a λ vector that obtains a large value of the lower bound $\theta(\lambda)$; i.e., that approximates the solution to θ. To accomplish this we use subgradient optimization (see, for example, Geoffrion, 1974; Held et al., 1974; or Goffin, 1977). The vector $h = (h_1, h_2, \ldots, h_{n_J})$, where

$$h_j = \sum_{f\in F} D_f y^*_{jf} - \sum_{k\in K} S_k z^*_{jk}, \quad j \in J, \tag{20}$$

defines a subgradient of $\theta(\lambda)$ at $(x^*_{ijf}, y^*_{jf}, z^*_{jk})$. The subgradient h provides a direction for adjusting λ. The sequence of values λ^t can be generated using the following rule:

$$\lambda^{t+1} = \max\{\lambda^t + \rho_t h^t, 0\},$$

where ρ_t is a sequence of scalars and h^t is calculated using Eq. (20) and the solution to the Lagrangian problem in iteration t. Under certain conditions on the ρ_t (see for example, Held et al., or Goffin), $\theta(\lambda^t)$ converges to θ. Because the

convergence is not monotonic, it is necessary to keep track of the best solution and bound identified by the subgradient procedure. In practice, limits are typically placed on the total number of subgradient iterations, as well as on the number of consecutive iterations of the subgradient procedure that are permitted with no improvement in the lower bound.

Since the problem data are assumed to be integers, the DUALOC procedure has been coded to take advantage of the all-integer calculations, which facilitates the execution time of the procedure. Because the DUALOC procedure is called for repeatedly in the calculation of the lower bound, it is therefore desirable to begin with integer multipliers and then maintain integrality of the multipliers λ by utilizing an integer-valued step size.

With regard to selecting an initial value for λ^0, rather than initializing $\lambda^0 = 0$, we found that a better choice is given by initially setting

$$\lambda_j^0 = \min_{k \in K} \left\{ \left\lfloor \frac{G_{jk}}{S_k} \right\rfloor \right\}, \quad j \in J,$$

where $\lfloor x \rfloor$ is the greatest integer less than or equal to x. Observe that this rule identifies the most efficient facility type for facility site j with respect to cost per unit of capacity. Using this rule, $(G_{jk} - \lambda_j S_k) \geqslant 0$, for all $k \in K$, so initially $\theta_2(\lambda, j) = 0$ for $j \in J$.

In the calculation of the step size, the subgradient optimization procedure for MPCFL uses the step size defined by setting

$$\rho_t = \begin{cases} \hat{\rho}/|h^t| & \text{if } h^t \neq 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\hat{\rho}$ is an integer constant. Through experimentation we found that this step size is effective in producing good lower bounds and that convergence occurs quickly.

To provide an indication of the relative strength of the lower bound that can be derived using this lower-bounding procedure, we consider briefly the two-product, three-site, two-facility-type MPCFL example defined in Lee (1991) (see Lee, 1991, pp. 175–177) for the specification of the problem data). The optimal solution to this example problem has a value of 15,590. The linear programming relaxation of the MPCFL formulation (1)–(9) corresponding to this problem has an optimal solution value of 13,990. Applying the subgradient method to the Lagrangian relaxation defined above yields this same lower bound. While this lower bound could theoretically improve upon the linear programming relaxation bound (since UFL does not satisfy the Integrality Property (see Geoffrion, 1974)), it is not surprising that the same bound is attained (recall that the strong formulation of the UFL often does not exhibit a duality gap). If, however, constraints (13) are replaced by constraints (16) (i.e., if the weak formulation is used), the optimal value of the linear programming relaxation is 12,523.3. This lower-bounding procedure thus closes 48% of the gap between the optimal objective value and the objective value of the LP relaxation using the weak formulation, providing a substantial improvement in the quality of the lower bound.

3. A Lagrangian relaxation-based heuristic for MPCFL

Another advantage of solving the Lagrangian problem is that the solutions (x_{ijf}^*, y_{jf}^*) to the UFL problems (11)–(15) typically can be *extended* with little additional effort to produce a feasible solution to MPCFL. This can be done, for $j \in J$, by simply identifying the facility type k_j^* having least cost $G_{jk_j^*}$ such that

$$S_{k_j^*} \geqslant \sum_{f \in F} D_f y_{jf}^*.$$

If such a facility type can be identified for all $j \in J$, the corresponding solution is feasible to MPCFL. This subprocedure is applied during each iteration of the subgradient method, and the best heuristic solution found is retained. This procedure thus constitutes a subgradient optimization-Lagrangian relaxation-based heuristic for MPCFL. The use of a Lagrangian-based heuristic embedded in a subgradient-optimization search procedure has been found to be useful on a variety of combinatorial optimization problems (see, for example, Gavish and Pirkul (1986), Jörnsten and Näsberg (1986),

Mazzola and Neebe (1986) or Tempelmeier and Derstroff (1996); also, Barcelo and Casanovas (1984) and Klincewicz and Luss (1986) define Lagrangian-based heuristics for the pure-integer CFL).

Prior to the initial application of the subgradient-optimization procedure, an initial feasible solution $(x'_{ijf}, y'_{jf}, z'_{jk})$ to MPCFL is constructed using a simple two-stage *initial-solution* heuristic subprocedure. Let $(x'_{ijf}, y'_{jf}, z'_{jk})$ be the partially constructed heuristic solution vector generated during each step of this subprocedure; all components of this vector are initially set to 0. In the first stage of the subprocedure each of the products $f \in F$ is considered in turn. When considering product $\hat{f}$, let $\hat{T}_j$ be the total demand assigned thus far by the subprocedure to site j (i.e., $\hat{T}_j = \sum_{f=1}^{\hat{f}-1} D_f y'_{jf}$, and let $\hat{m} = \min_{j \in J} \hat{T}_j$. Then, for all sites j satisfying $\hat{T}_j = \hat{m}$, the *single* site $\hat{j}$ is identified that minimizes

$$E_{j\hat{f}} + \sum_{i \in I} C_{ij\hat{f}} x'_{ij\hat{f}}$$

(ties are broken arbitrarily), and set $y'_{\hat{j}\hat{f}} = x'_{i\hat{j}\hat{f}} = 1$, for all $i \in I$.

Observe that this initial-solution heuristic subprocedure assigns all of the production of family $\hat{f}$ to a single site $\hat{j}$. Also, at least one such production site for product $\hat{f}$ is necessary. Hence, the solution generated by this subprocedure will satisfy the additional system configuration constraints (10).

The second stage of the initial-solution heuristic subprocedure then selects the facility types. Specifically, for each $j \in J$, identify the facility type k^* with least cost G_{jk^*} such that

$$S_{k^*} \geqslant \sum_{f \in F} D_f y'_{jf}$$

and set $z'_{jk^*} = 1$. By limiting the selection in stage one to sites exhibiting the least assigned demand, the heuristic subprocedure thus attempts to foster the identification of facility types in stage two with sufficient capacity to meet the required demand.

A *solution-improving* heuristic subprocedure is also applied to the initial feasible solution (obtained by the subprocedure described above), as well as to each (extended) feasible solution generated during each iteration of the subgradient method. To define this subprocedure, let $(\hat{x}_{ijf}, \hat{y}_{jf}, \hat{z}_{jk})$ be a feasible solution to MPCFL. Also, let $\hat{J} = \{j \in J | \hat{z}_{jk} = 1 \text{ for some } k \in K\}$; i.e., $\hat{J}$ is the set of all sites at which a facility has been opened in this solution. For each $j \in \hat{J}$, $\hat{F}_j = \{f \in F | \hat{y}_{jf} = 1\}$, which is the set of products produced at facility j. Finally, for each $f \in F$, let $\hat{J}_f$ be the set of sites in $\hat{J}$ at which product f is produced.

We assume that the solution satisfies the properties that (i) for each $j \in \hat{J}, \hat{F}_j \neq \emptyset$, (ii) for each $f \in \hat{F}_j, j \in \hat{J}$, there exists at least one $i \in I$ for which $\hat{x}_{ijf} > 0$, and (iii) for each $f \in \hat{F}_j, j \in \hat{J}$, if $\hat{x}_{ijf} > 0$ for some $i \in I$, then $C_{ijf} = \min_{h \in \hat{J}_f} \{C_{ihf}\}$, and $\hat{x}_{ijf} = 1$. If property (i) is not satisfied, the solution can be immediately improved by closing the site, in which case G_{jk} is saved. If property (ii) is not satisfied, site j can be eliminated as a production location for product f, thus saving E_{jf}. Similarly, if the first part of property (iii) is not satisfied, an immediate cost saving can be realized by shifting production of product f to the facility with cheaper shipping cost. The second part of property (iii) follows from the observation that if $\hat{x}_{ijf} > 0$, then $\hat{y}_{jf} = 1$; hence, we may assume that $\hat{x}_{ijf} = 1$ (and also that $\hat{x}_{ihf} = 0$, for all $h \in J \setminus \{j\}$). It is straightforward to examine a solution for these properties and to modify it so that they are satisfied.

The solution-improving subprocedure considers three types of moves which involve shifting the supply (or production) of products from sites in $\hat{J}$. The use of similar moves has proven to be effective within the context of a tabu-search heuristic for a multifacility production loading problem under capacity-based economies of scope (Mazzola and Schantz, 1997). For each $j \in \hat{J}$, each product $f \in \hat{F}_j$ that is assigned for production at facility j is considered (in arbitrary order). The first move considers closing down production of product f at the facility and shifting production to facilities in $\hat{J}$ that also produce product f. Specifically, if $\hat{J}_f = \{j\}$, then facility j must continue to produce f, and so this first move type is not considered further. Otherwise, let $J'_f = \hat{J}_f \setminus \{j\}$; let k_j be the unique $k \in K$ for which $z_{jk_j} = 1$, and let k' be the smallest $k \in K$ for which $\sum_{\ell \in \hat{F}_j \setminus \{f\}} D_\ell \leqslant S_{k'}$ (recall that the set K is ordered according to increasing values of S_k). Compute

$$E_{jf} + \sum_{i \in I} \left(C_{ijf} - \min_{h \in J'_f} \{C_{ihf}\} \right) + (G_{jk_j} - G_{jk'}).$$

Observe that this is the total cost saving arising from transferring production of product f to the least expensive facility in $\hat{J}$ other than j that can supply product f, thereby saving the requisite fixed production cost and possibly allowing a smaller size facility at site j (and thus possibly resulting in an additional saving in fixed cost).

For each $j \in \hat{J}$, and each product $f \in \hat{F}_j$, the second type of move considers shifting some of the production of product f to other facilities at sites in $\hat{J}$ that are not already producing product f. In particular, consider each site $h \in \hat{J} \setminus \{j\}$ and for which $f \notin \hat{F}_h$. Let $R_h = S_{k_h} - \sum_{\ell \in \hat{F}_h} D_\ell$, which is the remaining (unused) capacity of the facility at site h. If $D_f > R_h$, then site h is not considered further for this type of move. Otherwise, let $\hat{I}_{jf} = \{i \in I | x_{ijf} = 1\}$, which is the set of customers who are supplied product f from site j. Compute

$$\sum_{i \in \hat{I}_{jf}} \max\{C_{ijf} - C_{ihf}, 0\} - E_{hf}.$$

This is the cost saving from expanding production of the facility at site h to include product f and shifting to it those customers who can be provided this product at a lower shipping cost than formerly available from the facility at site j.

The third type of move is much like the second, except that the set of products that can be produced at the facility at site h is expanded to include product f and *all* of the production of product f at site j is shifted to sites in $J'_f \cup \{h\}$. As in the previous type of move, consider each site $h \in \hat{J} \setminus \{j\}$ and for which $f \notin \hat{F}_h$. Once again, this type of move is only considered if $D_f \leqslant R_h$. The calculation of the cost saving for this move is

$$(E_{jf} - E_{hf}) + \sum_{i \in \hat{I}_{jf}} \left(C_{ijf} - \min_{\ell \in J'_f \cup \{h\}} \{C_{i\ell f}\} \right) + (G_{jk_j} - G_{jk'}),$$

where k_j and k' are defined as in the first type of move.

The solution-improving heuristic subprocedure identifies the best of these three types of moves over all $j \in \hat{J}$ and $f \in \hat{F}_j$ (with ties broken arbitrarily by the applicable rule type with the smallest index, as enumerated above). If this best rule type yields a positive cost saving, the move is performed and the solution is modified accordingly. It is then checked to ensure that properties (i), (ii), and (iii) noted above are satisfied (if not, the solution is modified appropriately). The solution-improving process is performed iteratively until no further improvement in the solution quality is realized.

Note that this subprocedure is also easily modified to accommodate the system configuration constraints (10). In particular, the number of sites supplying a product f can only increase if the second type of move is performed. If such a move would result in the number of such sites exceeding $n_{\max}$, then this move is excluded from further consideration.

The initial-solution heuristic subprocedure, the subgradient optimization-Lagrangian relaxation based heuristic search, and the application of the solution-improving heuristic subprocedure to the initial solution and to each feasible solution generated during the application of the subgradient method combine to form the MPCFL heuristic. The best solution identified in the course of this search is then returned as the heuristic solution. Computational experience with this MPCFL heuristic is presented in Section 5.

4. A branch-and-bound algorithm for MPCFL

This section discusses a polychotomous branching rule which is used to define a breadth-first tree-search procedure for solving MPCFL. We also present logical tests that are used to enhance the tree search. When used in conjunction with the Lagrangian-relaxation bounding procedure discussed in the previous section, this enhanced tree-search procedure forms a branch-and-bound algorithm for MPCFL.

The search tree for solving MPCFL branches on both the z_{jk} and the y_{jf} variables. At each level of the tree, either a z_{jk} variable or a y_{jf} variable is selected for branching. In the top part of the tree the selection of the branching variable is limited to the z_{jk} variables. As the search progresses down the

search tree, after all of the z_{jk} have been specified along a particular branch, the search procedure enters the lower part of the search tree and begins to branch on the y_{jf} variables. As the z_{jk} variables are specified in the tree search, logical tests can be applied to fix the values of some of the y_{jf} variables. Thus, many of the y_{jf} variables will have been fixed prior to entering the lower part of the search tree. We now discuss the details of this tree search within the context of the branch-and-bound algorithm.

At each node $\hat{v}$ of the search tree, a corresponding partial solution $(\hat{z}, \hat{y})$ is used to define the subproblem at the node. In this subproblem, specific values of $\hat{z}_{jk}$ and $\hat{y}_{jf}$ will have been set equal to 0 or 1 as specified by the branches along the path from the top node (node 0) to node $\hat{v}$. In addition, at this node some of the values of $\hat{y}_{jf}$ can also be fixed as a result of logical conditions implied by the partial specification of the $\hat{z}_{jk}$ variables. Specifically, if $\hat{z}_{jk} = 0$ for all $k \in K$, then $\hat{y}_{jf} = 0$ for all $f \in F$. Also, if $\hat{z}_{jk} = 1$, then from constraint (4) any completion of the $\hat{y}_{jf}$ variables for which $\sum_{f \in F} D_f \hat{y}_{jf} > S_k$ is infeasible (i.e., can be fathomed). This latter condition allows for some of the $\hat{y}_{jf}$ to be set to zero. For example, if there exists a product f for which the variable y_{jf} is free (i.e., not fixed to either a value of 0 or 1) and for which D_f exceeds the remaining capacity at some site j, then y_{jf} can be fixed to zero.

The branch-and-bound search begins at the top node. At this node, all variables are free, and no variables are fixed in the corresponding partial solution. The subgradient-optimization procedure is applied to the Lagrangian relaxation of MPCFL yielding a lower bound $\underline{Z}$ on the optimal objective value of MPCFL. Let $(x^*_{ijf}, y^*_{jf}, z^*_{jk})$ be the solution identified by the lower-bounding procedure corresponding to this lower bound, and let λ^* be the corresponding multiplier. Simultaneously, the heuristic is applied, generating a solution with value Z^{H} (if the heuristic fails to identify a feasible solution, then $Z^{\mathrm{H}} = \infty$). The upper bound $\overline{Z}$ on the optimal objective value of MPCFL is set equal to Z^{H}, and if $\overline{Z} < \infty$, the incumbent solution to MPCFL (i.e., the best solution of MPCFL found thus far in the search) is set equal to the solution identified by the heuristic. If $\underline{Z} = Z^{\mathrm{H}}$, then the heuristic solution is optimal, and the procedure terminates. Otherwise, the branching procedure is initiated. This begins by initializing the set N of active nodes by setting $N = \emptyset$. Designating this node as the current node, the branch-and-bound search proceeds to make a forward (branching) move.

A forward move (away from the current node) begins by identifying the site j' for which

$$\sum_{f \in F} D_f y^*_{j'f} - \sum_{k \in K} S_k z^*_{j'k} = \max_{j \in J} \left\{ \sum_{f \in F} D_f y^*_{jf} - \sum_{k \in K} S_k z^*_{jk} \right\} \tag{21}$$

(ties are broken arbitrarily) in the solution returned by the bounding procedure. Observe that if the left hand side of (21) is positive, then constraint (4) is violated; thus this rule seeks to identify the site for which constraint (4) is the most violated. This rule is still used if constraint (4) is not violated. However, in that case the solution $(x^*_{ijf}, y^*_{jf}, z^*_{jk})$ is feasible to MPCFL; if this occurs, its objective value is compared with that of $\overline{Z}$, and if there is an improvement, the incumbent solution and $\overline{Z}$ are updated. If the incumbent is updated, all active nodes for which the corresponding lower bound is greater than or equal to $\overline{Z}$ are removed from N. (This update of N is performed each time the incumbent solution is updated.) Also, when a feasible solution to MPCFL is identified, the solution $(x^*_{ijf}, y^*_{jf}, z^*_{jk})$ and the multiplier λ^* are examined to identify whether the complementary slackness condition is met. If so, the current solution is optimal to the MPCFL subproblem corresponding to the current node. The node is then fathomed immediately, and the search performs a backtracking step (as described below).

If the branch occurs in the top part of the search tree (where the branching is limited to the z_{jk} variables), $n_K + 1$ potential descendant nodes of the current (parent) node are considered for possible inclusion in the set N of active nodes. Specifically, $n_K + 1$ branches corresponding to $\sum_{k \in K} z_{j'k} = 0$, and $z_{j'k} = 1$ for all $k = 1, \ldots, n_K$ are considered. For each of these potential descendant nodes, the two logical tests discussed above are first applied to fix, if possible, some of the y_{jf}

variables. Then, additional logical conditions (described below) can be applied in an attempt to fathom the node immediately.

If the node is not fathomed, then the subgradient-optimization bounding procedure is applied to the corresponding subproblem. If the resulting lower bound is greater than or equal to $\overline{Z}$, the node is fathomed and eliminated from further consideration. As before, if the solution from the bounding procedure is feasible to MPCFL, its value is compared with that of the incumbent solution's, and the incumbent solution, $\overline{Z}$, and N are modified accordingly. Also, if the solution is feasible to MPCFL and if the complementary slackness condition is met, the node can be fathomed. Otherwise, the node is included in the set N of active nodes. The heuristic procedure is applied during each application of the lower-bounding procedure. If a solution with a better objective value than $\overline{Z}$ is found, the incumbent and $\overline{Z}$ are updated, as is N.

The forward move is the same in the lower part of the tree except that instead of creating $n_K + 1$ branches, two branches are considered which correspond to $y_{j'f'} = 0$ or $y_{j'f'} = 1$, where f' is selected arbitrarily from among those $f \in F$ for which $y_{j'f}$ has not been fixed in the branching or by the logical conditions arising from the branching on the z_{jk} variables. If no such f' exists, then the search backtracks. We also employ an additional logical test. Specifically, since all the z_{jk} variables have been fixed in this part of the tree, if $\sum_{j\in J}\sum_{k\in K} S_k \hat{z}_{jk} < \sum_{f\in F} D_f$, then the total capacity is not sufficient to meet the total demand, and the node can be fathomed.

After a forward move has been completed, the search backtracks by selecting the node in N with the smallest corresponding lower bound (again, ties are broken arbitrarily). This node becomes the current node, and the search then proceeds to examine this node by removing it from the set of active nodes and reapplying the forward move procedure. If a forward move from the current node cannot be made, then the search procedure backtracks again by selecting from N the active node with the smallest lower bound, and the process is repeated. The branch-and-bound search is completed when a backtrack move is attempted and $N = \emptyset$.

The branch-and-bound algorithm can easily accommodate the system configuration constraints (10) by incorporating them into the branching routine. Specifically, in the lower part of the tree whenever a branch requires the setting of a y_{jf} variable to 1, the corresponding constraint (10) is checked, and if the branch will cause the constraint to be violated, the search backtracks immediately.

Two additional logical conditions are now described which can potentially facilitate the fathoming of nodes immediately prior to their possible placement in the set of active nodes. These conditions are based on the logical tests employed in Khumawala (1972) for the UFL problem.

Given a partial solution $(\hat{z}, \hat{y})$ corresponding to a node $\hat{v}$ of the search tree, for each $f \in F$, let $J_f^0 = \{j \in J \mid \hat{y}_{jf} = 0\}$, $J_f^1 = \{j \in J \mid \hat{y}_{jf} = 1\}$, and $J_f^2 = J \setminus (J_f^0 \cup J_f^1)$, which define the sets of sites serving product f that (by the branching) are fixed closed, open, or free to supply product f, respectively, in a completion of the partial solution $(\hat{z}, \hat{y})$. Also, for $j \in J$, let $F_j^0 = \{f \in F \mid y_{jf} = 0\}$, $F_j^1 = \{f \in F \mid y_{jf} = 1\}$, and $F_j^2 = F \setminus (F_j^0 \cup F_j^1)$, which define the set of products that cannot, can already, or are free to be supplied from site j, respectively, in a completion of the current partial solution.

For each product $f \in F$, the set $J_f^1 \cup J_f^2$ specifies those facilities that can potentially supply this product to any customer $i \in I$ at this node or any of its descendants. Also, for any site $j \in J_f^2$, let

$$S'_j = \begin{cases} S_{k_j} & \text{if } \hat{z}_{jk_j} = 1 \text{ for some } k_j \in K, \\ 0 & \text{otherwise,} \end{cases}$$

(recall from constraint (5) that if $k_j = 1$ for some $k_j \in K$, then k_j is unique).

The following logical condition pertains to the fixing of a variable $y_{jf} = 1$. For $j \in J_f^2$, assume that the condition

$$\sum_{f\in F_j^1} D_f + \sum_{f\in F_j^2} D_f \leqslant S'_j \tag{22}$$

is satisfied by the partial solution $(\hat{z}, \hat{y})$. Then, for $f \in F$ and $i \in I$, let

$$\nabla_{ij}^f = \min_{h\in\left(J_f^1\cup J_f^2\right)\setminus\{j\}} \{(C_{ihf} - C_{ijf})^+\},$$

(where $(x)^+ = \max\{x, 0\}$), and let

$$\Delta_j^f \equiv \left(\sum_{i\in I} \nabla_{ij}^f\right) - E_{jf}.$$

Proposition 1. *If condition (22) holds for* $j \in J_f^2$, *and* $\Delta_j^f > 0$ for $f \in F$, then $y_{jf} = 1$ *in any optimal solution that is a completion of the partial solution* $(\hat{z}, \hat{y})$.

Proof. Suppose that $y_{jf}^* = 0$ in an optimal solution (z^*, y^*) that is a completion of $(\hat{z}, \hat{y})$. Modify (z^*, y^*) by setting $y_{jf}^* = 1$. The increase in fixed cost arising from this change is equal to E_{jf}. Since condition (22) holds, this modified solution will satisfy constraint (4) for site j. Now, because $\Delta_j^f > 0$, the cost saving from supplying customers from the newly open site is strictly greater than the increase in fixed cost, thus contradicting the optimality of (z^*, y^*). □

Remark. The conditions of this proposition need to be modified to accommodate the presence of system configuration constraints (6). For example, for constraints (10) in which there exists an upper limit, $n_{\max}$, on the number of facility sites that can supply product f, if the conditions of the proposition are modified to also include the additional requirement on $j \in J_f^2$ that $|F_j^1 \cup F_j^2| \leqslant n_{\max}$, the result then follows.

The following logical condition can be used to fix sites closed. For $f \in F$, $j \in J_f^2$, and $i \in I$, define

$$\omega_{ij}^f = \min_{h\in J_f^1}\{(C_{ihf} - C_{ijf})^+\},$$

where $\omega_{ij}^f \equiv \infty$ if $J_f^1 = \emptyset$.

Proposition 2. *For* $j \in J_f^2$ *and* $f \in F$, *if* $\Omega_j^f \equiv (\sum_{i\in I} \omega_{ij}^f) - E_{jf} < 0$, *then* $y_{jf} = 0$ *in any optimal solution that is a completion of the partial solution* $(\hat{z}, \hat{y})$.

The proof of this proposition is immediate. Also, the conditions under which this result remains valid would need to be modified to accommodate the general system configuration constraints (6). Note, however, that no additional modification is needed for constraints (10), since this proposition is used to fix sites closed.

In the MPCFL branch-and-bound algorithm, each time the subgradient-optimization bounding procedure is applied after the first node, the multipliers are initialized with the best multipliers obtained at the parent node. Also, whenever the incumbent solution is updated to reflect an improvement, the solution-improving subprocedure is called in an attempt to further improve the incumbent solution.

5. Computational experience

A set of computational experiments were performed to test the effectiveness of the MPCFL solution procedures. The MPCFL branch-and-bound algorithm and heuristic were coded in FORTRAN and run on a 200 MHz pentium processor using a Microsoft compiler. To decrease computational effort, the entire code, including the DUALOC subroutine used to solve the UFL problems, was written for integer arithmetic.

In regard to the branch-and-bound algorithm, the subgradient-optimization bounding procedure uses the parameter $\hat{\rho} = 40$. Because the bounding procedure exhibits relatively rapid convergence, it is terminated (at each node) whenever there is no improvement in the lower bound after two consecutive iterations. Preliminary experiments also suggested that the Δ_j^f-based logical test was not effective computationally (the conditions of Proposition 1 were seldom satisfied); so this test was not included in the final version of the branch-and-bound algorithm. The MPCFL heuristic is applied at the top node of the search tree, and the resulting solution is used to initialize the branch-and-bound search.

To test the performance of the MPCFL heuristic and branch-and-bound algorithm, we also developed FORTRAN versions of the embedded Benders' decomposition (EBD) and the cross-decomposition (CD) algorithms defined in Lee (1991, 1993). The EBD algorithm is based on a 0-1 implicit enumeration procedure (Balas, 1965). In addition, Benders' decomposition is embedded in the search procedure so that the tree search is

performed only once (see Lee (1991) for details). Lee (1993) defines a CD algorithm for MPCFL which uses the CD approach (Van Roy, 1983, 1986) to develop lower and upper bounds on the optimal objective value of MPCFL. Since the extent to which the upper and lower bounds can converge under the CD procedure is limited by the size of the duality gap, the final step of the CD algorithm calls for the solution of the problem by a branch-and-bound algorithm or in the case of MPCFL, as suggested in Lee (1993), the EBD algorithm. On the basis of the computational experiments reported in Lee (1993), the CD and EBD algorithms offered (at the time of that study) the most effective solution approaches to MPCFL, with the CPU time of the CD algorithm being somewhat less than that of the EBD algorithm except for some small problems. We therefore included in our computational experiments a comparison of the MPCFL branch-and-bound algorithm defined here with the CD and EBD algorithms.

The CD algorithm begins by applying the CD (bounding) procedure, which iteratively generates upper and lower bounds. The CD procedure requires the repeated solution of two integer programming subproblems; one occurring in the primal decomposition (the Benders' master problem), and the other occurring in the dual decomposition (see Lee (1993) for details). The version of the CD algorithm that we developed solves these subproblems using a branch-and-bound algorithm which utilizes bounds derived from linear programming relaxations. The CD procedure is then applied until successive iterations of the bounding procedure no longer yield a sufficient improvement in the gap between the upper and lower bounds. The CD bounding phase is then terminated, and the algorithm enters the final step in which the remaining gap is closed using an exact algorithm. In our implementation, the EBD algorithm is applied at this point to solve the problem to optimality. We therefore refer to this approach as the CD-EBD algorithm, since it combines the CD bounding phase with the EBD final phase.

As noted in Lee (1993) the CD bounding phase can be accelerated by using a tolerance factor which artificially reduces the gaps between the upper and lower bounds encountered in the solution of the dual (decomposition) subproblem and the Benders' master subproblem (noted in the foregoing paragraph). The use of such a factor no longer guarantees the optimality of the solution generated at the end of the CD phase; however, since MPCFL typically exhibits a positive duality gap, the CD-EBD algorithm seldom generates an optimal solution at the end of the CD phase, and if no such tolerance factor is used in the final, EBD phase, the optimality of the final solution obtained by the CD-EBD algorithm will be preserved. The version of the CD-EBD algorithm that we developed therefore takes advantage of this and uses such a tolerance factor in the solution of these subproblems. This then necessitates the application of the EBD phase. The EBD phase of the algorithm does not employ a tolerance factor.

A total of 240 problems, falling into two problem classes, were generated. Following Lee (1993) for all problems the transportation costs C_{ijf} are randomly drawn from $U[1000, 5000]$, the uniform distribution on the interval $[1000, 5000]$, while the fixed facility preparation costs E_{jf} are randomly drawn from $U[5000, 10000]$. In the first Random Problem Class, the fixed facility setup costs G_{jk} are randomly drawn from $U[10000, 20000]$, the demands D_f from $U[10, 40]$, and the capacities S_k from $U[25, 125]$.

A second class of problems was also generated to test the procedures on problems exhibiting more structured costs. For the Structured Problem Class problems, the demands D_f are randomly drawn from $U[10, 90]$, and (since $n_K = 3$) the capacity S_1 of the first facility type is drawn from $U[30, 70]$, the second S_2 from $U[70, 110]$, and the third S_3 from $U[110, 150]$. Finally, for this problem class, the setup costs $G_{jk} = 5000 + 100S_k$, for all $j \in J$. Note that unlike in the Random Problem Class, in the Structured Problem Class problems there are no dominated facility types, since $S_{k_1} > S_{k_2}$ and $G_{jk_1} > G_{jk_2}$ for all $j \in J$ and $k_1 > k_2$.

Both of the problem classes also reflect the system configuration constraint (10) with $n_{\max} = \min\{n_J, 10\}$. For each problem classes, 12 problem sets were generated of varying dimension. A necessary condition for problem feasibility is that $\max_{k \in K} S_k \geqslant \max_{f \in F} D_f$. Hence, only problems

satisfying this condition were included in the problem sets. Each of the resulting 24 problem sets was then replicated randomly 10 times, resulting in a total of 240 test problems.

Note that the problems in the Random Problem Class often contain dominated facility types (i.e., a facility type for which there exists an alternative facility type with equal or greater capacity but at a lower fixed cost). A preprocessing step was therefore included in all of the solution procedures in which the dominated facility types at each location are identified at the outset of the procedure and are then eliminated from further consideration by setting their corresponding fixed costs equal to an arbitrarily large value.

The results of the computational experiments are reported in Tables 1–4. Tables 1 and 2 report the performance of the MPCFL heuristic and branch-and-bound algorithm on the Random and the Structured Problem Classes, respectively. The first five columns of these tables specify the problem dimension. The next two columns provide information on the performance of the MPCFL heuristic. The first of these columns indicates the average Actual Percent Optimality of the heuristic, as measured by $100[1-(Z^{\mathrm{H}}-Z^*)/Z^*]$ and reported as a percentage, where Z^{H} is the objective value of the best solution found at the top node of the tree, and Z^* is the optimal objective value to MPCFL. The next column reports the average CPU time (in seconds) of the heuristic. The last two columns provide statistics for the branch-and-bound algorithm, giving the average number of nodes that were generated in the search tree and the average CPU time required to solve the problems to optimality. This CPU time is also measured in seconds and is exclusive of the initial heuristic solution time.

From the tables we observe that the subgradient-optimization Lagrangian-relaxation-based heuristic is extremely effective in identifying high quality solutions to MPCFL. Overall, the solutions generated are on average within 2% of optimality, with the Actual Percent Optimality (APO) for the Random Problem Class and for the Structured Problem Class averaging 97.72% and 98.37%, respectively. The MPCFL heuristic is also seen to require only a modest amount of computational effort.

The MPCFL branch-and-bound algorithm is also observed to be effective, successfully solving all 240 test problems to optimality. This includes problems in the Random Problem Class with $n_I = 30$ customers, $n_J = 20$ facilities, $n_F = 5$ product families, and $n_K = 5$ facility types, which can have as many as 200 0-1 variables and 3000 continuous variables. The data in the tables indicate that the solution effort of the algorithm increases substantially as the size of the problem increases. This is not surprising, given that MPCFL is NP-hard. Comparing the two tables, we also observe that the Structured Problem Class problems require more

Table 1
MPCFL heuristic and branch-and-bound algorithm performance on random problem class

					MPCFL heuristic		Branch-and-bound algorithm	
n_I	n_J	n_F	n_K	$n_{\max}$	Actual percent optimality	CPU (s)	Number of nodes	CPU (s)
5	5	5	5	5	98.50%	0.08	54.8	0.06
10	5	5	5	5	98.96	0.07	71.3	0.11
10	10	5	5	10	97.57	0.08	199.1	0.48
15	10	5	5	10	98.97	0.09	357.8	2.33
20	10	5	5	10	97.22	0.12	587.5	7.60
15	15	5	5	10	97.91	0.11	1284.1	14.53
20	15	5	5	10	97.85	0.12	3202.8	68.12
25	15	5	5	10	97.23	0.18	3419.9	196.37
30	15	5	5	10	96.46	0.31	3279.4	354.07
20	20	5	5	10	97.71	0.12	9296.7	374.60
25	20	5	5	10	97.36	0.28	12626.5	1255.99
30	20	5	5	10	96.90	0.53	29463.8	5059.03

Table 2
MPCFL heuristic and branch-and-bound algorithm performance on structured problem class

					MPCFL heuristic		Branch-and-bound algorithm	
n_I	n_J	n_F	n_K	$n_{\max}$	Actual percent optimality	CPU (s)	Number of nodes	CPU (s)
5	5	3	3	5	97.48%	0.07	131.2	0.12
10	5	3	3	5	99.45	0.07	177.2	0.16
10	10	3	3	10	98.56	0.07	1644.4	1.98
15	10	3	3	10	99.45	0.08	544.2	1.23
20	10	3	3	10	98.71	0.09	827.8	5.31
15	15	3	3	10	98.29	0.08	2628.4	11.16
20	15	3	3	10	99.38	0.10	4829.0	58.30
25	15	3	3	10	97.86	0.11	18052.6	535.31
30	15	3	3	10	97.39	0.19	52500.8	2638.05
20	20	3	3	10	98.64	0.08	11357.8	294.74
25	20	3	3	10	98.45	0.14	55774.4	3186.66
30	20	3	3	10	96.83	0.25	70342.8	6361.74

Table 3
CD-EBD and EBD performance on random problem class

					CD-EBD algorithm				
					CD phase			EBD phase	EBD algorithm
n_I	n_J	n_F	n_K	$n_{\max}$	Actual percent optimality	Gap	CPU (s)	CPU (s)	CPU (s)
5	5	5	5	5	98.22%	4.84%	3.46	1.15	1.51
10	5	5	5	5	98.40	11.54	3.92	5.23	13.49
10	10	5	5	10	95.93	14.03	1003.23	4412.10	9309.19

Table 4
CD-EBD and EBD performance on structured problem class

					CD-EBD algorithm				
					CD phase			EBD phase	EBD algorithm
n_I	n_J	n_F	n_K	$n_{\max}$	Actual percent optimality	Gap	CPU (s)	CPU (s)	CPU (s)
5	5	3	3	5	99.45%	3.04%	0.65	0.08	0.15
10	5	3	3	5	98.90	8.07	0.53	0.14	0.23
10	10	3	3	10	97.58	7.83	305.58	2.45	4.34

effort to solve to optimality using this algorithm than do the Random Problem Class problems.

Tables 3 and 4 report the performance of the CD-EBD and EBD algorithms on the Random and Structured Problem Classes, respectively. For the CD-EBD algorithm, these tables report the APO of the solution obtained at the end of the CD phase and also the CPU times required by the CD and EBD phases. In addition, the CPU time required by the EBD algorithm is also reported. We observe that the effort required to solve the test problems using these algorithms also increases with problem size. This increase in solution effort, however, is significantly more substantial than that observed earlier in Tables 1 and 2. This is readily apparent in the Random Problem Class problems, where, for example, the optimal solution of the problems with the $n_I = n_J = n_{\max} = 10$ and $n_F = n_K = 5$ required on the average 5415.3 seconds for the CD-EBD Algorithm and 9309.2 s for the EBD algorithm. Comparing the average solution times for each problem set in these tables with the

corresponding solution times in Tables 1 and 2, the MPCFL branch-and-bound algorithm when used in conjunction with the MPCFL heuristic is seen to dominate substantially both the CD-EBD and EBD algorithms. Because of the observed superior performance of the MPCFL algorithm, together with the prohibitively large solution times for even relatively small problems in the Random Problem Class, the testing of the CD-EBD and EBD algorithms was limited to a subset of the problem sets in these two problem classes (as reported in the tables).

We also observe that the quality of the heuristic solutions (as measured by the APO) generated by the CD phase is comparable to that obtained by the MPCFL heuristic, averaging 97.52% and 98.64% over the problem classes in the Random and Structured Problem Classes, respectively, as compared with averages of 98.34% and 98.50% for the corresponding problem classes in Tables 1 and 2. The corresponding solution times, however, are significantly larger for the CD approach. Tables 3 and 4 also report the average "Gap" between the upper bound, $\bar{z}_{CD}$, and lower bound, $\underline{z}_{CD}$, obtained at the end of the CD phase, as measured by $100(\bar{z}_{CD} - \underline{z}_{CD})/\bar{z}_{CD}$. This Gap measure provides some indication of the relative size of the duality gaps among the different problem classes. There appears to be a positive correlation between problem difficulty (as measured by the solution time) and the size of the Gap. The Structured Problem Classes exhibit smaller Gap measures than do the Random Problem Classes, and the corresponding solution times are smaller in Tables 3 and 4. The most difficult problem class for the CD-EBD and EBD algorithms are the problems with the largest Gap measure. We note, however, that this Gap measure does not necessarily provide an accurate estimate of the size of the actual duality gap, since the use of the tolerance factor to accelerate the (CPU time) performance of the CD phase will affect the validity of the lower bound; also, the stopping rule used to establish convergence of the upper and lower bounds at the end of the CD phase will also affect this Gap measure, with more stingent stopping rules allowing for a more accurate measure but (also requiring more computation time).

6. Conclusions

This paper has considered Lagrangian-relaxation-based exact and heuristic solution procedures for MPCFL. The branch-and-bound algorithm is successfully able to take advantage of the underlying structure of the MPCFL, which through Lagrangian relaxation, can be decomposed into uncapacitated facility location problems and easily-solvable 0-1 knapsack problems. This Lagrangian relaxation, in conjunction with the use of the strong formulation of the UFL, the Erlenkotter (1978) dual-based procedure for the UFL, and the application of subgradient optimization, combine to form an effective bounding procedure for MPCFL. The computational experiments demonstrated that the branch-and-bound algorithm offers a more effective means of obtaining optimal solutions to MPCFL than does either the embedded Benders' decomposition algorithm (Lee, 1991) or the cross decomposition algorithm (Lee, 1993). The MPCFL heuristic was also seen to be effective in generating high-quality solutions to MPCFL. The effectiveness of this approach once again reinforces the utility of subgradient-optimization Lagrangian-relaxation-based heuristics for solving combinatorial optimization problems.

Acknowledgements

The authors would like to thank Dr. Vikas Sinha and the anonymous reviewers for their helpful comments. The authors would also like to thank Professor Donald Erlenkotter for making available a copy of the DUALOC code. This research was supported in part by the Business Associates Fund and the Ann Lewallen Spencer Fund. The authors are grateful for this assistance.

References

Akinc, U., Khumawala, B.M., 1977. An efficient branch and bound algorithm for the capacitated warehouse location problem. Management Science 23, 585–594.

Balas, E., 1965. An additive algorithm for solving linear programs with zero-one variables. Operations Research 13, 517–546.

Barcelo, J., Casanovas, A., 1984. A heuristic Lagrangean algorithm for the capacitated plant location problem. European Journal of Operational Research 15, 212–226.

Barcelo, J., Hallefjord, Å., Fernandez, E., Jörnsten, K., 1990. Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing. OR Spektrum 12, 79–88.

Cornuéjols, G., Fisher, M.L., Nemhauser, G.L., 1977. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. Management Science 23, 789–810.

Efroymson, T., Ray, T., 1966. A branch bound algorithm for plant location. Operations Research 14, 361–368.

Erlenkotter, D., 1978. A dual-based procedure for uncapacitated warehouse location problems. Operations Research 26, 992–1009.

Fisher, M.L., 1981. The Lagrangian relaxation method for solving integer programming problems. Management Science 27, 1–18.

Fisher, M.L., 1985 An applications oriented guide to Lagrangian relaxation. Interfaces 15(2), 10-21.

Gavish, B., Pirkul, H., 1986. Computer and database location in distributed computing systems. IEEE Transactions on Computers 35, 583–590.

Geoffrion, A.M., 1974. Lagrangean relaxation for integer programming. Mathematical Programming Study 2, 82–114.

Geoffrion, A.M., Graves, G.W., 1974. Multicommodity distribution design by Benders decomposition. Management Science 20, 822–844.

Geoffrion, A.M., McBride, R., 1978. Lagrangean relaxation applied to capacitated facility location problems. AIIE Transactions 10, 40–47.

Goffin, J.L., 1977. On the convergence rates of subgradient optimization methods. Mathematical Programming 13, 329–347.

Guignard, M., Opaswongkarn, K., 1990. Lagrangian dual ascent algorithms for computing bounds in capacitated plant location problems. European Journal of Operational Research 46, 73–83.

Held, M.H., Wolfe, P., Crowder, H.D., 1974. Validation of subgradient optimization. Mathematical Programming 6, 62–88.

Jörnsten, K., Näsberg, M., 1986. A new Lagrangian relaxation approach to the generalized assignment problem. European Journal of Operational Research 27, 313–323.

Khumawala, B.M., 1972. An efficient branch and bound algorithm for the warehouse location problem. Management Science 18, 718–731.

Klincewicz, J.G., Luss, H., 1986. A Lagrangean relaxation heuristic for capacitated facility location with single-source constraints. Journal of the Operational Research Society 37, 495–500.

Klincewicz, J.G. , Luss, H., 1987 A dual-based algorithm for multiproduct uncapacitated facility location. Transportation Science 21, 198-206.

Krarup, J., Pruzan, P.M., 1983. The simple plant location problem: Survey and synthesis. European Journal of Operational Research 12, 36–81.

Lee, C.Y., 1991. An optimal algorithm for the multiproduct capacitated facility location problem with a choice of facility type. Computers and Operations Research 18, 167–182.

Lee, C.Y., 1993. A cross decomposition algorithm for a multiproduct-multitype facility location problem. Computers and Operations Research 20, 527–540.

Mazzola, J.B., Neebe, A.W., 1986. Resource-constrained assignment scheduling. Operations Research 34, 560–572.

Mazzola, J.B., Schantz, R.H., 1997. Multiple-facility loading under capacity-based economies of scope. Naval Research Logistics 44, 229–256.

Neebe, A.W., Rao, M.R., 1983. An algorithm for the fixed-charge assigning users to sources problem. Journal of the Operational Research Society 34, 1107–1113.

Nemhauser, G.L. Wolsey, L.A., 1988. Integer and Combinatorial Optimization, Wiley, New York .

Sridharan, R., 1995. The capacitated plant location problem. European Journal of Operational Research 87, 203–213.

Tempelmeier, H., Derstroff, M., 1996. A Lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. Management Science 42, 738–757.

Van Roy, T.J., 1983. Cross decomposition for mixed integer programming. Mathematical Programming 25, 46–63.

Van Roy, T.J., 1986. A cross decomposition algorithm for capacitated facility location. Operations Research 34, 145–163.