Journal of Heuristics, 11: 307–336, 2005

Parametric Ghost Image Processes for Fixed-Charge Problems: A Study of Transportation Networks

FRED GLOVER
Leeds School of Business, UCB 419, University of Colorado, Boulder, CO 80309-0419
email: Fred.Glover@colorado.edu

MEHDI AMINI
Fogelman College of Business and Economics, The University of Memphis, Memphis, TN 38152
email: mamini@memphis.edu

GARY KOCHENBERGER
College of Business Administration, University of Colorado, Denver, CO 80217-3364
email: Gary.Kochenberger@cudenver.edu

Submitted in December 2003 and accepted by David Woodruff in May 2005 after 4 revisions

Abstract

We present a parametric approach for solving fixed-charge problems first sketched in Glover (1994). Our implementation is specialized to handle the most prominently occurring types of fixed-charge problems, which arise in the area of network applications. The network models treated by our method include the most general members of the network flow class, consisting of *generalized networks* that accommodate flows with gains and losses.
Our new parametric method is evaluated by reference to transportation networks, which are the network structures most extensively examined, and for which the most thorough comparative testing has been performed. The test set of fixed-charge transportation problems used in our study constitutes the most comprehensive randomly generated collection available in the literature. Computational comparisons reveal that our approach performs exceedingly well. On a set of a dozen small problems we obtain ten solutions that match or beat solutions found by CPLEX 9.0 and that beat the solutions found by the previously best heuristic on 11 out of 12 problems. On a more challenging set of 120 larger problems we uniformly obtain solutions superior to those found by CPLEX 9.0 and, in 114 out of 120 instances, superior to those found by the previously best approach. At the same time, our method finds these solutions while on average consuming 100 to 250 times less CPU time than CPLEX 9.0 and a roughly equivalent amount of CPU time as taken by the previously best method.

Key Words: fixed-charge problems, networks, generalized networks, ghost image processes, tabu search

1. Introduction

Diverse and exceedingly widespread applications of the fixed-charge problems arise in optimization theory and practice. Documented practical applications include natural gas pipeline systems (Rothfarb, et al., 1970), offshore platform drilling (Balas and Padberg, 1976), bank account location (Cornuejols et al., 1977), distribution center location (Nozick and Turnquist, 1998a and 1998b), telecommunication network switching (Luna et al.,

1987), and network design (Mirzain, 1985; Crainic, Frangioni, and Gendron 2001). The largest and undoubtedly the most significant body of these problems arise in network-related applications, as discussed in Glover Klingman and Phillips (1992).

The capacitated fixed-charge problem may be written in the form

$$\begin{array}{lll} \textbf{FC} & \text{Minimize} & x_o[\text{FC}] = cx + F(x) \\ & \text{Subject to} & Ax = b \\ & & U \geq x \geq 0 \end{array}$$

where $F(x) = \sum (F_j(x_j) : j \in N)$ and N denotes the index set of x. The "fixed charge" function $F_j(x_j)$ is expressed by $F_j(x_j) = F_j$, a positive constant, if $x_j > 0$ and by $F_j(x_j) = 0$ if $x_j = 0$. We may allow for the case where not all components of x are fixed charge variables by defining a separate index set for these variables. This introduces a trivial bookkeeping change in our following development, and thus for simplicity of notation we refer to N as if each variable carries an associated fixed charge. (We could also stipulate that $F_j = 0$ for variables without fixed charges.) The problem FC may also be written as a 0–1 mixed integer program by introducing a 0–1 vector $z = (z_j : j \in N)$ and letting $F = (F_j : j \in N)$, to give

$$\begin{array}{lll} \textbf{FC-MIP} & \text{Minimize} & x_o[\text{FC}] = cx + Fz \\ & \text{Subject to} & Ax = b \\ & & Uz \geq x \geq 0 \\ & & z = \{0, 1\} \end{array}$$

In the past four decades, a host of exact and approximation approaches have been offered for solving fixed-charge problems. The observation of Hirsch and Dantzig (1968), which showed that optimal solutions to the fixed-charge problem occur at extreme points, opened a fertile area for developing a class of exact methods. These and other types of exact methods include cutting plane approaches (Rousseau, 1973), vertex ranking strategies (Murty, 1968; McKeown, 1975), and a number of branch and bound approaches with penalty based search tree pruning mechanisms and capacity improvement techniques (Gray, 1971; Kennington, 1976; Kennington and Unger, 1976; Fisk and McKeown, 1979; McKeown and Sinha, 1980; Barr, Glover and Klingman, 1981; McKeown, 1981; Cabot and Erenguc, 1984; and 1986; Schaffer, 1989; McKeown and Ragsdale, 1990; Palekar, Karwan and Zionts, 1990; Lamar and Wallace, 1997; Bell, Lamar and Wallace, 1999; Glover, Amini, and Kochenberger, 2003; and Ortega and Wolsey, 2003). Inherent exponential growth of computational effort required by the exact methods for fixed-charge problems in general and for its network instances in particular has confined these applications to problems with a low level of complexity, typically having limited size and restricted ranges of the fixed-charges.

These practical limitations have occasioned a considerable research effort focusing on approximation approaches, heuristics and metaheuristics. These methods take advantage

of various strategies; some utilizing relaxation approaches (Wright et al., 1989 and 1991) and others employing extreme point search techniques and embedded network procedures (Balinski, 1961; Kuhn and Baumol, 1961; Denzler, 1964; Dwyer, 1966; Cooper and Drebes, 1967; Cooper, 1975; Walker, 1976; Stienberg, 1970 and 1977; Shetty, 1990; Diaby, 1991; Khang and Fujiwara, 1991; Sun and McKeown, 1993; Gottlieb and Eckert, 2002; Adlakha and Kowalski, 2003). A notable metaheuristic approach, a tabu search method by Sun et al. (1998), has established itself as the best method for solving a collection of fixed-charge transportation network problems that constitutes the largest body of randomly generated problems in the literature.

The present work is based on a solution design proposed in the *ghost image process* (GIP) approach by Glover (1994). GIP can be applied to a wide variety of optimization problems. For example, an interesting application of GIP ideas in the context of calculating the minimum covariance determinant estimators has been developed by Woodruff (1995).

Within the context of fixed-charge networks, the proposed GIP design involves a parameterization of the objective function that is progressively modified and coordinated with a metaheuristic improvement procedure that incorporates basic tabu search notions. With regard to the parameterized objective function, related ideas have also recently been proposed by Kim and Pardalos (1999 and 2000), utilizing essentially the same form of the objective but excluding the metaheuristic improvement strategies introduced in the earlier paper. An improved approach integrating the Kim and Paradalos method with Lagrangian perturbation and strategies inspired by tabu search has more recently been applied to multicommodity fixed-charge network problems in Gendron, Potvin, and Soriano (2003a and 2003b) and Crainic, Gendron, and Hernu (2004). As we show, the inclusion of a simple version of these strategies, coordinated with the basic parametric updating ideas of the original GIP proposal, yields a remarkably effective method for fixed charge transportation networks.

The remainder of this paper is organized as follows. A background discussion of the parametric ghost image process, and a parametric GIP algorithmic development for the fixed-charge network problem is presented in Section 2. Section 3 describes implementation strategies. Computational experiments, including a discussion of testbed problem characteristics, hardware platform, and solution results are presented in Section 4. Finally, Section 5 summarizes our conclusions.

2. The parametric GIP approach

The general form of parametric ghost image processes encompasses a wide range of features that derive from a collection of problem solving principles detailed in Glover (1994). In this paper we focus only on the specifics of applying the GIP framework to fixed-charge problems.

Within this setting, our approach exploits the FC problem by introducing a non-negative parameter vector $v = (v_j : j \in N)$ and an associated parameterized cost vector given by $c(v) = (c_j + F_j/v_j : j \in N)$ to give a parametric linear programming relaxation of the

fixed-charge problem

$$\begin{array}{lll} \textbf{LP}(v) & \text{Minimize} & x_o(v) = c(v)x \\ & \text{Subject to} & Ax = b \\ & & U \geq x \geq 0 \end{array}$$

Let $p_j = F_j/v_j$ denote the "parameterized penalty" associated with c_j. Then we also write $c(v) = (c_j + p_j : j \in N)$, and note that p_j allocates the fraction $1/v_j$ of the fixed cost F_j to the total cost of x_j. We apply the convention that a denominator v_j close to 0 (smaller than a chosen ε value) translates into setting $p_j = \text{M}$, where M is a large positive number.

At an extreme, where all $v_j = \infty$ (hence all $p_j = 0$), we have $c(v) = c$, and obtain the simple linear programming relaxation

$$\begin{array}{lll} \textbf{LP}: & \text{Minimize} & x_o = cx \\ & \text{Subject to}: & Ax = b \\ & & U \geq x \geq 0 \end{array}$$

The method sketched in Glover (1994) begins by solving LP, and then solves a succession of problems LP(v) produced by progressively modifying and updating v_j in alternation with applying an improvement method for enhancing the solution to LP(v), utilizing adaptive memory strategies from tabu search.

An outline of the parametric GIP method for the FC problem can be described as follows. Each solution obtained throughout these steps is evaluated as a candidate for the best solution x^* currently found.

Step 0: Solve LP, yielding an optimum solution as a first candidate for x^*, and set $v \leftarrow U$.
Step 1: Solve LP (v), yielding a solution x'.
Step 2: Starting from x', use restriction to obtain a refined solution and apply an Improvement Method to obtain a further refined solution x''.
Step 3: Update v as a function of its current value and x''. If a maximum allowed iteration is not reached return to Step 1. Otherwise, terminate the process with the best solution x^* at hand.

In the following, we give details of these steps as adapted to the present context. Throughout this exposition we use the convention of identifying the value of the (nonlinear) fixed charge objective function x_o[FC] for a given trial solution vector x (e.g., $x = x'$, x'', etc.) as x_o[FC: x]. The values U_o and U_j defined below are used as estimated upper bounds for x_j that will be introduced to replace the original bound U_j in certain calculations of the algorithm.

***Step 0:* Solve the linear program LP and create an initial parameter vector v.**

Additional Notation:

Iter: the current iteration of the algorithm,
λ: the current value of a scalar constant, where $0 \leq \lambda \leq 1$,
$\lambda_{\min}$: the minimum value for λ, where $0 \leq \lambda_{\min} \leq \lambda_{\max}$,
z_{iter}: the current number of non-improving iterations,
v_{iter}: the first number of iterations for which the parameter vector v is updated based on one strategy, while in the remaining iterations another strategy is used,
x^*: the best solution found so far. (We assume the fixed charge objective function value for this solution, x_o[FC:x*], is automatically updated whenever x^* changes.)

We also make reference to:

U_o: a scalar identifying the maximum x_j value in the first solution to the problem LP.
U_j: a scalar identifying the maximum value obtained the variable x_j within the first v_{iter} iterations of the search process,
Max_{iter}: the maximum allowable iterations for the algorithm.

Initialization Steps:

0.1. Set $Iter \leftarrow 0$, $z_{\text{iter}} \leftarrow 0$, $\lambda \leftarrow \lambda_{\min}$, and $v \leftarrow U$.
0.2. Solve LP, yielding an optimal linear solution x'.
0.3. Initialize $x^* \leftarrow x'$.
0.4. Set $U_o \leftarrow \text{Max}\{x'_j : j \in N\}$ and $U_j \leftarrow x'_j$, $j \in N$.

Step 1: **Generate and solve LP (v) to obtain a new trial solution x'.**

Steps:

1.1. Set $Iter \leftarrow Iter + 1$.
1.2. If $Iter > Max_{\text{iter}}$, then terminate the process with the best solution found so far, x^*, and its associated objective function value, x_o[FC:x^*]
1.3. Solve LP(v) by linear programming post-optimization, yielding an optimal solution x'.
1.4. For the associated fixed-charge objective value x_o[FC:x'] if x_o[FC:x'] $< x_o$[FC:x^*], then set $x^* \leftarrow x'$.
1.5. If $Iter \leq v_{\text{iter}}$ then update $U_j \leftarrow \text{Max}\ (U_j, x'_j)$, for each $j \in N$.

Step 2: **Improve the current solution x$'$.**

Additional Notation:

τ: the number of non-improving iterations after which the search is terminated,
B: the index set of basic variables in the current solution, x, where $B \subset N$,
NB: the index set of current nonbasic variables in the current solution, x, where $N = B \cup NB$, and
M: a large positive number.

Steps:

2.1. ***Phase I***: ***Refinement by LP Restriction.***

2.1.1. By reference to x' from Step 1, set $v_j \leftarrow 0$ if $x'_j > 0$; and $v'_j \leftarrow M$ if $x'_j = 0$, for $j = 1, \ldots, N$.
2.1.2. Starting from the LP basis that produced x', identify and post-optimize the problem LP(v), yielding an optimal solution x''.
2.1.3. If x_o[FC:x''] $< x_o$[FC:x^*], then set $x^* \leftarrow x''$.
2.1.4. If $Iter \leq v_{\text{iter}}$ then update

$$U_j \leftarrow \text{Max}(U_j, x''_j), \text{ for each } j \in N.$$

2.2. ***Phase II***: ***Improvement Phase.***

2.2.1. Consider the current FC basis representation, yielding x''.

2.2.2. Set $v \leftarrow 0$, hence, the current objective function for LP(v) is $x_o(v) = cx''$.

2.2.3. Set $j^* \leftarrow 0$, and $x_{oj^*} \leftarrow \infty$, initializing the index of a "best" nonbasic variable x_{j^*} and its associated objective function change x_{oj^*} (improving if negative), caused by a pivot that brings x_{j^*} into the current basis for LP(v). (This basis can change in step 2.2.5 below.) Also, set $k^* \leftarrow 0$, initializing the index of a current basic variable x_{k^*} that will leave the basis when x_{j^*} enters.

2.2.4. Solve LP(v) and consider each variable x_j, $j \in NB$ as a potential candidate to enter the current basis.

2.2.4.1. Conduct a ratio test to determine the outgoing basic variable x_k and determine the change r_j in the objective function value of LP(v) caused by the associated pivot exchanging x_j and x_k. If x_j stays nonbasic by moving from one of its bounds (lower or upper) to the other, then no basic variable x_k is selected to leave the basis, coded by setting $k = 0$ (as in the initialization of k^*).

2.2.4.2. Determine the net impact of the potential pivot on the objective function x_o[FC:x''] (as a result of changing x'') calculated by

$$x_{oj} = r_j + \sum (G_h : h \in P)$$

where P is the set of arcs on the basis exchange path determined by and including the arc for the entering variable x_j, and where G_h is defined relative to the fixed charge F_h by the following pivot transitions:[1]

$$x_h = 0 \text{ to } x_h > 0 : G_h = F_h$$
$$x_h > 0 \text{ to } x_h = 0 : G_h = -F_h$$
$$x_h \text{ unchanged} : G_h = 0.$$

2.2.4.3. If $x_{oj} < x_{oj^*}$ then set $j^* \leftarrow j$ (and hence $x_{oj^*} \leftarrow x_{oj}$).

2.2.5. If $x_{oj^*} < 0$ then perform the pivot with variable pair (j^*, k^*). Also, identify the associated new basic solution and designate it to be the solution x''. Then go to Step 2.2.3 to pursue further improvement.

2.2.6. Else, if $x_{oj^*} \geq 0$ then

2.2.6.1. If $x_o[\text{FC:}x''] < x_o[\text{FC:}x^*]$ then set $x^* \leftarrow x''$, and $z_{\text{iter}} \leftarrow 0$.
2.2.6.2. Else, if $x_o[\text{FC:}x''] \geq x_o[\text{FC:}x^*]$ then set $z_{\text{iter}} \leftarrow z_{\text{iter}} + 1$.

2.2.7. If $z_{\text{iter}} = \tau$ then terminate the search process with the best solution found x^* and its associated objective function value $x_o[\text{FC:}x^*]$ at hand.

2.2.8. If $Iter \leq v_{\text{iter}}$ then update

$$U_j \leftarrow \text{Max}(U_j, x''_j), \text{ for each } j \in N.$$

Step 3: **Use the improved solution x'' to update the parameter vector v.**

Additional Notation:

$\lambda_{\max}$: the maximum value for λ, where $0 \leq \lambda_{\max} \leq 1$,
λ_Δ: the λ increment, where $0 < \lambda_\Delta \leq \lambda_{\max}$,
λ_{liter}: the last iteration number at which the current λ is being incremented by λ_Δ,
λ_{iter}: the number of iterations after which the current λ is being incremented by λ_Δ,
ω: the greater than zero multiplier used in updating the parameter vector v′,

Steps:

3.1 If $Iter - \lambda_{\text{liter}} = \lambda_{\text{iter}}$ then set $\lambda \leftarrow \lambda + \lambda_\Delta$ and $\lambda_{\text{liter}} = Iter$.
3.2 If $\lambda > \lambda_{\max}$ then terminate the search process with the best solution found x^* and its associated objective function value $x_o[\text{FC:}x^*]$ at hand.
3.3 If $Iter \leq v_{\text{iter}}$ then for $j \in N$ update the parameter vector as follows:

$$v'_j \leftarrow \lambda\omega U_o + (1 - \lambda)x''_j$$

3.4 Else, for $j \in N$ update the parameter vector as follows:

$$v'_j \leftarrow \lambda\omega U_j + (1 - \lambda)x''_j$$

3.5 If $v = v'$ then diversify the parameter vector as follows:

$$v'_j \leftarrow \lambda(U_j - x''_j) + (1 - \lambda)v_j$$

3.6 Set $v \leftarrow v'$ and go to Step 1.

Note that U_j in Step 3.5 is the original bound for x_j, as opposed to the proxy bound U_j in Step 3.4

In the initialization step, Step 0, the original linear programming relaxation LP is solved obtaining an optimal solution, x'. The best solution is initialized with x' and its associated objective function with the "true" objective function, including both variable and fixed charges. Also, to initiate alternative formulas for updating the parameter vector v, the constant U_o is initialized to be the largest x_j value obtained in solving LP. In addition, the maximum solution value for each variable x_j is recorded in U_j.

In Step 1, if the iteration counter exceeds a pre-determined maximum value, Max_{iter}, then the search process is terminated with the best solution x^*_o and its associated objective function value x^*_o[FC]. Otherwise, starting from the most recently solved previous instance of LP(v) (which on the initial iteration corresponds to LP), the *new trial problem* LP(v) is solved to obtain an optimum solution, x'. The "true" fixed-charge objective function value for x' is calculated and the new trial solution replaces the incumbent solution if its "true" objective function is better than x^*_o[FC]. We continue to update the values U_j designated to maintain the maximum value attained by x_j for the first v_{iter} iterations.

To investigate the potential for further improvement to the current solution, x', in Step 2-Phase I the objective function coefficients of the variables with nonzero and (virtually) zero values are set to their variable costs and M, respectively, resulting in the specified form of LP(v), which is then solved by post-optimization, yielding x''. The main purpose of setting the cost of variables with the zero values in the trial solution to M is to maintain their values at zero during the current post-optimization process, and the variables alternatively could simply be masked during this step. Following the calculation of the true objective function value that accounts for fixed costs, the new solution replaces the incumbent, if it turns out to be a better solution. Also, in *Phase I* the value U_j, identifying the maximum value for each x_j throughout the first v_{iter} iterations, is updated.

To further improve the current solution, x'', *Phase II* focuses on the basis representation provided by the LP solution that produced x''. First, the current objective function coefficient vector is replaced by the original variable cost vector, c in defining the current LP(v) (i.e., v is set to the 0 vector) so that each solution will be evaluated relative to the original variable costs. Next, through a tentative pivot exploration process, each nonbasic variable is considered as a potential entering variable, $x_{j,}$ a ratio test is conducted to determine the associated leaving variable, $x_{k,}$ followed by calculation of the associated (total) fixed-charge objective function change, z_{jk}, for a basis exchange involving the variable pair indexed by (j, k). At the completion of the tentative pivot explorations, the variable pair that yields the greatest reduction in the total cost, including changes in the fixed charges, is selected for pivoting followed by post-optimization. To further improve the current solution, the process returns to the tentative pivot exploration phase, using the current basis representation.

The tentative pivot exploration continues until no further improving pivots are available. At this stage, again the true fixed-charge objective function value for the optimum solution at hand is calculated. If the new solution improves upon the best, it replaces the incumbent solution and the process moves to Step 3 to update the parameter vector. Otherwise; if the process arrives at Step2–Phase II for τ iterations without gaining any improvement in the best solution found, then the parametric ghost image process is halted. If the solution process is not terminated, the values U_j are updated and the algorithm moves to Step 3.

In Step 3, the value of constant λ is incremented every λ_{iter} iterations by the increment λ_Δ. If the updated value of λ exceeds the maximum value that λ can attain, $\lambda_{\max}$, then the algorithm terminates with the current best solution at hand. To update the parameter vector, two formulas are applied. The first formula, applied within the first v_{iter} iterations, is the one applied to the current improved solution obtained in Step 2–Phase II, involving x'', the constant U_o, and a pre-determined multiplier, ω. For the remaining iterations of the algorithm, the second formula is used, which differs from the first one by replacing the first two constants, U_o and ω, in the first term with the value U_j—representing the maximum value attained by each decision variable in the first v_{iter} iterations. In both formulas, the second term takes advantage of the current improved solution, x''. Also, when the updated parameter vector v' is the same as the previous vector v, diversification is achieved by a formula that incorporates the upper bound array U, x'', and the current parameter vector, v.After the parameter vector is updated, the process returns to Step 1 to generate and solve a new trial problem with the updated parameter vector.

Additional comments on the method's rationale

We elaborate briefly on the overall rationale of the method to tie the previous observations together. First, the different solutions generated by the method are related in the following manner. The original LP relaxation, LP, provides a lower bound solution, x' in Step 0. Given the basis provided by x' and setting $v = U$, LP (v) is post-optimized to obtain x'. Note that Step 0 is considered an initialization step and the bulk of the algorithm iterates between Steps 1 and 3. Given a new ghost image vector, v', produced by Step3, we solve in Step 1 a new ghost image of the original problem, LP(v), providing the solution x'. Given the x' basis, we solve the associated new LP(v) problem defined in Step 2.1.1 to obtain a new solution x''. Given the basis thus obtained, we create and solve a restricted ghost image problem LP(v) by setting some v_j values to M to force the 0-valued x' variables to remain 0. (Remaining v_j values are set to 0, and hence have no effect on the problem.) This yields a potentially (though not necessarily) better solution x'' that we submit to an improvement process to produce x''.

It may also be useful to comment on the motivation behind the two different ways of updating v. The first update is the more straightforward one, relying on an implicit assumption that a single value of U_o in the update formula provides a meaningful bound for all the variables involved. Consequently, in the second update formula we use a separate U_j value for each fixed charge variable in order to be more responsive to differences among the variables. We conjectured originally that a single U_o value may be best for the larger problems. Our findings, however, show that using both formulas (in conjunction) led to

improved solution quality on average across all problem sizes relative to the solutions provided by using either a single U_o value or the individual upper bound value, U_j.

3. Implementation in a generalized network context

While our experimental focus is on the fixed-charge transportation problem, we have designed a more general implementation that applies to fixed charge problems using generalized networks. In this section, we discuss characteristics of a solver applied to optimize or post-optimize a series of such problems

At the heart of our implementation for the fixed-charge transportation problem is a two-multiplier generalized network solver, GN2, developed by Glover (1996). The solver is capable of providing solutions to uncapacitated and capacitated generalized and pure networks. The modular characteristics of the GN2 procedure facilitates the implementation of heuristic/metaheuristic approaches. The current implementation extends the capabilities of GN2 to solve the class of fixed-charge generalized and pure network problems. Transformation strategies introduced by Glover, Klingman, and Phillips (1992) further allow fixed-charge optimization and other 0–1 optimization problems that at first seem to bear no connection with networks to be handled by the generalized network fixed-charge formulation, and thus provide access to a wide range of additional applications.

It is important to note that throughout the parametric GIP solution process there is only one problem that requires optimization from scratch, the original linear programming relaxation LP in Step 0. The remaining problems generated throughout the solution process differ only in their objective function coefficients, and hence can be handled by post-optimization. This translates into significant solution time improvement.

The algorithmic description presented in the previous section gives a basic implementation strategy used in Step 2.2 for the Phase II improvement process. In reality, our implementation differs slightly from this description. We apply a strategy that is somewhat more involved than simply using the "most improving" rule in selecting nonbasic arcs for the tentative pivot exploration step. The efficiency of this latter rule diminishes as the problem size increases, and hence to be made effective, the rule must be amended by embodying it within a candidate list strategy. A variety of such candidate list strategies are proposed in connection with tabu search, including aspiration plus, elite, successive filter, sequential fan, and bounded change candidate lists (Glover and Laguna, 1997). In our present implementation, we exploit the Elite Candidate List (ECL) strategy which begins by identifying a subset of nonbasic variables associated with the tentative pivots that create the greatest improvements (positive or negative). On successive stages of the exploration of tentative pivots, attention is restricted to members of this list in making the current evaluation to find the nonbasic variable that provides the most attractive current pivot alternative—the alternative that is actually implemented during the current stage of execution. The list is updated after a certain number of pivots are implemented, scanning a predetermined number of nonbasic variables in a manner similar to that of constructing the list initially. The parameters of the elite candidate list are the list size, the number of nonbasic variables to scan for refreshing the list, and the maximum number of nonbasic variables to select from the list before refreshing, denoted by l_{size}, l_{scan}, and l_{max}, respectively.

It is to be emphasized that we rely on notions employed in tabu search only in a very simple and rudimentary way. In this sense our approach constitutes a preliminary investigation, since the door is evidently open to incorporating a variety of more advanced tabu search strategies in the improving phase. Nevertheless, we have established that the current improvement approach coordinates with the parametric updating process to produce a highly effective procedure.

To carry out this coordination, the solution process can be implemented as a *multi-start* strategy or as an *extended sequential* approach. In a *multi-start* strategy, the process is executed for a sequence of consecutively updated scalars λ. For each value of λ in the sequence, the problem instance at hand is solved and the best solution found is updated. In this strategy, the best solution found for a value of λ does not provide a starting solution for the next value in the sequence. The *extended sequential method*, by contrast, spends a certain number of iterations using one value of the scalar λ before incrementing λ by λ_Δ to give the value used for the next λ_{iter} iterations. In this case, the best solution obtained from previous stages provides a starting solution for the current stage. We have coordinated the improvement procedure with parametric updating in our study by means of the extended sequential strategy.

4. Computational experiments, results, and analysis

Our algorithm is implemented in Compaq® Visual FORTRAN, Professional Edition, Version 6.1, under Microsoft® Visual Studio, Version 6.0. In compilation of the code, the "Full Optimization" option is utilized. The hardware platform for code development, compilation, and computational experiments is a Dell, Latitude Laptop, Pentium III, 1 GHZ, with 256K Cache running on Windows® 2000 operating system.

To investigate relative computational effectiveness, solution time and quality, of the current parametric GIP approach against the alternative methods requires the availability of a comprehensive fixed-charge transportation problem (FCTP) testbed, and access to the best solution quality and time for the testbed obtained by competing method(s). Although the search for an effective solution procedure for the FCTP has been in progress in the past four decades, throughout most of this time no common testbed and associated solution results have been available to researchers. Recently, however, a comprehensive FCTP testbed with different problem sizes and ranges of complexity has been provided by Sun et al. (1998). The FCTP testbed was used to conduct computational comparisons of their effective tabu search approach against competing exact and approximation methods.

The Sun et al. study found the Palekar, Karwan, and Zionts (1990) approach to be generally the most effective of the exact methods and the Steinberg (1970 and 1977) approach to be generally the most effective of the heuristic search procedures. Hence, they compared effectiveness of their tabu search method against these two selected approaches. Their computational experiments on their comprehensive FCTP testbed show that the efficiency of the tabu search procedure, on very small and easy problem sets, is comparable to the solution time required by the competing heuristic in obtaining solutions of the same or better quality. However, for testbed problems of larger size and higher degree of complexity, the tabu search method was three to four times faster than the heuristic

Table 1. Fixed-charge transportation testbed problems characteristics.

			Range of fixed costs	
Problem size	Total supply	Problem type	Lower limit	Upper limit
10 × 10	10,000	A	50	200
10 × 20	15,000	B	100	400
15 × 15	15,000	C	200	800
10 × 30	15,000	D	400	1,600
50 × 50	50,000	E	800	3,200
30 × 100	30,000	F	1,600	6,400
50 × 100	50,000	G	3,200	12,800
		H	6,400	25,600

approach, and found significantly better solutions for all problem instances. Implementation of the tabu search approach was in FORTRAN and all computational experiments were conducted on the IBM 9672, Model E01, mainframe computer with VM/CMS operating system.

Availability of the FCTP testbed, access to the Sun et al. TS implementation, and the fact that the Sun et al. tabu search method was established as the most effective approximation approach for the FCTP has motivated us to carry out comparative testing using the same FCTP testbed. This testbed includes eight problem types, A through H, each in seven problem sizes. For a given problem size, problem types differ from each other by the range of fixed costs, which increases upon progressing from problem type A through problem type H. Each problem type includes 15 randomly generated problems. All problems are 100% variable and fixed cost dense. The variable costs range over the discrete values from 3 to 8. The seven problem types present different levels of difficulty for alternative solution approaches. The problem sizes, types, supplies/demands, and fixed costs ranges are shown in Table 1. Using problem input files provided by Sun et al. (1998), problem instances are generated by a modified version of NETGEN (Klingman, Napier and Stutz, 1974; Barr, Glover, and Klingman, 1981).

For computational experiments, two sets of run parameters are used, common and specific run parameters. The common run parameters have the same values in solving all problem instances while the values of specific run parameters are determined based on the problem instance size. The common parameters include $\lambda_{\min}$, $\lambda_{\max}$, λ_{Δ}, ω, and τ with values equal to 0.25, 0.35, 0.01, 2, and 3, respectively. The specific run parameters encompass λ_{iter}, Max_{iter}, v_{iter}, l_{size}, l_{scan}, and $l_{\max}$. Given a value for λ_{iter}, Max_{iter} is set to $(\lambda_{\max} - \lambda_{\min} + 1)\,\lambda_{\text{iter}}$. Also, v_{iter} is assigned the value $\text{Max}_{\text{iter}}/4$. Hence, we set the values of run parameters associated with the elite candidate list as follows: $l_{\text{scan}} = d$, $l_{\text{size}} = 0.10d$, and $l_{\max} = 0.05d$, where d is the number of demand nodes in an FCTP instance.

While some of the specific run parameters related to the elite candidate list, including l_{size}, l_{scan}, and $l_{\max}$, are associated with the generalized network solver, the remaining run parameters are specific to the GIP procedure. Selection of values for the three candidate

lists' specific run parameters is based on the past three decades of studies on network optimization, requiring no calibrations. Calibrations of the remaining specific and common run parameters were conducted by applying a systematic approach.

Our computational experimental plan divided the testbed problems into two subclasses. The first six problem sizes constitute a preliminary "small and easy" set and the seventh problem size constitutes a "large and difficult" set. To calibrate run parameters, from each problem size within the first class, we randomly selected two problem instances, creating a subset of 12 "small and easy" problems. Also, from the second class, we randomly selected three problem instances from each of the eight problem types, A through H, obtaining a subset of 24 preliminary "large and difficult" problems. Given the "small and easy" subset, we solved the problems with $[\lambda_{\min}, \lambda_{\max}]$ where the lower and upper values of interval range within interval [0.0, 0.90], while allowing (a) the value of λ_{Δ} to range (in sequence) over the values 0.10, 0.07, 0.05, 0.03, 0.02, and 0.01; (b) the value of λ_{iter} to range from 100 to 500, in increments of 100; and (c) the ω value to increase from 2 to 10, in increments of 2. The value of τ was set to Max_{iter}, in all run parameter combinations.

For each of the run parameter combinations, we recorded the problems best objective function obtained, the CPU times to reach the best solutions found, and the total execution CPU times. Comparing the results for all run parameter combinations, we identified the combination that resulted in the largest number of new best solutions. The final parameter calibrated was τ. Given the chosen run parameter combination, we re-solved the "small and easy" problem subset with the value of τ ranging from 1 to 10, in increments of 2. The best of the run parameter combinations identified was used in the proceeding comparative computation experiments for the "small and easy" problem subset.

To determine the best run parameter combination for the "large and difficult" subset, we applied a similar calibration strategy. Knowing that the subset includes larger and more complex problems, the only change we introduced in the calibration was to extend the upper values of λ_{iter} from 500 to 1000. The best run parameter combination identified in the calibration process was then utilized in solving the entire set of 120 "large and difficult" testbed problems, presented below.

Our first computational experiment includes a representative subset of the first six problem sizes, including two randomly selected problem instances from each size. Henceforth, we refer to this sample problem set as the (final) "small and easy" set. It is important to note that this problem subset is different from the preliminary one we randomly selected for the purpose of the parametric GIP calibration. This sample problem set, is solved by CPLEX 9.0 twice. The first run does not take advantage of the AMPL pre-processing option and the mixed-integer programming solver pre-processing option, while the second run activates the pre-processing options for both AMPL and the MIP solver. The maximum time limit for CPLEX 9.0 imposed on these runs was 11,000 CPU seconds, 1000 times the average solution time of the same problem set with the parametric GIP method.

Problem size and ID, objective function values, lowest CPU time for the best solutions found by the two runs, and lowest total execution CPU times for both runs are reported in Table A-1 in the appendix. A comparison of the objective functions indicates that both pre-processing options found optimal solutions for the first eight problems in the set. Neither of the two runs found optimal solutions to the four remaining problems within the

Table 2. Specific run parameters for the first computational experiment.

	Specific Run Parameters				
Problem size (s, d)	λ_{iter}	Max_{iter}	$l_{\text{scan}} = d$	$l_{\text{size}} = 0.10d$	$l_{\text{max}} = 0.05d$
10×10	100	1,100	10	1	1
15×15	200	2,200	15	2	1
10×20	300	3,300	20	2	1
10×30	300	3,300	30	3	1
50×50	300	3,330	50	5	2
30×100	300	3,300	100	10	5

maximum CPU time limit. For three out of four remaining problems, CPLEX 9.0 with the pre-processing option found solutions better than the run without pre-processing. The minimum, maximum, and average CPU time for the best solutions found are 0.02, 10.77, and 2.79, respectively. Also, the minimum, maximum, and average execution CPU times are 0.11, 11,000, and 3.69, respectively.

Given the same sample testbed problems and the best objective function values and CPU times provided by CPLEX 9.0 from the first experiment, the second computational experiment focused on comparing relative performances of CPLEX 9.0, the Sun et al. (1998) TS code, and the parametric GIP implementation. The run parameters applied to solve the problem set by the TS code are the same ones reported by Sun et al. (1998). The specific run parameters used for the parametric GIP code in this experiment are shown in Table 2. Also, a maximum CPU time limit similar to the first experiment was imposed on CPLEX 9.0. The computational results for this experiment, summarized in Table 3, show the problem size and ID, the best objective function values found, the best solution CPU times, and the execution CPU times for the three solution methods.

Examination of Table 3 indicates that the optimal solutions found by CPLEX 9.0 for five out of twelve problem instances are matched by the two competing methods. For two other testbed problems, the solutions found by the parametric GIP are verified to be optimal by CPLEX 9.0. For the same two problems the TS method finds solutions of lower quality. The solution quality of another pair of problems obtained by CPLEX 9.0 were not matched by the other two methods. For the three largest size problems in the set, the parametric method obtained solutions of higher quality than CPLEX 9.0 and the TS method.

The CPU time ranges and averages for the best solutions found by CPLEX 9.0, TS method, and the parametric GIP approach are 0.02 to 10,767 with an average of 2,791, 0 to 4.39 with an average of 0.60, and 0 to 36.60 with an average of 6.34, respectively. For the same codes, the execution CPU time ranges and averages are 0.11 to 11,000 with an average of 3,691, 0.03 to 15.87 with an average of 3.52, and 0.18 to 39.97 with an average of 11.18, respectively. The average CPU time to find the best solution and the average execution CPU times indicate that the TS method runs faster than the two other codes, followed by the parametric GIP method, and CPLEX 9.0.

It is important to note that the relative efficiency of the TS and the parametric GIP approaches are affected by differences in the implementation strategies applied in the two

Table 3. Solution Results for a Sample of "Easy" FCTPs.

Problem Size (s, d)	Problem ID	CPLEX 9.0 Best O.F.V.	CPLEX 9.0 Best time	CPLEX 9.0 Exec. time	Sun et al. (1998) Best O.F.V.	Sun et al. (1998) Best time	Sun et al. (1998) Exec. time	Parametric GIP Best O.F.V.	Parametric GIP Best time	Parametric GIP Exec. time
10 × 10	N104	40,258	0.05	0.11	40,258	0.00	0.03	40,258	0.12	0.26
	N107	42,029	0.02	0.33	42,029	0.01	0.04	42,029	0.00	0.18
15 × 15	N204	54,502	2.56	2.70	54,502	0.00	0.11	54,502	0.00	1.12
	N207	**53,596**	4.75	5.37	53,601	0.09	0.13	53,601	0.00	1.06
10 × 20	N304	**56,366**	0.15	1.00	56,391	0.01	0.13	**56,366**	0.00	0.78
	N307	49,742	0.50	4.61	49,742	0.00	0.12	49,742	0.00	0.89
10 × 30	N504	57,130	26.64	237.33	57,130	0.00	0.28	57,130	1.15	2.86
	N507	**52,903**	19.81	43.57	52,977	0.10	0.24	**52,903**	1.84	1.39
50 × 50	N1004	163,599	10,766.50	11,000	163,793	4.39	5.48	**163,585**	4.03	22.95
	N1007	162,300	10,451.20	11,000	162,313	1.96	6.75	**162,237**	26.62	35.43
30 × 100	N2004	104,046	4,851.22	11,000	104,193	0.00	15.87	**104,001**	36.60	39.97
	N2007	**104,147**	7,372.78	11,000	104,341	0.68	13.08	104,256	5.75	27.23
Min.		40,258	0.02	0.11	40,258.00	0.00	0.03	40,258.00	0.00	0.18
Max.		163,599	10,767	11,000	163,793.00	4.39	15.87	163,585.00	36.6	39.97
Ave.		78,385	2,791.35	3,691.25	78,439.17	0.60	3.52	78,384.17	6.34	11.18
Std.		44713.23	4364.52	5398.27	44,761.73	1.32	5.64	44,703.19	12.13	15.48

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

codes. The use of a specialized transportation network solver, as used in the TS code, versus a generalized network solver, as used in the GIP code, improves the solution CPU time.

To provide additional challenges for the three competitive codes, the third, fourth, and fifth computational experiments focused on the "largest" and "difficult" problems offered in the testbed, a 50×100 problem set. The problem size set with $s = 50$ and $d = 100$ includes eight problem types, where each type includes 15 randomly generated problems, giving a total of 120 test problem instances. For the third computational experiment, we selected a subset of 24 problems from the testbed of 120 "large" and "difficult" instances, where problems five, ten, and fifteen were chosen from each of the eight problem types. The purpose of this experiment is to investigate the relative performance of CPLEX 9.0 where no pre-processing option is included versus the case when pre-processing option is activated for both AMPL and the mixed-integer programming solver. We set the maximum CPU time limit for each problem to 19,000 CPU seconds, which is equal to the 100 times of the overall average CPU time of the parametric GIP approach for solving the entire set of 120 "large" and "difficult" problem set. Table A-2, in the appendix, shows the solution results for the third experiment. The results indicate that CPLEX 9.0 was not able to obtain optimal solution for any of the 24 problem instances within the maximum time limit. For 10 out of 24 problems, the best solutions were found by excluding the pre-processing option, while for the remaining 14 problems the best solutions were found when pre-processing option was utilized. The last two columns in Table A-2 show the CPU times required to reach the best solutions found and the total execution times. The minimum, maximum, and average CPU seconds to obtain the best solutions are 3,650.16, 18,346.50, and 13,606.75, respectively.

Given the best solutions found by CPLEX 9.0 (the better of the two obtained by including and excluding the pre-processing option), the objective of computational experiment number 4 was to compare relative performances of the three codes on the set of 24 selected "large" and "difficult" problems instances. In solving the problem subset by the TS method, we applied the run parameters utilized by Sun et al. (1998). The parametric GIP code used the same common run parameters to solve this problem set as the ones used for previous computational experiments: $\lambda_{\text{iter}} = 700$, $l_{\text{scan}} = d$, $l_{\text{size}} = 0.10\ d$, $l_{\text{max}} = 0.05\ d$, and $\text{Max}_{\text{iter}} = (\lambda_{\text{max}} - \lambda_{\text{min}} + 1)\lambda_{\text{iter}} = 7,700$. The solution results obtained by the three approaches are summarized in Table 4. These results reveal that the parametric GIP approach found 24 solutions of higher quality than CPLEX 9.0, and 23 better solutions than the ones found by the TS method. A comparison of the TS code against the CPLE 9.0 discloses that the TS approach found solutions of higher quality than CPLEX 9.0 in 18 out of the 24 instances. The average CPU time required to reach the best solution found by each particular method was 13,606.75 seconds for CPLEX 9.0, 7.59 seconds for the TS method and 115.52 seconds for the GIP method, while the average total execution CPU times for the three methods are 19,000 seconds, 35.38 seconds, and 190.67 seconds, respectively. Again, it is important to note that the solution times for the GIP method should be divided by a factor of somewhere between 4 and 12 in order to be compared appropriately to the times reported for the TS code.

Table 4. Solution results for a sample of "difficult" FCTPs.

	CPLEX 9.0			Sun et al. (1998)			Parametric GIP		
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time
N3004	**167,328**	9,606.45	19,000	167,581	7.49	33.08	**167,275**	110.80	194.51
N3009	167,423	11,029.10	19,000	**167,206**	30.79	30.79	**167,193**	17.61	110.50
N300E	**169,621**	12,898.60	19,000	169,865	6.90	28.90	**169,375**	61.71	155.39
N3104	**179,715**	16,175.40	19,000	179,828	0.00	42.36	**179,230**	135.50	211.28
N3109	**178,016**	8,235.38	19,000	178,077	0.00	38.18	**177,599**	34.44	134.07
N310E	180,551	18,251.20	19,000	**180,273**	11.24	47.55	**179,763**	88.85	190.15
N3204	202,503	12,222.30	19,000	**201,748**	8.67	42.21	**201,089**	78.87	184.59
N3209	199,672	12,406.90	19,000	**199,160**	8.49	29.43	**198,262**	161.34	215.64
N320E	**201,365**	10,755.60	19,000	201,583	6.21	38.14	**200,178**	63.29	172.56
N3304	244,496	16,512.70	19,000	**243,778**	40.03	40.03	**241,295**	119.31	208.32
N3309	241,870	17,645.30	19,000	**238,961**	11.03	34.61	**238,233**	92.41	197.42
N330E	**241,239**	17,926.40	19,000	241,727	0.00	44.41	**238,434**	153.18	197.21
N3404	324,783	14,327.20	19,000	**316,053**	3.84	44.56	**314,941**	126.28	191.29
N3409	318,978	18,346.50	19,000	**315,370**	0.00	31.81	**312,060**	130.11	200.84
N340E	319,028	17,493.10	19,000	**316,113**	4.53	32.62	**311,349**	157.64	207.81
N3504	471,897	14,154.40	19,000	**458,946**	9.87	28.63	**454,244**	125.64	205.35
N3509	469,705	13,707.50	19,000	**453,419**	4.39	27.47	**451,451**	112.08	199.47
N350E	469,890	15,471.50	19,000	**451,834**	1.76	32.40	**449,546**	116.12	210.32
N3604	761,440	13,324.30	19,000	**731,657**	13.78	28.66	**719,948**	193.04	208.90
N3609	744,103	12,634.70	19,000	**717,733**	9.56	27.83	**713,193**	154.76	208.72
N360E	747,799	14,649.90	19,000	**722,009**	3.49	30.48	**712,120**	158.4	211.16
N3704	1,314,480	12,673.20	19,000	**1,247,729**	0.00	40.40	**1,230,928**	57.01	172.40
N3709	1,285,720	3,650.16	19,000	**1,220,415**	0.00	39.13	**1,211,783**	167.97	187.12
N370E	1,291,020	12,464.30	19,000	**1,219,003**	0.00	35.33	1,220,285	156.00	201.02
Min.	167,328.00	3,650.16	19,000	167,206.00	0.00	27.47	167,193.00	17.61	110.50
Max.	1,314,480.00	18,346.50	19,000	1,247,729.00	40.03	47.55	1,230,928.00	193.04	215.64
Ave.	453,860.08	13,606.75	19,000	439,169.50	7.59	35.38	435,823.92	115.52	190.67
Std.	375847.82	3491.40	0.00	353,378.06	9.68	6.09	350,537.06	45.56	25.80

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.
**CPLEX 9.0 reports the best solutions found by *Presolve* option.

The final computational experiment includes a relative performance analysis of the two competing metaheuristics, the TS code and the parametric GIP implementation on the entire set of 120 "large" and "difficult" test problem instances. This experiment excluded CPLEX 9.0 due to its inferior performance both in solution time and solution quality shown in the

Table 5. Summary solution results for the types a through *h* "difficult" FCTPs.

Problem type	Number of test problems	Number of better sols. found	Number of worse sols. found
A	15	14	1
B	15	15	0
C	15	15	0
D	15	15	0
E	15	15	0
F	15	15	0
G	15	14	1
H	15	11	4
Total:	120	114	6

previous experiments. The run parameters for both the TS code and the parametric GIP code used in this experiment are the same as the previous experiment. Tables A-A through A-H in the appendix show detailed computational results for problem types A through H. Each table includes information about problem instance ID number, solution results provided by the competing method, the best objective function values and solution time generated by the parametric ghost image processes method, and relative objective function improvements gained by the new approach. Also, included in each table are the columns' min, max, average, and standard deviation.

Drawing on the Tables A-A through A-H, solution summary results for the 50×100 testbed problems are presented in Tables 5–7. Table 5 indicates that the parametric GIP method was successful in obtaining solutions of higher quality for 114 out of the 120 problem instances in the types A through H sets. Table 6 shows the objective function improvement or deterioration for types A through H problem sets. The table also identifies the problem types, the best objective function and percentage improvement, and average and average percentage improvement. Also, for the 6 problems for which better solutions were not found, the table includes the objective function worst and percentage deterioration, and average and average percentage deterioration. In addition, the table presents the min, max, and average values for the columns, and reveals that the best percentage improvement in the objective function value obtained by the parametric GIP approach ranges between 0.36 and 2.99%, averaging 1.55%. The minimum, maximum, and average percentage improvements in the objective function for the subset of 114 problems are 0.02, 0.13, and 0.06%, respectively. As the problem set becomes more complex and its members become harder to solve, the improvement in the objective function gained by the parametric GIP approach increases. In addition, Table 6 shows that the worst percentage deterioration in the objective function value obtained by the parametric GIP method ranges between 0.02 and 0.45%. The minimum, maximum, and average percentage deteriorations in the objectives for the subset of 29 problems are 0.02, 0.45, and 0.06%, respectively. As an overall performance indicator, the average percentage improvement in solutions obtained

Table 6. Summary solution improvement for the types a through H "Difficult" FCTPs.

	O.F.V. Improvement				O.F.V. Deterioration			
Problem type	Best improv.	Best improv. (%)	Ave. improv.	Ave. improv. (%)	Worst deterio.	Worst deterio. (%)	Ave. deterio.	Ave. deterio. (%)
A	612	0.36	36,7.93	0.02	31	0.02	31	0.02
B	959	0.54	515.80	0.02				
C	3,117	1.57	1,458.27	0.05				
D	3,674	1.54	1,828.13	0.05				
E	5,020	1.62	2,351.13	0.05				
F	8,607	1.93	3,716.67	0.06				
G	12,853	1.81	7,288.07	0.07	3,194	0.45	3,194	0.45
H	36,400	2.99	17,100.55	0.13	1,583	0.13	1,070.50	0.02
Min.	612.00	0.36	36,7.93	0.02	31.00	0.02	31.00	0.02
Max.	36,400.00	2.99	17,100.55	0.13	3,194.00	0.45	3,194.00	0.45
Ave.	8,905.25	1.55	4,328.32	0.06	1,602.67	0.08	536.94	0.06

Table 7. Summary execution CPU time for the types A through H "Difficult" FCTPs.

	Sun et al. (1998) solution CPU time (secs.)			Parametric GIP Solution CPU time (secs.)			
Problem ID	Min	Max	Ave	Min	Max	Ave	Relative speed
A	27.38	40.38	33.14	110.50	204.43	176.44	5.32
B	30.25	47.55	37.97	120.21	212.18	181.43	4.78
C	29.29	54.21	39.13	134.98	220.96	195.60	5.00
D	30.90	47.34	37.82	157.40	210.07	191.03	5.05
E	27.48	44.56	34.75	99.57	207.86	181.35	5.22
F	27.47	38.85	31.77	172.93	212.77	196.93	6.20
G	26.24	36.01	30.81	104.18	218.51	176.21	5.72
H	24.35	43.97	32.55	110.62	225.03	185.74	5.71
Min	223.36	352.87	277.95	1,010.39	1,711.81	1,484.72	4.78
Max	24.35	36.01	30.81	99.57	204.43	176.21	6.20
Ave	30.90	54.21	39.13	172.93	225.03	196.93	5.37

*CPU time on Dell, Latitude, Pentium III, Laptop.

by the parametric GIP method for the 114 problems of types A through H equals the average percentage deterioration observed in the objective function of the remaining six problems.

With regard to the execution solution CPU times, a review of Table A-A through A-H reveals that in every problem instance the TS method outperformed the parametric GIP approach. A summary of solution CPU times for the eight problem types is shown in Table 7. The table includes the minimum, maximum, and average execution CPU times for both the TS and the parametric GIP approaches. Also, the table shows minimum, maximum, and average values for each column. In addition, the table includes the relative speed of the TS versus the parametric GIP method. From this table, we can conclude that for each problem type the min, max, and average CPU times associated with the TS approach is smaller than the ones provided by the parametric GIP method. The relative speed column indicates that the TS code is faster than the parametric GIP code by 4.78 to 6.20 times, averaging a speed factor of 5.37. Accounting for the expected difference in times that magnifies the solution time for the GIP code by a factor of 4 to 12, the TS code and the GIP code may be seen to perform at about the same level of efficiency in solving the "large" and "difficult" problem instances.

5. Summary and conclusions

In the early 90s, the ghost image process (GIP) approach was proposed as a progressively staged process for solving a variety of optimization problems, based on a collection of solution principles derived from tabu search and adapted to the setting of staged

solution methods. A specific instance of this approach utilizing a progressive modification of a parameterized objective function was proposed for fixed-charge problems. The current study utilizes a version of this design as a foundation for solving fixed-charge pure and generalized network problems. Our work specifically focuses on computational testing of the fixed-charge transportation problem, which is the most widely examined class of fixed-charge problems in the literature. Our computational experiments compare our approach with CPLEX 9.0 and the approach of Sun et al. (1998) that has previously been found to be the most effective procedure for fixed-charge transportation problems across the most comprehensive problem testbed available in the literature.

The first computational experiment, which focus on representative problem instances from the six smallest and easiest-to-solve problem sets, shows that the parametric GIP approach obtains solutions whose quality matches or exceeds that of the best solutions obtained by CPLEX 9.0 and the Sun et al. (1998) method in all but two instances. Also, the solution efficiency of our parametric GIP method proves to be roughly 250 times greater than that of CPLEX 9.0, but 3 times less than that of TS method. Accounting for implementation differences that reduce the speed of the GIP code by a factor of 4 to 12 compared to the TS code (by using a general purpose solver for two-multiplier generalized networks versus a solver specialized for transportation networks, and by not exploiting re-pricing processes in the generalized network approach), the GIP code and the TS code may be considered approximately equal in efficiency in solving the "small" and "easy" problem instances.

Our second comparative experiment include a sample of 24 "large" and 'difficult" problems from the testbed. For all problem instances in this experiment, the parametric GIP approach found higher quality solutions than those obtained by CPLEX 9.0. Also, the GIP approach obtained 23 solutions of higher quality than those provided by the TS method, which in turn obtained better solutions than CPLEX 9.0 in 18 of the 24 cases. The parametric GIP code proved approximately 100 times more efficient than CPLEX 9.0, but was slower than the Sun et al. (1998) method by a factor of roughly 5. Again, this falls within the expected factor of 4 to 12, and hence after adjusting for this factor, the GIP and the TS methods may be considered to run at roughly comparable levels of efficiency in solving the subset of "large" and "difficult" problems. However, the efficiency differences compared to CPLEX 9.0 are likely to be somewhat greater than indicated since the CPLEX runs were truncated after reaching a pre-set iteration limit.

Our third and more comprehensive computational experiment focuses on the entire set of "large" and "difficult" problems (the seventh set) from the testbed, having eight problem subsets with different degrees of complexity. This experiment includes a comparison between the two metaheuristics and CPLEX 9.0 was excluded due to its inferior performance in the previous experiments. In this case, the new GIP method found solutions of higher quality for 114 out of 120 problem instances. The GIP method ran from 4.78 to 5.71 times slower than the TS method, again making the methods roughly comparable in efficiency after adjusting for the expected 4 to 12 difference factor.

The most conspicuous avenue for potential future improvement of our method lies in replacing its Phase 2 improvement process, which presently makes only very limited recourse to notions from tabu search, with a process that relies on more advanced tabu search strategies such as those adapted to the fixed charge setting in Sun et al. (1998), Gendron, Potvin and Soriano (2003), and Crainic, Gendron, and Hernu (2004). In addition, further improvement of the competing metaheuristic approaches compared in this study may be gained from more effective implementation strategies, including a coordinated post-optimization strategy and a more advanced re-pricing technique for the parametric GIP approach. Such enhancements also paves the way for treating more general model applications with greater effectiveness. Since our implementation is not specialized to transportation problems, but rather incorporates a two-multiplier generalized network solver that handles problems of considerably greater generality, enhancements to the present design will have immediate implications for solving problems from this broader domain. Well-known model transformation techniques permit the members of this domain to encompass fixed-charge and other 0–1 optimization problems that at first seem to bear no connection with networks, and thus open the door to a wide range of additional applications.

Appendix

Table A-1. CPLEX 9.0 solution results for a sample of "Easy" FCTPs.

		Best Solution Found		Solution time	
Problem size (*s, d*)	Problem ID	Presolve = 0	Presolve = 1	Best solution time	Exec. time
10 × 10	N104	40,258	40,258	0.05	0.11
	N107	42,029	42,029	0.02	0.33
15 × 15	N204	54,502	54,502	2.56	2.70
	N207	53,596	53,596	4.75	5.37
10 × 20	N304	56,366	56,366	0.15	1.00
	N307	49,742	49,742	0.50	4.61
10 × 30	N504	57,130	57,130	26.64	237.33
	N507	52,903	52,903	19.81	43.57
50 × 50	N1004	163,669	**163,599**	10,766.50	11,000
	N1007	162,327	**162,300**	10,451.20	11,000
30 × 100	N2004	**104,046**	104,061	4,851.22	11,000
	N2007	104,331	**104,147**	7,372.78	11,000
Min.		40,258	40,258	0.02	0.11
Max.		163,669	163,599	10,767	11,000
Ave.		78,408	78,386	2,791	3,691
Std.		44739.63	44714.02	4364.52	5398.27

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.
**Presolve was set for both AMPL and SOLVER.

Table A-2. CPLEX 9.0 solution results for a sample of "Difficult" FCTPs.

	Best solution found		Solution time (secs.)	
Problem ID	Presolve = 0	Presolve = 1	Best Solution time	Exec. time
N3004	**167,328**	167,416	9,606.45	19,000
N3009	**167,423**	167,530	11,029.10	19,000
N300E	169,719	**169,621**	12,898.60	19,000
N3104	**179,715**	179,757	16,175.40	19,000
N3109	178,058	**178,016**	8,235.38	19,000
N310E	**180,551**	180,858	18,251.20	19,000
N3204	202,566	**202,503**	12,222.30	19,000
N3209	**199,672**	200,174	12,406.90	19,000
N320E	201,499	**201,365**	10,755.60	19,000
N3304	245,804	**244,496**	16,512.70	19,000
N3309	242,303	**241,870**	17,645.30	19,000
N330E	242,531	**241,239**	17,926.40	19,000
N3404	**324,783**	325,003	14,327.20	19,000
N3409	319,978	**318,978**	18,346.50	19,000
N340E	320,987	**319,028**	17,493.10	19,000
N3504	478,177	**471,897**	14,154.40	19,000
N3509	**469,705**	471,769	13,707.50	19,000
N350E	**469,890**	469,969	15,471.50	19,000
N3604	763,554	**761,440**	13,324.30	19,000
N3609	**744,103**	752,846	12,634.70	19,000
N360E	**747,799**	750,687	14,649.90	19,000
N3704	1,318,810	**1,314,480**	12,673.20	19,000
N3709	1,297,060	**1,285,720**	3,650.16	19,000
N370E	1,305,320	**1,291,020**	12,464.30	19,000
Min.	167,328	167,416	3,650.16	19,000.00
Max.	1,318,810	1,314,480	18,346.50	19,000.00
Ave.	455,722	454,487	13,606.75	19,000.00
Std.	378720.4	376212.4	3491.40	0.00

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.
**Presolve was set for both AMPL and SOLVER.
***Best solution time is determined based on CPLEX 9.0 best results.

Table A-A. Solution results for the Type A "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
N3000	168,460	0.00	33.44	168,057	89.20	193.23	403
N3001	166,930	0.00	40.38	166,678	120.04	178.12	252
N3002	167,888	24.17	37.31	167,919	83.53	195.38	(31)
N3003	168,847	5.05	27.38	168,434	35.83	144.81	413
N3004	167,581	7.49	33.08	167,275	110.80	194.51	306
N3005	168,251	3.75	37.84	167,639	161.48	195.59	612
N3006	166,287	18.12	31.56	165,862	89.37	179.85	425
N3007	167,845	15.05	32.85	167,364	79.11	196.17	481
N3008	165,944	28.34	28.34	165,576	86.87	187.98	368
N3009	167,206	30.79	30.79	167,193	17.61	110.50	13
N300A	167,895	25.00	29.54	167,358	55.32	142.26	537
N300B	168,807	8.56	35.68	168,504	152.01	175.73	303
N300C	165,765	32.10	32.11	165,295	90.68	192.61	470
N300D	166,295	37.89	37.89	166,217	160.87	204.43	78
N300E	169,865	6.90	28.90	169,375	61.71	155.39	490
Min.	165,765.00	0.00	27.38	165,295.00	17.61	110.5	(31.00)
Max.	169,865.00	37.89	40.38	169,375.00	161.48	204.43	612.00
Ave.	167,591.07	16.21	33.14	167,249.73	92.96	176.44	341.33
Std.	1,185.49	12.69	3.94	1,156.16	42.70	26.56	191.39

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-B. Solution results for the Type B "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
N3100	179,672	35.79	43.89	179,019	104.74	192.27	653
N3101	178,518	15.08	34.88	177,861	79.69	190.51	657
N3102	179,021	0.00	33.77	179,007	88.12	190.77	14
N3103	179,278	15.74	31.33	179,017	178.34	212.18	261
N3104	179,828	0.00	42.36	179,230	135.50	211.28	598
N3105	178,714	12.78	42.55	178,160	58.84	147.71	554
N3106	177,304	0.00	46.67	176,546	141.36	183.65	758
N3107	178,567	18.27	30.25	177,904	193.58	206.68	663
N3108	176,540	4.05	36.27	176,266	156.38	185.68	274
N3109	178,077	0.00	38.18	177,599	34.44	134.07	478
N310A	179,432	5.89	32.80	178,703	118.83	188.21	729
N310B	180,020	16.14	34.38	179,647	31.05	120.21	373
N310C	176,106	13.78	36.55	175,850	69.59	162.26	256
N310D	178,287	14.60	38.10	177,328	139.22	205.80	959
N310E	180,273	11.24	47.55	179,763	88.85	190.15	510
Min.	176,106.00	0.00	30.25	175,850.00	31.05	120.21	14.00
Max.	180,273.00	35.79	47.55	179,763.00	193.58	212.18	959.00
Ave.	178,642.47	10.89	37.97	178,126.67	107.90	181.43	515.80
Std.	1,233.76	9.69	5.47	1,227.19	49.55	28.03	244.03

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-C. Solution results for the Type C "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3200	201,441	8.83	31.67	199,611	136.85	185.97	1,830
3201	199,720	31.54	49.61	198,843	173.35	206.12	877
3202	201,728	22.17	33.54	199,986	159.73	195.94	1,742
3203	200,648	25.44	34.54	199,338	142.91	220.96	1,310
3204	201,748	8.67	42.21	201,089	78.87	184.59	659
3205	199,576	30.98	54.21	198,764	157.08	210.42	812
3206	198,305	4.77	43.47	197,383	44.40	134.98	922
3207	200,195	18.68	39.88	198,006	115.06	208.88	2,189
3208	197,043	13.09	42.55	196,558	105.82	209.44	485
3209	199,160	8.49	29.43	198,262	161.34	215.64	898
320A	201,041	0.00	35.33	197,924	196.01	201.10	3,117
320B	202,682	29.29	29.29	201,108	81.63	187.57	1,574
320C	198,738	22.63	36.73	196,264	136.61	210.73	2,474
320D	199,738	3.55	46.42	198,158	99.29	189.05	1,580
320E	201,583	6.21	38.14	200,178	63.29	172.56	1,405
Min.	197,043.00	0.00	29.29	196,264.00	44.40	134.98	485.00
Max.	202,682.00	31.54	54.21	201,108.00	196.01	220.96	3,117.00
Ave.	200,223.07	15.62	39.13	198,764.80	123.48	195.60	1,458.27
Std.	1,531.90	10.76	7.32	1,472.82	43.87	21.60	733.48

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-D. Solution results for the Type D "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3300	240,209	31.09	34.80	239,115	180.85	210.02	1,094
3301	241,428	4.10	33.86	238,570	49.70	159.56	2,858
3302	240,555	0.00	39.57	239,876	128.86	197.97	679
3303	237,274	22.66	40.18	237,204	54.62	157.40	70
3304	243,778	40.03	40.03	241,295	119.31	208.32	2,483
3305	241,594	6.99	36.97	237,920	113.12	187.67	3,674
3306	237,461	47.34	47.34	236,061	174.36	197.88	1,400
3307	238,483	0.00	33.18	236,150	106.06	210.07	2,333
3308	236,800	16.06	33.91	234,479	64.34	159.81	2,321
3309	238,961	11.03	34.61	238,233	92.41	197.42	728
330A	242,350	11.87	38.09	242,000	112.47	187.10	350
330B	243,341	0.00	36.71	241,009	168.38	205.16	2,332
330C	237,911	0.00	30.90	235,173	131.85	201.58	2,738
330D	237,071	0.00	42.69	236,002	186.26	188.27	1,069
330E	241,727	0.00	44.41	238,434	153.18	197.21	3,293
Min.	236,800.00	0.00	30.9	234,479.00	49.70	157.4	70.00
Max.	243,778.00	47.34	47.34	242,000.00	186.26	210.07	3,674.00
Ave.	239,929.53	12.74	37.82	238,101.40	122.38	191.03	1,828.13
Std.	2,379.06	15.75	4.58	2,288.02	44.72	18.20	1,125.27

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-E. Solution results for the Type E "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3400	314,664	12.08	30.79	312,137	3.86	99.57	2,527
3401	314,568	30.12	39.20	309,548	166.67	204.21	5,020
3402	317,426	10.36	31.60	314,136	86.46	177.59	3,290
3403	306,564	0.00	27.48	305,622	137.67	207.86	942
3404	316,053	3.84	44.56	314,941	126.28	191.29	1,112
3405	314,182	0.00	34.66	311,120	171.22	189.40	3,062
3406	308,776	1.86	27.91	308,427	31.48	128.46	349
3407	305,782	24.08	41.57	305,442	76.28	186.30	340
3408	312,399	15.87	30.73	308,179	129.18	182.53	4,220
3409	315,370	0.00	31.81	312,060	130.11	200.84	3,310
340A	317,228	0.00	37.29	316,336	72.51	170.83	892
340B	316,352	5.64	36.68	314,486	108.63	185.11	1,866
340C	310,844	0.00	34.58	308,288	108.62	198.96	2,556
340D	309,496	2.59	39.80	308,479	184.79	189.42	1,017
340E	316,113	4.53	32.62	311,349	157.64	207.81	4,764
Min.	305,782.00	0.00	27.48	305,442.00	3.86	99.57	340.00
Max.	317,426.00	30.12	44.56	316,336.00	184.79	207.86	5,020.00
Ave.	313,054.47	7.40	34.75	310,703.33	112.76	181.35	2,351.13
Std.	3,859.69	9.44	5.02	3,353.41	51.44	29.87	1,570.61

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-F. Solution results for the Type F "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3500	457,658	0.00	29.88	450,795	66.58	172.93	6,863
3501	454,374	2.02	33.25	445,767	197.87	198.73	8,607
3502	451,039	0.00	38.85	449,374	72.80	183.67	1,665
3503	439,546	30.72	32.42	438,734	203.01	212.77	812
3504	458,946	9.87	28.63	454,244	125.64	205.35	4,702
3505	452,509	18.74	34.70	448,007	66.91	176.13	4,502
3506	447,534	25.64	29.24	442,903	62.25	176.31	4,631
3507	443,375	6.54	29.82	439,880	127.01	212.16	3,495
3508	450,745	0.00	33.12	447,131	181.06	186.79	3,614
3509	453,419	4.39	27.47	451,451	112.08	199.47	1,968
350A	459,302	24.64	30.49	455,810	117.07	187.09	3,492
350B	457,231	0.00	34.07	453,736	98.91	211.64	3,495
350C	451,829	9.46	31.12	449,021	118.22	211.48	2,808
350D	451,829	9.47	31.14	449,021	116.93	209.08	2,808
350E	451,834	1.76	32.40	449,546	116.12	210.32	2,288
Min.	439,546.00	0.00	27.47	438,734.00	62.25	172.93	812.00
Max.	459,302.00	30.72	38.85	455,810.00	203.01	212.77	8,607.00
Ave.	452,078.00	9.55	31.77	448,361.33	118.83	196.93	3,716.67
Std.	5,482.32	10.48	2.85	4,931.86	45.14	14.95	1,996.33

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-G. Solution results for the Type G "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3600	713,866	22.33	31.33	713,202	140.81	199.62	664
3601	722,657	2.36	34.62	709,804	94.50	178.03	12,853
3602	716,002	4.25	29.34	704,514	19.51	104.18	11,488
3603	702,926	8.07	29.05	698,859	204.77	211.93	4,067
3604	731,657	13.78	28.66	719,948	193.04	208.90	11,709
3605	710,327	0.00	36.01	701,139	148.87	185.72	9,188
3606	715,812	13.49	30.27	711,073	25.39	137.38	4,739
3607	701,699	2.52	34.61	696,387	174.97	218.51	5,312
3608	705,938	16.91	26.24	709,132	42.01	140.18	(3,194)
3609	717,733	9.56	27.83	713,193	154.76	208.72	4,540
360A	730,942	0.00	30.16	729,011	29.41	116.60	1,931
360B	727,058	0.00	31.87	718,435	132.13	174.39	8,623
360C	725,348	0.00	30.85	716,833	78.04	173.74	8,515
360D	725,348	0.00	30.85	716,833	78.15	174.06	8,515
360E	722,009	3.49	30.48	712,120	158.4	211.16	9,889
Min.	701,699.00	0.00	26.24	696,387.00	19.51	104.18	(3,194.00)
Max.	731,657.00	22.33	36.01	729,011.00	204.77	218.51	12,853.00
Ave.	717,954.80	6.45	30.81	711,365.53	111.65	176.21	6,589.27
Std.	9,654.85	7.23	2.64	8,626.68	63.30	36.46	4,525.92

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Table A-H. Solution Results for the Type H "Difficult" FCTPs.

	Sun et al. (1998)			Parametric GIP			
Problem ID	Best O.F.V.	Best time	Exec. time	Best O.F.V.	Best time	Exec. time	O.F.V. improvement
3700	1,253,875	0.00	33.09	1,217,475	178.18	188.25	36,400
3701	1,237,126	17.43	24.35	1,217,557	158.15	206.75	19,569
3702	1,230,006	20.04	26.20	1,211,000	81.35	179.78	19,006
3703	1,213,135	0.00	42.93	1,213,202	17.97	110.62	(67)
3704	1,247,729	0.00	40.40	1,230,928	57.01	172.40	16,801
3705	1,242,727	8.95	33.00	1,223,852	107.19	198.19	18,875
3706	1,240,640	0.00	30.05	1,233,656	99.53	187.55	6,984
3707	1,213,414	22.13	43.97	1,193,304	145.94	225.03	20,110
3708	1,220,714	1.47	28.59	1,222,297	178.48	194.78	**(1,583)**
3709	1,220,415	0.00	39.13	1,211,783	167.97	187.12	8,632
370A	1,255,481	3.33	26.29	1,256,831	153.14	170.28	**(1,350)**
370B	1,243,265	17.17	30.30	1,229,574	120.68	177.22	13,691
370C	1,243,686	1.45	27.29	1,229,667	176.72	191.00	14,019
370D	1,243,686	1.45	27.33	1,229,667	181.08	196.17	14,019
370E	1,219,003	0.00	35.33	1,220,285	156.00	201.02	**(1,282)**
Min.	1,213,135.00	0.00	24.35	1,193,304.00	17.97	110.62	(1,583.00)
Max.	1,255,481.00	22.13	43.97	1,256,831.00	181.08	225.03	36,400.00
Ave.	1,234,993.47	6.23	32.55	1,222,738.53	131.96	185.74	12,254.93
Std.	14,379.28	8.48	6.44	14,062.23	49.92	25.03	10,586.74

*CPU time (seconds) on Dell, Latitude, Pentium III, Laptop.

Acknowledgment

We are indebted to two referees whose insightful observations have improved the exposition of this paper. We also thank Professor Minghe Sun for providing us with the tabu search code developed for Sun et al. (1998) study and the fixed-charge transportation testbed problems.

Note

1. The basis exchange path, excluding the arc for x_j, can include up to two cycles in the GN setting. In all of these transitions, x_j starts at either 0 or U_j and x_k ends at either 0 or U_k. In the degenerate case where no flow change occurs, r_j and all G_h are 0, and both x_j and x_k remain at one of their two bounds, but exchange their nonbasic/basic status.

References

Adlakha, V. and K. Kowalski. (2003). "A Simple Heuristic for Solving Small Fixed-Charge Transportation Problems." *Omega* 31, 205–211.

Balas, E. and M.W. Padberg. (1976). "Set Partitioning: A Survey." *SIAM Review* 18, 710–760.

Balinski, M.L. (1961). "Fixed Cost Transportation Problem." *Naval Research Logistics Quarterly* 8, 41–54.

Barr, R.S., F. Glover, and D. Klingman. D. (1981). "A New Optimization Method for Large Scale Fixed Charge Transportation Problems." *Operations Research* 29, 443–463.

Bell, G.J., B.W Lamar, and C.A. Wallace. (1999). "Capacity Improvement, Penalties, and the Fixed Charge Transportation Problem." *Naval Research Logistics Quarterly* 46, 341–355.

Cabot, A.V. and S.S Erenguc. (1984). "Some Branch and Bound Procedures for Fixed Charge Transportation Problems," *Naval Research Logistics Quarterly* 31, 145–154.

Cabot, A.V. and S.S. Erenguc. (1986). "Improved Penalties for Fixed Cost Transportation Problems." *Naval Research Logistics Quarterly* 31, 856–869.

Cooper, L. (1975). "The Fixed Charge Problem-1: A New Heuristic." *Computers and Mathematics with applications* 1(1), 89–96.

Cooper, L. and C. Drebes. (1967). "An Approximation Algorithm for the Fixed Charge Problem." *Naval Research Logistics Quarterly* 14, 89–96.

Cornuejols, G., L. Fisher, and G.L. Nemhauser. (1977). "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms." *Management Science* 23, 789–810.

Crainic, T.G., A. Frangioni, and B. Gendron. (2001). "Bundle-Based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design." *Discrete Applied Mathematics* 112, 73–99.

Crainic, T.G., B. Gendron, and G. Hernu. 2004. A slope scaling/Lagrangian perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics* 10(5), 525–545.

Denzler, D.R. (1964). "An Approximate Algorithm for the Fixed Charge Problem." *Naval Research Logistics Quarterly* 16, 411–416.

Diaby, M. (1991). "Successive Linear Approximation Procedure for Generalized Fixed-Charge Transportation Problem." *Journal of Operation Research Society* 42, 991–1001.

Dongarra, J.J. (2002). "Performance of Various Computers Using Standard Linear Equations Software." Research working paper CS-89–85, University of Tennessee, Knoxville, TN 37996.

Dwyer, P.S. (1966). "Use of Completely Reduced Numbers in Solving Transportation Problems with Fixed Charge." *Naval Research Logistics Quarterly* 13(3), 289–313.

Fisk, J. and P.G. McKeown. (1979). "The Pure Fixed Charge Transportation Problem." *Naval Research Logistics Quarterly* 26, 631–641.

Gendron, B., J.Y. Potvin, and P. Sorian. (2003a). "A Tabu Search with Slope Scaling for the Multicommodity Capacitated Location Problem with Balancing Requirements." *Annals of Operations Research* 122, 193–217.

Gendron, B., J.Y. Potvin, and P. Sorian. (2003b). "A Parallel Hybrid Heuristic for the Multicommodity Capacitated Location Problem with Balancing Requirements. *Parallel Computing* 29, 592–606.

Glover, F. (1994). "Optimization by Ghost Image Processes in Neural Networks." *Computers and Operations Research* 21(8) 801–822.

Glover, F. 1996. GN2PC: An MS DOS Based Network Optimizing System.

Glover, F., M.M. Amini, and G. Kochenberger. (2003). "Discrete Optimization via Netform Representations and a New Dynamic Branch and Bound Method." Working research paper, University of Colorado-Boulder, CO.

Glover, F., D. Klingman, and N. Phillips. (1992). *Network Models in Optimization and their Applications in Practice*. New York:Wiley.

Glover, F. and M. Laguna. (1997). *Tabu Search*. Kluwer Academic Publishers. Hingham, MA.

Gottlieb, J. and C. Eckert. (2002). "A Comparison of Two Representations for the Fixed Charge Transportation Problem." In J.J. Merelo et al. (eds.), *Parallel Problem Solving From Nature—PP3NVII, Lecture Notes in Computer Science* Vol. 2439. Springer Publishing Company, pp. 77–87.

Gray, P. (1971). "Exact Solution of the Fixed Charge Transportation Problem." *Operations Research* 19, 1529–1538.

Hirsch, W.M. and G.B. Dantzig. (1968). "The Fixed Charge Problem." *Naval Research Logistics Quarterly* 15, 413–424.

Kennington, J.L. (1976). "The Fixed Charge Transportation Problem: A Computational Study with a Branch and Bound Code." *AIIE Transaction* 8, 241–247.

Kennington, J.L. and E. Unger. (1976). "A New Branch and Bound Algorithm for the Fixed Charge Transportation Problem." *Management Science* 22, 1116–1126.

Khang, D.B. and O. Fujiwara. (1991). "Approximation Solution of Capacitated Fixed-Charge Minimum Cost Network Flow Problems." *Networks* 21, 689–704.

Kim, D. and P.M. Pardalos. (1999). "Ä Solution Approach to the Fixed-Charge Network Flow Problem Using a Dynamic Slope Scaling Procedure." *Operations Research Letters* 24, 195–203.

Kim, D. and P.M. Pardalos. (2000). "Dynamic Slope Scaling and Trust Interval Techniques for Solving Concave Piecewise Linear Network Flow Problems." *Networks* 35, 216–222.

Klingman, D., A. Napier, and J. Stutz. (1974). "NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Network Problems." *Management Science* 5, 814–821.

Kuhn, H.W. and W.J Baumol. (1961). "An Approximate Algorithm for the Fixed Charge Transportation Problem." *Naval Research Logistics Quarterly* 9(1), 1–16.

Lamar, B.W. and C.A. Wallace. (1997). "Revised Modified Penalties for Fixed Charge Transportation Problems." *Management Science* 43, 1431–1436.

Luna, H.P.L., N. Ziviani, and R.M.B. Cabral. (1987). "The Telephonic Switching Center Network Problem: Formalization and Computational Experience." *Discrete and Applied Mathematics* 18, 199–210.

McKeown, P.G. (1975). A Vertex Ranking Procedure for Solving the Linear Fixed Charge Problem." *Operations Research* 23, 1183–1191.

McKeown, P.G. (1981). "A Branch-and-Bound Algorithm for Solving Fixed Charge Problems." *Naval Research Logistics Quarterly* 28, 607–617.

McKeown, P.G. and C.T. Ragsdale. (1990). "A Computational Study of Using Preprocessing and Stronger Formulations to Solve General Fixed Charge Problem." *Operations Research* 17, 9–16.

McKeown, P.G. and P. Sinha. (1980). "An Easy Solution to Special Class of Fixed Charge Problems." *Naval Research Logistics Quarterly* 2, 621–624.

Mirzain, A. (1985). "Lagrangian Relaxation for the Star-Star Concentrator Location Problem: Approximation Algorithm and Bounds." *Networks* 15, 1–20.

Murty, K.G. (1968). "Solving the Fixed Charge Problem by Ranking Extreme Points." *Operations Research* 16, 268–279.

Nozick, L. and M. Turnquist. (1998a). "Two-Echelon Inventory Allocation and Distribution Center Location Analysis." In *Proceedings of Tristan III* (Transportation Science Section of INFORMS), San Juan, Puerto Rico.

Nozick, L. and M. Turnquist. (1998b). "Integrating Inventory Impacts into a Fixed Charge Model for Locating Distribution Centers." *Transportation Research Part E 31 E* (3), 173–186.

Ortega, F. and L.A. Wolsey. (2003). "A Branch-and-Cut Algorithm for the Single-Commodity, Uncapacitated, Fixed-Charge Network Flow Problem." *Networks* 41, 143–158.

Palekar, U.S., M.K. Karwan, and S. Zionts. (1990). "A Branch and Bound Method for the Fixed Charge Transportation Problem." *Management Science* 36, 1092–1105.

Rothfarb, B., H. Frank, D.M. Rosembaun, and K. Steiglitz. (1970). "Optimal Design of Offshore Natural-Gas Pipeline Systems." *Operations Research* 18, 992–1020.

Rousseau, J.M. (1973). "A Cutting Plane Method for the Fixed Cost Problem." Doctoral dissertation Massachusetts Institute of Technology. Cambridge, MA.

Schaffer, J.E. (1989). "Use of Penalties in the Branch-and-Bound Procedure for the Fixed Charge Transportation Problem." *European Journal of Operations Research* 43, 305–312.

Shetty, U.S. (1990). "A Relaxation Decomposition Algorithm for the Fixed Charge Network Problem." *Naval Research Logistics Quarterly* 32(2), 327–340.

Steinberg, D.I. (1970). "The Fixed Charge Problem." *Naval Research Logistics Quarterly* 7(2), 217–236.

Steinberg, D.I. (1977). "Designing a Heuristic for the Fixed Charge Transportation Problem." Reprints in Mathematics and the Mathematical Sciences. Southern Illinois University at Edwardsville, Edwardsville. IL

Sun, M., J.E. Aronson. , P.G. McKeown, and D. Drinka. (1998). "A Tabu Search Heuristic Procedure for the Fixed Charge Transportation Problem." *European Journal of Operational Research* 106, 441–456.

Sun, M. and P.G. McKeown. (1993). Tabu Search Applied to the General Fixed Charge Problems." *Annals of Operations Research* 41(1–4), 405–420.

Walker, W.E. (1976). "A Heuristic Adjacent Extreme Point Algorithm for the Fixed Charge Problem." *Management Science* 22(3), 587–596.

Woodruff, D. (1995). "Ghost Image Processing for Minimum Covariance Determinant Estimators." *ORSA Journal on Computing* 7, 468–473.

Wright, D. and C. Haehling von Lanzenauer. (1989). "Solving the Fixed Charge Problem with Lagrangian Relaxation and Cost Allocation Heuristics." *European Journal of Operations Research* 42, 304–312.

Wright, D. and C. Haehling von Lanzenauer. (1991). "COLE: A New Heuristic Approach for Fixed Charge Problem Computational Results." *European Journal of Operations Research* 52, 235–246.