Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design

Ilfat Ghamlouche, Teodor Gabriel Crainic, Michel Gendreau,

To cite this article:
Ilfat Ghamlouche, Teodor Gabriel Crainic, Michel Gendreau, (2003) Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. Operations Research 51(4):655-667. http://dx.doi.org/10.1287/opre.51.4.655.16098

CYCLE-BASED NEIGHBOURHOODS FOR FIXED-CHARGE CAPACITATED MULTICOMMODITY NETWORK DESIGN

ILFAT GHAMLOUCHE

Département d'informatique et recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P.6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7, ilfat@crt.umontreal.ca

TEODOR GABRIEL CRAINIC

Département de management et technologie, Université du Québec à Montréal, and Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada, theo@crt.umontreal.ca

MICHEL GENDREAU

Département d'informatique et recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada, michelg@crt.umontreal.ca

We propose new cycle-based neighbourhood structures for metaheuristics aimed at the fixed-charge capacitated multicommodity network design formulation. The neighbourhood defines moves that explicitly take into account the impact on the total design cost of potential modifications to the flow distribution of several commodities simultaneously. Moves are identified through a shortest-pathlike network optimization procedure and proceed by redirecting flow around cycles and closing and opening design arcs accordingly. These neighbourhoods are evaluated and tested within a simple tabu search algorithm. Experimental results show that the proposed approach is quite powerful and outperforms existing methods reported in the literature.

Received January 2001; revision received November 2001; accepted April 2002.

Subject classifications: Networks/graphs, multicommodity: fixed-charge capacitated multicommodity network design. Networks/graphs, heuristics: cycle-based neighbourhoods for metaheuristics. Transportation models, network: static and dynamic applications in infrastructure and service design.

Area of review: Transportation.

1. INTRODUCTION

The fixed-charge capacitated multicommodity network design formulation (CMND) represents a generic model for a wide range of applications in planning the construction, development, improvement, and operation of transportation, logistics, telecommunication, and production systems (Balakrishnan et al. 1997, Magnanti and Wong 1986, Minoux 1986). In these applications, multiple commodities (goods, data, people, etc.) must be routed between different points of origin and destination over a network of limited capacity. Moreover, other than the routing cost proportional to the number of units of each commodity transported over a network link, a fixed cost must be paid the first time the link is used, representing its construction (opening) or improvement costs. The objective of CMND is to identify the optimal design; that is, to select the links to include in the final version of the network to minimize the total system cost, computed as the sum of the fixed and routing costs, while satisfying the demand for transportation.

Fixed-charge network design problems are difficult (they belong to the NP-hard complexity class; Magnanti and Wong 1986), and capacitated ones are even more so (Balakrishnan et al. 1997) due, among other factors, to the competition of commodities for the limited network capacity and the subsequent multiplication of equivalent-cost flow distributions for a given design, and the difficulty of representing trade-offs between the arc fixed costs and capacities. As a consequence, for now, exact methods cannot solve realistically dimensioned cases (Crainic et al. 2001, Gendron et al. 1998, Holmberg and Yuan 2000). For capacitated problems of any interesting size, only specially tailored heuristics have proved of any help. The simplest of these are drop and add procedures based on reduced cost calculations, which determine the marginal value of including or excluding an arc from the network (Powell 1986, Koskosidis et al. 1992), while the more sophisticated make use of the marginal values of paths in the network (Crainic and Rousseau 1986, Farvolden and Powell 1994, Jarvis and Mejia de Martinez 1977). These local search heuristics were applied to particular problem classes, however, and did not attempt to explore past the first local optimum. Recently, Crainic et al. (2000) proposed a tabu search metaheuristic for the path-based formulation of the problem. The method explores the space of the path-flow variables by using pivotlike moves and column generation. Even though the method produced very impressive results, its search efficiency may be limited by the fact that each move considers the impact of changing the flow of one commodity only (a pivot from one path to another). Moreover, one cannot guarantee that all paths will be considered, neither explicitly nor implicitly.

The goal of this paper is to propose a new class of neighbourhoods for metaheuristics aimed at the CMND. The neighbourhood defines moves that may explicitly take

0030-364X/03/5104-0655 $05.00
1526-5463 electronic ISSN

into account the impact on the total design cost of potential modifications to the flow distribution of several commodities simultaneously. The fundamental idea is to explore the space of the arc design variables by redirecting flow around cycles and closing and opening design arcs accordingly. Thus, compared to path-based neighbourhoods, not only are the move evaluations more comprehensive (because all commodities on a cycle are explicitly considered), but also the range of moves is broader because flow deviations are no longer restricted to paths linking origins and destinations of actual commodities. To illustrate the quality of this neighbourhood, we implement two instances in a tabu-based local search metaheuristic. Computational experiments on problems of various sizes (up to 700 arcs and 400 commodities) show that the tabu search algorithm based on these neighbourhood structures is quite powerful, outperforming existing methods in solution quality for similar computational efforts.

The contribution of this paper is twofold. First, it introduces a new class of cycle-based neighbourhood structures for metaheuristics aimed at the CMND, together with efficient procedures to identify good moves. Second, to demonstrate the quality of the cycle-based neighbourhoods, it proposes a very simple tabu search procedure that offers the current best heuristic solutions for the CMND.

This paper is organized as follows. In §2, we recall the formulation of the capacitated multicommodity network design problem. Section 3 introduces the concept of cycle-based neighbourhood structures and describes procedures to identify associated "best" moves. The tabu search algorithm is described in §4, while calibration and computational results are reported in §5. We conclude with perspectives and a number of research avenues.

2. MATHEMATICAL FORMULATION

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network with set of nodes $\mathcal{N}$ and set of directed arcs $\mathcal{A}$. Without loss of generality, we assume that all $(i, j) \in \mathcal{A}$ are design arcs. Let $\mathcal{P}$ denote the set of commodities to move using this network, where each commodity p has a single origin $o(p)$, a single destination $s(p)$, and a flow requirement of w^p units between its origin and destination nodes. The arc-based formulation of the CMND can then be written as follows:

$$\min z(x, y) = \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \tag{1}$$

subject to

$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p \quad \forall i \in \mathcal{N}, \forall\ p \in \mathcal{P}, \tag{2}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leqslant u_{ij}y_{ij} \quad \forall (i, j) \in \mathcal{A}, \tag{3}$$

$$x_{ij}^p \geqslant 0 \quad \forall (i, j) \in \mathcal{A}, \forall p \in \mathcal{P}, \tag{4}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in \mathcal{A}, \tag{5}$$

where y_{ij}, $(i, j) \in \mathcal{A}$, represent the design variables that equal 1 if arc (i, j) is selected in the final design (and 0 otherwise), x_{ij}^p stand for the flow distribution decision variables indicating the amount of flow of commodity $p \in \mathcal{P}$ on arc (i, j), and

$\mathcal{N}^+(i) =$ set of outward neighbours of node i,
$\mathcal{N}^-(i) =$ set of inward neighbours of node i,
$u_{ij} =$ capacity of arc (i, j),
$f_{ij} =$ fixed cost applied on arc (i, j),
$c_{ij}^p =$ cost of one unit flow of commodity p on arc (i, j),

and

$$d_i^p = \begin{cases} w^p & \text{if } i = o(p), \\ -w^p & \text{if } i = s(p), \\ 0 & \text{otherwise.} \end{cases}$$

The objective function (1) accounts for the total system cost, the fixed cost of arcs included in a given design plus the cost of routing the product demand, and aims to select the minimum cost design. Constraints (2) represent the network flow conservation relations, while constraints (3) state that for each arc, the total flow of all commodities cannot exceed its capacity if the arc is opened ($y_{ij} = 1$) and must be 0 if the arc is closed ($y_{ij} = 0$). Relations (4) and (5) are the usual nonnegativity and integrality constraints for decision variables.

Finally, recall that for a given design vector $\bar{y}$, the arc-based formulation of the CMND becomes a capacitated multicommodity minimum cost flow problem (CMCF):

$$\min z(x(\bar{y})) = \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}(\bar{y})} c_{ij}^p x_{ij}^p \tag{6}$$

subject to (2) plus

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leqslant u_{ij}\bar{y}_{ij} \quad \forall (i, j) \in \mathcal{A}(\bar{y}),$$

$$x_{ij}^p \geqslant 0 \quad \forall (i, j) \in \mathcal{A}(\bar{y}), \forall p \in \mathcal{P},$$

where $\mathcal{A}(\bar{y})$ stands for the set of arcs corresponding to the design $\bar{y}$.

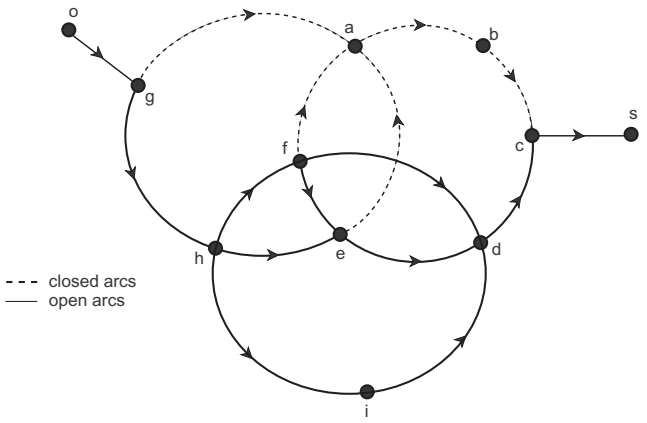
3. CYCLE-BASED NEIGHBOURHOODS

Consider the solution space of the design variables y. A solution to the CMND is then an assignment $\bar{y}$ of 0 or 1 to each design variable, plus the optimal flow of the corresponding multicommodity minimum cost flow problem $x^*(\bar{y})$. Similarly, the objective function value associated to a solution $(\bar{y}, x^*(\bar{y}))$ is the sum of the fixed cost of the open arcs in $\bar{y}$ and the objective function value of the CMCF associated to $x^*(\bar{y})$:

$$z(\bar{y}, x^*(\bar{y})) = \sum_{(i,j)\in\mathcal{A}(\bar{y})} f_{ij}\bar{y}_{ij} + z(x^*(\bar{y})). \tag{7}$$

The classical neigbourhood used to explore such a space changes the value of one, sometimes two, variables at a

Figure 1. Example of network, paths, and cycles.



time (Glover and Laguna 1997). This corresponds to adding an arc to the design, dropping an arc from the network, or swapping the values of an open and a closed arc. Such moves have performed well in related cases such as location problems (e.g., Crainic et al. 1993). They fail in the case of capacitated network design problems. The main reason is that, contrary to location problems where, in most cases, paths between origins and destinations are very short (one link, usually) and each site has a significant impact on the design and corresponding flow distribution, in design problems there are many arcs, and origin-to-destination paths are many arcs long. An arc is then only one of many and its impact is generally limited. One may often reroute traffic and obtain an almost equivalent solution. Or, as illustrated by arc (a, b) in Figure 1, it may not even be connected to the other currently open arcs. Introducing such an arc into the network has no influence whatsoever on the current solution. Therefore, one needs moves that acknowledge that commodities move on paths and that open and close several arcs simultaneously. The goal of this paper is to introduce such a neighbouring structure. But first, a few basic graph definitions:

1. A simple directed *path* from node i_0 to i_n is a sequence of closed or opened arcs $\{(i_0, i_1), (i_1, i_2), \ldots, (i_{n-1}, i_n\}$ such that the origin node of each arc is the destination node of the preceding arc in the sequence, and $i_0, \ldots, i_n$ are all distinct nodes. In Figure 1, $\{(h, f), (f, a), (a, b), (b, c), (c, s)\}$ is a path.

2. A simple *chain* is similar to a path except that arcs are not restricted to following the same direction. In Figure 1, $\{(a, b), (b, c), (c, d)\}$ represents a chain.

3. A (simple) *cycle* is a closed chain, such as $\{(a, b), (b, c), (c, d), (d, e), (e, f), (f, a)\}$ in Figure 1.

3.1. Neighbourhood Definition and Exploration

The fundamental idea behind the new neighbourhood class is that one may move from one solution to another by:

1. Identifying two points in the network together with two paths connecting these points, thus closing a cycle.

2. Deviating the total flow from one path to another such that at least one currently open arc becomes empty.

3. Closing all previously open arcs in the cycle that are empty following the flow deviation and, symmetrically, opening all previously closed arcs that now have flow.

To illustrate, paths $\{(f, e), (e, d), (d, c)\}$ and $\{(f, a), (a, b), (b, c)\}$ in the partial network depicted in Figure 1 form a cycle, and flow from the former may be deviated to the latter. Three arcs will thus be open, (f, a), (a, b), and (b, c) and, provided sufficient flow may be deviated, at least one previously open arc will be closed. The general neighbourhood structure we propose for the CMND may then be written as:

$$\mathscr{V}(\bar{y}) = \{y\text{: obtained from } \bar{y} \text{ by complementing the status of a number of arcs following the deviation of flow in a given cycle in } \mathscr{A}(\bar{y})\}.$$

Such neighbourhoods are huge and their explicit and exhaustive exploration is not practical in most situations. Moreover, the complete evaluation of any design modification involves the resolution of a capacitated multicommodity network flow problem, which rapidly becomes extremely computationally intensive. Thus, to select the best move out of a given solution, one cannot simply identify all neighbours explicitly, evaluate them, and retain the best one. A more efficient procedure must be implemented that (1) avoids the complete evaluation of every examined move, and (2) generates a limited number of cycles that include the "good" moves. One such procedure is presented in the following and it is implemented and tested in §4.

Note that not all cycles are of equal interest. We seek, in particular, moves that modify the status of several arcs and that lead to a significant modification of the flow distribution. Therefore, moves that close at least one arc and open new paths for a group of commodities appear attractive.

To close an arc (i, j), one must be able to deviate all its flow. Let the *residual capacity* of a cycle denote the maximum flow one can deviate around the cycle. The residual capacity of any cycle that includes (i, j) must then be at least as large as $\sum_{p \in \mathscr{P}} x_{ij}^p$, the total flow on arc (i, j). Consequently, cycles of interest to us display a residual capacity equal to one of the values in Γ defined as the set of the total (strictly positive) volumes on the open arcs of the corresponding network:

$$\Gamma(\bar{y}) = \left\{ \sum_{p \in \mathscr{P}} x_{ij}^p > 0 \ : \ (i, j) \in \mathscr{A}(\bar{y}) \right\}. \tag{8}$$

Cycles have thus to be identified on *residual networks* of $\mathscr{A}(\bar{y})$ defined according to $\Gamma(\bar{y})$, and the one leading to the network modification that yields the largest improvement (smallest deterioration, eventually) in Equation (7) corresponds to the best move in the neighbourhood of $\bar{y}$. But, again, finding all such cycles and explicitly evaluating the corresponding moves is impractical and some heuristic must be used. We therefore propose a network optimization procedure to implicitly evaluate a large number of cycles according to criteria that approximate the evaluation criterion defined by Equation (7). We thus progressively build

a set of good candidate neighbours (cycles) among which the best move is then selected.

Let $\mathscr{C} \subseteq \mathscr{A}$ denote a set of *candidate* links, where each arc $(i, j) \in \mathscr{C}$ will be considered as the starting point of cycles. Let $\mathscr{C}(\gamma) \subseteq \mathscr{C}$ include all arcs $(i, j) \in \mathscr{C}$ that can support a movement of γ units of flow. A closed arc may belong to $\mathscr{C}(\gamma)$ only if its capacity is at least γ. For an open arc, either its flow or its residual capacity must be at least γ units. The heuristic procedure we propose to explore the neighbourhood $\mathscr{V}(\bar{y})$ of a given solution $\bar{y}$ may be synthesized as follows:

- Build sets $\mathscr{C}(\bar{y})$ and $\Gamma(\bar{y})$;
- For each $\gamma \in \Gamma(\bar{y})$
  - — Build the γ-residual network as indicated in §3.2;
  - — For each arc $(i, j) \in \mathscr{C}(\gamma)$, find the lowest-cost cycle by using the network optimization procedure of §3.3;
- Select and implement the best move;
- Evaluate the new solution.

This procedure is embedded in the tabu-based local search procedure described in §4. First, however, we define γ-residual networks and describe the network optimization procedure used to find low-cost cycles in residual networks.

3.2. The γ-Residual Network

The objective is to build a network such that, given a flow value γ, the network optimization procedure of §3.3 may: (1) identify low-cost cycles that support the deviation of at least γ units of flow, and (2) explicitly consider the impact of a potential closure or opening of an arc due to the deviation of flow.

To build the γ-*residual network* corresponding to a flow variation of γ units, replace each arc (i, j) of the original network by at most two arcs $(i, j)^+$ and $(j, i)^-$.

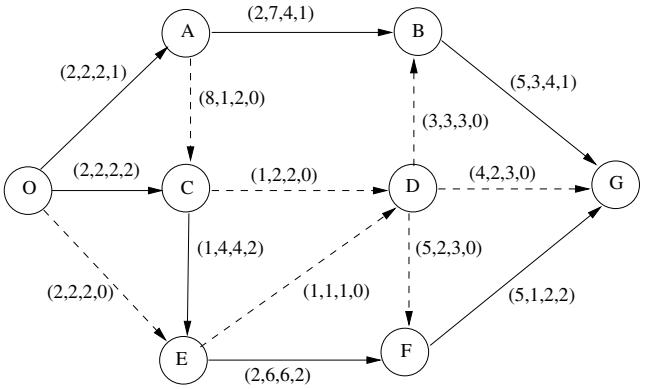
Arc $(i, j)^+$ is included only if an additional amount of γ units of flow may pass on arc (i, j); that is, if its residual capacity $u_{ij} - \sum_{p \in \mathscr{P}} x_{ij}^p$ is greater than or equal to γ. The cost c_{ij}^+ associated to $(i, j)^+$ approximates the cost of routing γ units of additional flow on arc (i, j). It equals the average commodity routing costs on the arc, plus the fixed cost if it is currently closed:

$$c_{ij}^+ = \frac{\sum_{p \in \mathscr{P}} c_{ij}^p}{|P|}\gamma + f_{ij}\left(1 - \left(\left\lceil \min\left(1, \sum_{p \in \mathscr{P}} x_{ij}^p\right)\right\rceil\right)\right).$$

Symmetrically, arc $(j, i)^-$ is not included in the γ-residual network if the total flow on arc (i, j), $\sum_{p \in \mathscr{P}} x_{ij}^p$, is less than γ. Otherwise, we associate to $(j, i)^-$ the cost c_{ji}^- that approximates the value of a reduction of γ units of the total flow (all commodities) currently using arc (i, j). This approximation is computed as the weighted average of the routing costs of the commodities currently using arc (i, j). The fixed cost of (i, j) is then subtracted if the reduction of γ units of flow leaves the arc empty:

$$c_{ji}^- = \begin{cases} -\dfrac{\sum_{p \in \mathscr{P}} c_{ij}^p x_{ij}^p}{\sum_{p \in \mathscr{P}} x_{ij}^p}\gamma - f_{ij} & \text{if } \sum_{p \in \mathscr{P}} x_{ij}^p = \gamma, \\ -\dfrac{\sum_{p \in \mathscr{P}} c_{ij}^p x_{ij}^p}{\sum_{p \in \mathscr{P}} x_{ij}^p}\gamma & \text{if } \sum_{p \in \mathscr{P}} x_{ij}^p > \gamma. \end{cases}$$

Figure 2. Network example.



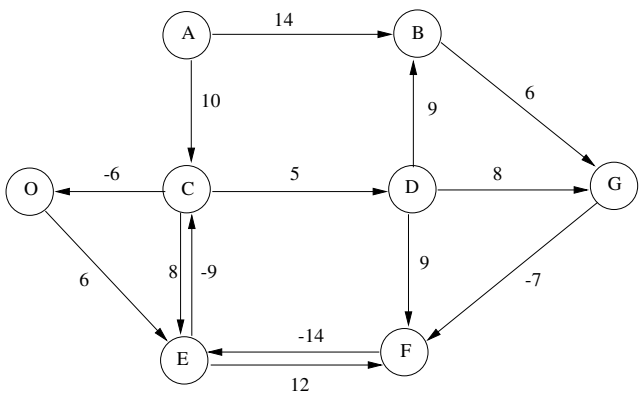
To illustrate this procedure, consider the network of Figure 2. In this example, arcs are labeled by their fixed cost, routing cost (only one commodity is considered), capacity, and total flow, respectively. We build a two-residual network and look for the lowest-cost cycle that passes through arc (C, D).

Two directed cycles pass through arc (C, D) in the two-residual network of Figure 3: $(D, F), (F, E), (E, C), (C, D)$ and $(D, G), (G, F), (F, E), (E, C), (C, D)$. The latter has the lowest cost with value -17. This cost represents the net change in the objective function of the design formulation when moving two units of flow from path $(C, E), (E, F), (F, G)$ to path $(C, D), (D, G)$, opening arcs (C, D) and (D, G), and closing arcs $(C, E), (E, F)$, and (F, G) in the original network. This neighbour is thus reached by a complex move that involves opening two arcs $((C, D), (D, G))$ and closing three others $((C, E), (E, F), (F, G))$.

3.3. Heuristic to Identify Low-Cost Cycles

The enumeration of all cycles passing through a given arc in a γ-residual network is expensive. Because one is interested in the lowest-cost cycle only, one would like to find this particular cycle without enumerating all the

Figure 3. Associated two-residual network.



others. A network optimization procedure is proposed for this purpose.

The main idea is to find for a given arc (i, j) the shortest path from j to i to complete the cycle. Yet, because the γ-residual network may contain negative cost directed cycles, finding a shortest path is NP-hard (Ahuja et al. 1993). Consequently, we do not attempt to solve the problem exactly. Rather, we develop a heuristic based on a modification of the shortest-path label-correcting algorithm that avoids getting trapped in negative directed cycles and allows us to find low-cost cycles.

Label-correcting (Ahuja et al. 1993) is one of the algorithms designed to find shortest paths between a source node and all other nodes in a network that contains no negative directed cycles. The algorithm starts by labelling each node j of the network with $d(j)$, the current distance (cumulated cost) in the shortest-path tree from the source to the node. Nodes are also labelled with their respective predecessors in the current shortest paths, pred(j), to be able to retrace the shortest paths. The algorithm then proceeds by decreasing, at every iteration, the distance label of a node j to satisfy the shortest-path optimality conditions

$$d(j) \leqslant d(i) + c_{ij} \quad \forall (i, j) \in \mathcal{A},$$

where c_{ij} denotes the cost of arc (i, j).

Label-correcting algorithms operate by maintaining a LIST of nodes i that may be the origin of links for which the optimality condition is not yet satisfied. At each iteration, one node i is removed from LIST and the nodes $j \in \mathcal{N}^+(i)$ are scanned. If $d(j) > d(i) + c_{ij}$, $d(j)$ is set to $d(i) + c_{ij}$ and j is added to LIST (if not already included). The algorithm terminates when LIST is empty.

In the presence of negative-cost directed cycles, the algorithm gets trapped and keeps cycling. To avoid this, we modify the label-correcting algorithm to explicitly verify if a scanned node j already belongs to the current shortest path from the source node to its predecessor node i, in which case the label of node j is not modified (we forbid node j to be labeled from node i). The label-correcting-based heuristic then becomes

$d(s) = 0$, pred$(s) = 0$, $d(j) = \infty$ for each $j \neq s$;
LIST $= \{s\}$
While LIST $\neq \phi$
  Remove an element i from LIST
  For each node j in its forward star $\mathcal{N}^+(i)$
    If $(d(j) > d(i) + c_{ij})$ and (j is not in the path from s to i)
    - $d(j) = d(i) + c_{ij}$;
    - pred$(j) = i$;
    - If $j \notin$ LIST, then add j to LIST.

The path generated by this heuristic is not necessarily the shortest path. Computational results show, however, that the heuristic produces very good cycles.

4. TABU SEARCH WITH CYCLE-BASED NEIGHBOURHOODS

Tabu search is an iterative metaheuristic (Glover 1986, 1989, 1990) that has consistently been successful in addressing hard optimization problems (Glover and Laguna 1997), including the CMND (Crainic et al. 2000). To evaluate the concepts introduced in the previous section, we developed a simple tabu-search-based local search procedure that integrates two versions of the cycle-based neighbourhood: One that considers the flow of all commodities when determining cycles and a second one that refines the search by implementing moves resulting from the deviation of the flow of only one commodity at a time. First, we present the basic tabu search framework we use. Then, we examine how the procedure deals with infeasible solutions and how the search is intensified following a move to a "good" solution.

4.1. The Tabu Search Procedure

Following an initialization phase, the tabu search procedure explores the y solution space using a simple local search framework based on the neighbourhood defined in §3. At each iteration, the best non-tabu move is determined and implemented regardless of whether it improves the overall solution or not. A short-term tabu memory is used to record characteristics of visited solutions to avoid cycling. When a particularly good solution is encountered, the search is intensified using a particular implementation of cycle-based neighbourhoods that considers the flow distribution of one commodity only. A solution is considered particularly good when it either improves the objective value of the best known solution or is within a predefined percentage of this value. The method terminates whenever a predefined stopping criterion (number of iterations, CPU time, etc.) is met.

To select the "best" move in the neighbourhood of a given solution $\bar{y}$, the procedure first determines the set of the flow deviation values $\Gamma(\bar{y})$, as defined by (8), and the set of candidate arcs $\mathcal{C}$. In the current implementation, $\mathcal{C}$ is restricted to a random subset of closed arcs. Then, following the framework detailed in §3.1, a γ-residual network is built for each value $\gamma \in \Gamma(\bar{y})$ (§3.2), and a low-cost cycle is found for each arc $(i, j) \in \mathcal{C}(\gamma)$ on the corresponding γ-residual network (§3.3). The lowest-overall cycle, for all $(i, j) \in \mathcal{C}(\gamma)$ and $\gamma \in \Gamma(\bar{y})$, is then considered as the "best" *local search move* from the current solution, and arcs are open and closed accordingly. An exact evaluation of the new design is then performed by solving exactly the associated capacitated multicommodity network formulation. This evaluation may result in some open arcs carrying no flow. One may then *trim*—close—these arcs, further reducing the associated solution value. Notice, however, that the trim procedure modifies the basic move definition as well as the search trajectory (§5.1). An outline of the tabu search procedure follows.

Initialization. Generate an initial feasible solution and set *BestSolution* and *CurrentSolution* to its objective value.

Main Local Search Loop

While a stopping criterion is not met

- Determine sets $\mathscr{C}$, $\Gamma(\bar{y})$, and $\mathscr{C}(\gamma)$ for the current solution $\bar{y}$; Determine the best cycle over all $(i, j) \in \mathscr{C}(\gamma)$, $\gamma \in \Gamma(\bar{y})$;
- Move to the new solution by opening and closing the appropriate arcs;
  Determine the solution value of the new solution by solving exactly the associated capacitated multicommodity network flow problem;
- Assign a tabu status to each complemented arc;
- Trim;
- If the solution is infeasible, perform a *restoration* phase;
- Perform an *Intensification* phase if current solution is "good":

$$\frac{CurrentSolution - BestSolution}{BestSolution} \leqslant IntensGap;$$

- If $CurrentSolution < BestSolution$, update *BestSolution*.

Short-term tabu memory is used to prevent the search from cycling. Thus, the arcs that are opened or closed receive a tabu status that forbids the reversal of the move for a given number of iterations. The tabu memory is updated following local search and intensification moves.

An initial feasible solution is produced by opening all arcs and solving the corresponding flow problem. All arcs with flow are then considered open, while all unused arcs are closed. An intensification phase is then performed until negative cycles are no longer detected for any commodity.

4.2. Restoration from Infeasible Moves

The solution produced by a local search move might be infeasible. This follows from the fact that commodities are aggregated when local search moves are determined on γ-residual networks. Consequently, in the new network configuration that follows the closing and opening of arcs in the original network, commodities might follow different paths than the expected ones (i.e., those in the local search cycle-move).

To illustrate, consider the partial network displayed in Figure 4 where three commodities p_1, p_2, and p_3 share the network. Let (x, y, z, t) denote, respectively, the fixed cost, routing cost, capacity, and flow on each arc and suppose that $(A, D, 2)$, $(E, G, 2)$, and $(B, F, 2)$ represent the origin, destination, and demand of commodity p_1, p_2, and p_3, respectively. The current solution routes two units of flow of commodity p_1 on path $\{(A, B), (B, C), (C, D)\}$, two units of flow of commodity p_2 on path $(E, C), (C, F), (F, G)$, and two units of flow of commodity p_3 on path $(B, C), (C, F)$. The objective function value associated to the current solution equals 1,021.

In this example, $\Gamma = \{2, 4\}$. Assume all closed arcs are in $\mathscr{C}$ and an empty tabu list. Then, the lowest-cost cycle in the four-residual network of Figure 5 shows an improvement of -989. In the two-residual network (not shown), the lowest-cost cycle yields a deterioration of the objective function. Consequently, the procedure would retain the cycle $\{(B, H), (H, I), (I, J), (J, F), (F, C), (C, B)\}$ and would move to the best neighbour by opening arcs (B, H), (H, I), (I, J), and (J, F) and closing the arcs without flow (B, C) and (C, F). The resulting network is infeasible since the demands of commodities p_1 and p_2 may no longer be satisfied.

To detect infeasible solutions, artificial arcs between the origin and destination nodes of each commodity may be added to the network. These arcs receive capacities equal to the demand of the associated commodities and high routing costs to ensure that an artificial arc will bear flow only if the solution is infeasible. Then, when the solution produced by a local search move is infeasible, a restoration phase is undertaken to reach a solution in the feasible domain. This phase is similar to the neighbourhood local search, except that the set $\mathscr{C}$ is made exclusively of artificial arcs with a positive flow, while the set Γ is restricted to the flow of commodities routed on these artificial arcs.

Figure 4. Example of infeasible move: Original network.

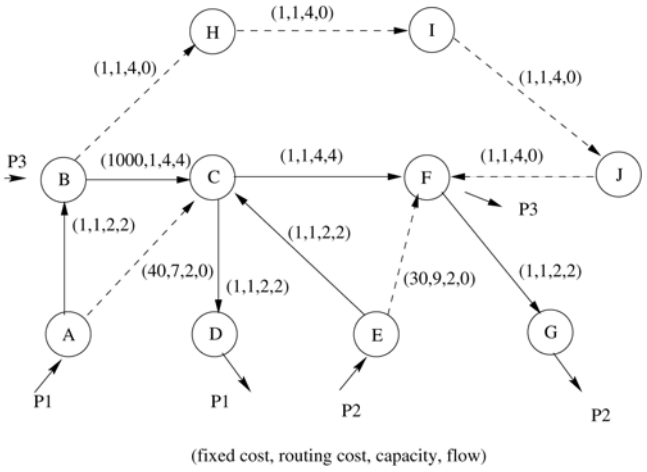
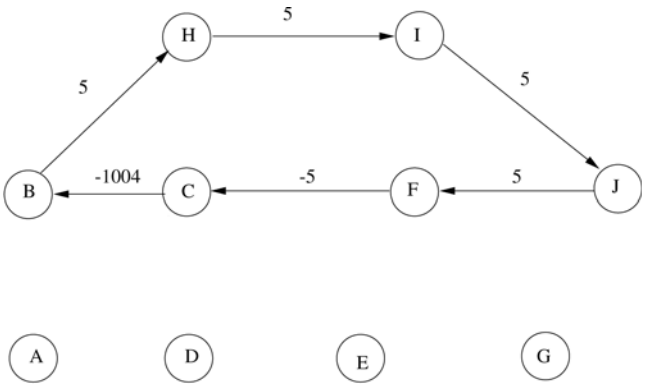


Figure 5. Example of infeasible move: Residual network.



4.3. Intensification

The intensification phase is called each time a local search move yields a "good" solution; that is, a solution that improves the best overall solution or is close to it. This phase searches to improve the solution further by iteratively modifying the flow distribution of one commodity at a time.

The neighbourhood definition of §3 is adapted for the intensification phase: Moves are detected by letting the flow of one commodity move around negative directed cycles while the flow of the other commodities is kept fixed. The set Γ is instantiated for each commodity (denoted Γ^p) to contain the flow of the particular commodity that exists on each arc of the network. Moreover, only improving moves are accepted during this phase to ensure that the chosen neighbour will always yield a feasible solution better than the current one. An outline of the intensification phase follows.

Intensification Phase

Repeat until no negative cycle is detected in any γ-residual network.

For each commodity $p \in \mathcal{P}$ and flow value $\gamma \in \Gamma^p$

- Build the γ-residual network
- Find a low-cost cycle for each arc $(i, j) \in \mathcal{C}(\gamma)$, $\gamma \in \Gamma^p$
- Select the best overall cycle; Denote by γ_{best} the associated volume of deviated flow;
- If the selected cycle is improving (negative cost)

  1. Update the solution:
     - — Modify the flow around the selected cycle by a quantity γ_{best};
     - — Open/close appropriate arcs;
     - — Update the cost of the current solution by adding the cost of the cycle (without solving the CMCF).
  2. Assign a tabu status to each complemented arc.

Solve exactly the CMCF associated to the current design.

5. COMPUTATIONAL RESULTS

Experiments have been performed to evaluate the performance of the simple tabu search metaheuristic and, more generally, of the cycle-based neighbourhood concept we propose. To ensure meaningful comparisons, we experiment on the same two sets of problem instances used by Crainic et al. (2000).

The computer code is written in C++. The exact evaluation of the capacitated multicommodity network flow problems is performed using the linear programming solver of CPLEX 6.5 (1999). Unless indicated otherwise, tests were conducted on a Sun Ultra-60/2300 workstation with 2 Gigabyte of RAM, operating under Solaris 2.6. Computing times are reported in seconds.

5.1. Calibration

A calibration phase was conducted to determine appropriate values of key parameters of the tabu search metaheuristic:

- The size of the neighbourhood set explored at each iteration. This size depends on the dimension of the candidate set $\mathcal{C}$ defined in the current implementation as a percentage of closed arcs randomly selected. Two values, 50% and 70% of the closed arcs, have been tested.
- The length of the tabu tenure of arcs opened and closed following a local search or an intensification move. Four values—1, 2, 3, and 5—were considered initially. The 1 and 5 values were rapidly dropped, however, because cycling was observed for a tabu tenure length of 1, while for a value equal to 5 the quality of solutions started to decrease.
- The threshold used to determine a "good" solution: Values of *IntensGap* equal to 7%, 9%, and 11% of relative improvement were tested.

Calibration experiments were conducted on the same problem instances used by Crainic et al. (2000) to calibrate their path-based tabu search metaheuristic. The 10 problems cover network sizes from 100 to 700 design arcs and from 10 to 400 commodities. They also display relatively high fixed costs compared to routing costs and are tightly capacitated. Moreover, since a random number generator controls the selection of arcs in $\mathcal{C}$, each run was repeated three times.

Each parameter combination was ranked for each problem instance according to the average gap (over three runs) relative to the best known solution (that of the branch-and-bound procedure of CPLEX 6.5, when available, or that obtained by Crainic et al. 2000, otherwise). A score of $10, 9, 8, \ldots, 2, 1$ is assigned to each of the first 10 places, respectively. The performance of each parameter setting is then aggregated over all 360 runs (10 problems tested 3 times for 12 parameter settings): scores are summed up, while average gaps and CPU solution times are averaged. Table 1 displays these aggregated results for each parameter combination. The first column holds the parameter setting, the second and third columns present the global average gaps and CPU times, respectively, while the last column displays the total score.

The results in Table 1 display two parameter settings that achieve the same highest aggregated score. The final choice was then made on the other performance measures and the parameter combination $\mathcal{C} = 50\%$, tabu tenure = 2, *IntensGap* = 9% was selected because it offers the lowest global average gap and CPU time. Two additional comments are prompted by the results displayed in Table 1. First, that global averages may hide important variations and may be misleading. Thus, the lowest average gap appears for the fourth ranking parameter setting (70%, 2 and 7%). Second, one notices that more than one parameter setting leads to very good performances, which is an indication of the robustness of the search.

To complete the calibration phase, we examined the impact of the intensification phase and the trim procedure

Table 1. Parameter setting performances.

Parameter Setting	Average Gap	Average CPU	Score
50%, 2,0.09	1.93%	10814.34	68
70%, 2,0.11	2.07%	13068.72	68
70%, 2,0.09	2.09%	13011.97	63
70%, 2,0.07	1.84%	13168.14	62
70%, 3,0.07	1.96%	12427.75	59
50%, 3,0.07	2.27%	10984.97	46
50%, 2,0.07	2.30%	11480.44	45
70%, 3,0.09	2.62%	12826.49	40
70%, 3,0.11	2.95%	13528.20	35
50%, 3,0.11	2.83%	11530.40	26
50%, 2,0.11	2.72%	10929.73	24
50%, 3,0.09	2.81%	11172.91	14

on the solution quality. We started the investigation by running five problems, using all four possible combinations of including or not the two procedures. Because the results were similar on all five problems, we decided not to continue the investigation. Figure 6 illustrates the behaviour of the four versions of the tabu search procedure on two problem instances, 20, 300, 200, F, L and 20, 300, 40, F, T (identified by the number of nodes, design arcs, commodities, a letter indicating high fixed costs, and rather loose and tight capacities, respectively). This behaviour is typical of the general method performance and thus, to not overload the paper, the other graphs are not shown. The results clearly indicate that including the intensification and trim procedures is beneficial for the quality of the search. Consequently, all subsequent experiments have included the two procedures.

5.2. Result Analysis

To evaluate the performance of the tabu search algorithm proposed in this paper, we compare its output to the optimal solution obtained using the branch-and-bound algorithm of CPLEX 6.5 (1999), as well as to the results of the TABU-PATH procedure presented by Crainic et al. (2000), which was found to be superior to both classical greedy descent and resource decomposition (Gendron and Crainic 1996).

Figure 6. Calibration for intensification and trim.

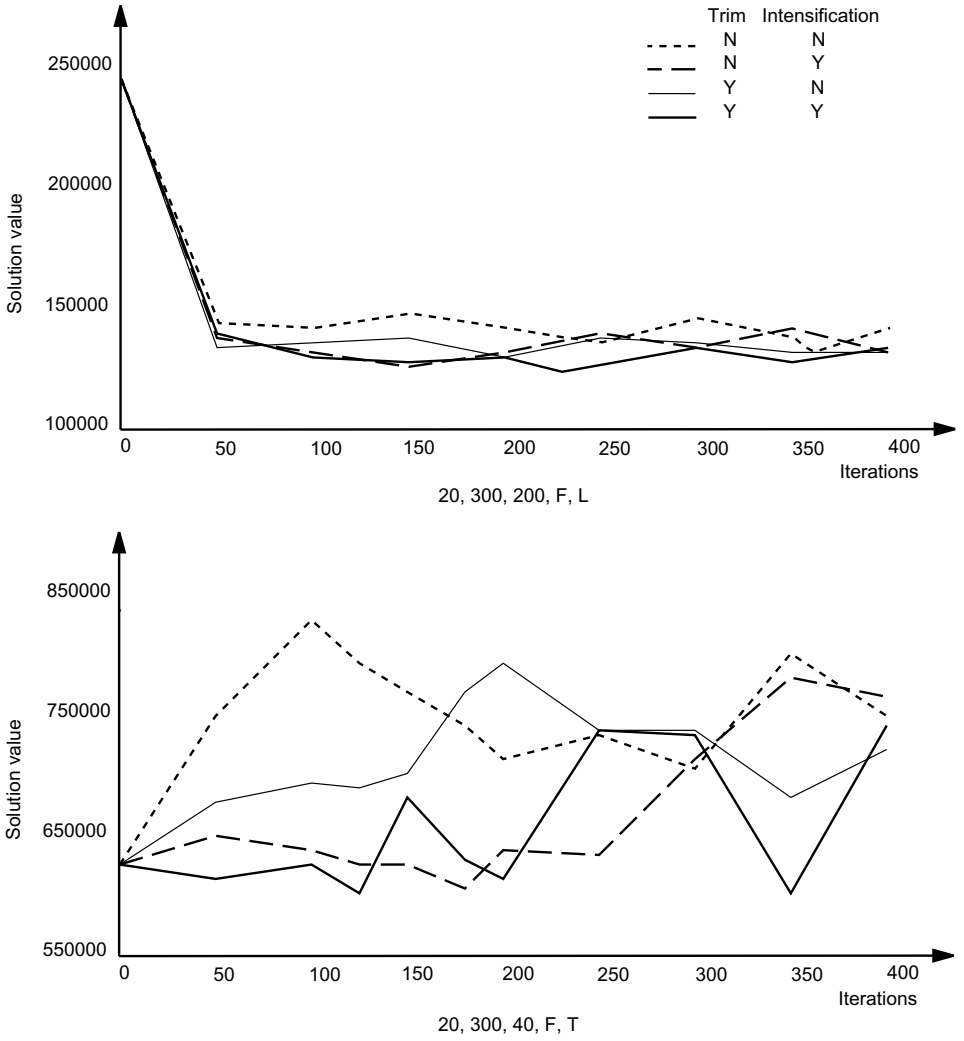


Table 2. Computational results, C problems.

Problem	Opt.	Tabu-Path	Gap	Tabu-Cycle	Gap	Tc (400)	Gap
25, 100, 10, V, L	14,712	14,712	0%	14,712	0%	14,712	0%
	(0.36)	(5.60)		(5.60)		(48.88)	
25, 100, 10, F, L	14,941	15,889	6.34%	14,941	0%	14,941	0%
	(53.64)	(8.37)		(8.37)		(53.77)	
25, 100, 10, F, T	49,899	51,654	3.52%	50,767	1.74%	49,899	0%
	(40.58)	(17.10)		(17.10)		(51.21)	
25, 100, 30, V, T	365,272	365,272	0%	365,385	0.03%	365,385	0.03%
	(16.62)	(16.57)		(16.57)		(223.67)	
25, 100, 30, F, L	37,055	38,804	4.72%	38,679	4.38%	37,583	1.42%
	(1,727.96)	(33.01)		(33.01)		(215.38)	
25, 100, 30, F, T	85,530	86,445	1.07%	86,296	0.90%	86,296	0.90%
	(534.18)	(71.84)		(71.84)		(224.64)	
20, 230, 40, V, L	423,848	425,046	0.28%	425,091	0.29%	424,778	0.22%
	(10.43)	(71.29)		(71.29)		(370.33)	
20, 230, 40, V, T	371,475	371,816	0.09%	372,583	0.30%	371,893	0.11%
	(52.37)	(90.28)		(90.28)		(435.61)	
20, 230, 40, F, T	643,036	644,172	0.18%	646,786	0.58%	645,812	0.43%
	(671.76)	(121.79)		(121.79)		(423.32)	
20, 300, 40, V, L	429,398	429,912	0.12%	429,837	0.10%	429,535	0.03%
	(3.76)	(71.05)		(71.05)		(611.5)	
20, 300, 40, F, L	586,077	589,190	0.53%	593,323	1.24%	593,322	1.24%
	(145.55)	(113.44)		(113.44)		(581.94)	
20, 300, 40, V, T	464,509	464,509	0%	466,525	0.43%	464,724	0.05%
	(83.88)	(145.33)		(145.33)		(589.64)	
20, 300, 40, F, T	604,198	606,364	0.36%	609,285	0.84%	607,100	0.48%
	(59.88)	(123.42)		(123.42)		(560.38)	
20, 230, 200, V, L	94,386	122,592	29.88%	103,930	10.11%	98,995	4.88%
	(t)	(504.50)		(504.50)		(2,663.24)	
20, 230, 200, F, L	141,737.4	188,590	33.06%	154,404	8.94%	146,535	3.38%
	(t)	(491.63)		(491.63)		(2,718.31)	
20, 230, 200, V, T	97,914	118,057	20.57%	108,274	10.58%	104,752	6.98%
	(t)	(548.36)		(548.36)		(2,565.71)	
20, 230, 200, F, T	137,271	182,829	33.19%	153,455	11.79%	147,385	7.37%
	(t)	(889.69)		(889.69)		(3,120.14)	
20, 300, 200, V, L	74,972.4	88,398	17.91%	81,628	8.88%	80,819	7.80%
	(t)	(982.21)		(982.21)		(4,086.83)	
20, 300, 200, F, L	117,306	151,317	28.99%	129,600	10.48%	123,347	5.15%
	(t)	(1,316.75)		(1,316.75)		(4,367.88)	
20, 300, 200, V, T	74,991	82,724	10.31%	80,510	7.36%	79,619	6.17%
	(t)	(938.29)		(938.29)		(3,807.93)	
20, 300, 200, F, T	108,252	135,593	25.26%	122,547	13.21%	114,484	5.76%
	(t)	(1,065.88)		(1,065.88)		(4,657.54)	

The same two data sets used by Crainic et al. (2000) are also used in this paper. Detailed results for the first set, denoted **C**, are presented in this section. Results for the second set, denoted **R**, are detailed in Ghamlouche et al. (2001a) and are summarized in the following. Problems in both sets are general transshipment networks with no parallel arcs. Each commodity corresponds to a single origin-destination pair. On each arc, routing costs are the same for all commodities. Problem instances have been generated to offer for each network size a variety of fixed cost to routing cost ratios and capacity to demand ratios. A detailed description of problem instances is given in Crainic et al. (2001; see also Gendron and Crainic 1994, 1996). The problem generators as well as the problem instances can be obtained from the authors.

Tables 2 and 3 display results for the first set, denoted **C**. Problems are identified in the first column by the number of nodes, arcs, and commodities, as well as two letters summarizing the fixed cost and capacity information: A relatively high or low fixed cost relative to the routing cost is signaled by the letter **F** or **V**, respectively, while letters **T** and **L** indicate, respectively, if the problem is tightly or somewhat loosely capacitated compared to the total demand. The OPT column corresponds to the solution of the the branch-and-bound algorithm solved using CPLEX 6.5 (1999) on one 400 MHz processor of a 64-CPU Sun Enterprise 10000 with 64 Gigabyte of RAM, operating under Solaris 2.7. A limit of 10 hours was imposed. An **X** indicates that the procedure has failed to produce a feasible solution within this time limit, while a **t** indicates that the procedure stopped due to a time limit condition.

Column TABU-PATH holds the results found by the path-based tabu search of Crainic et al. (2000) on the same workstations. For a valid comparison between the two

Table 3. Computational results, C problems.

Problem	Opt.	Tabu-Path	Gap	Tabu-Cycle	Gap	Tc (400)	Gap
100, 400, 10, V, L	28,423	28,485	0.22%	28,708	1.00%	28,677	0.89%
	(84.81)	(32.66)		(32.66)		(336.34)	
100, 400, 10, F, L	24,436	24,912	1.95%	24,576	0.57%	23,949	−1.99%
	(t)	(33.00)		(33.00)		(306.79)	
100, 400, 10, F, T	66,364	71,128	7.18%	68,004	2.47%	67,014	0.98%
	(t)	(81.23)		(81.23)		(626.46)	
100, 400, 30, V, T	385,544	385,185	−0.09%	385,508	−0.01%	385,508	−0.01%
	(t)	(277.50)		(277.50)		(1,975.34)	
100, 400, 30, F, L	50,496	58,773	16.39%	55,401	9.71%	51,552	2.09%
	(t)	(100.16)		(100.16)		(1,300.56)	
100, 400, 30, F, T	141,278	149,282	5.67%	146,455	3.66%	145,144	2.74%
	(t)	(215.71)		(215.71)		(1869.98)	
30, 520, 100, V, L	53,966	56,426	4.56%	55,358	2.58%	54,958	1.84%
	(t)	(995.64)		(995.64)		(3355.96)	
30, 520, 100, F, L	95,294	104,117	9.26%	103,747	8.87%	99,586	4.50%
	(t)	(939.24)		(939.24)		(4,032.35)	
30, 520, 100, V, T	52,085	53,288	2.31%	52,985	1.73%	52,985	1.73%
	(t)	(1,218.52)		(1,218.52)		(3,481.08)	
30, 520, 100, F, T	98,357	107,894	9.70%	106,877	8.66%	105,523	7.29%
	(t)	(670.29)		(670.29)		(3927.35)	
30, 700, 100, V, L	47,603	48,984	2.90%	48,824	2.56%	48,398	1.67%
	(1,736.05)	(1,265.11)		(1,265.11)		(4,396.42)	
30, 700, 100, F, L	60,525	65,356	7.98%	63,302	4.59%	62,471	3.22%
	(t)	(1,479.59)		(1,479.59)		(4,755.01)	
30, 700, 100, V, T	45,944.5	47,083	2.48%	47,025	2.35%	47,025	2.35%
	(t)	(2,426.02)		(2,426.02)		(4,560.11)	
30, 700, 100, F, T	55,709	58,804	5.56%	57,886	3.91%	57,886	3.91%
	(t)	(1,735.72)		(1,735.72)		(4,866.11)	
30, 520, 400, V, L	112,997.5	125,831	11.36%	122,048	8.01%	120,652	6.77%
	(t)	(5,789.27)		(5,789.27)		(36,530.8)	
30, 520, 400, F, L	X	177,409	—	167,837	−5.40%	161,098	−9.19%
	(t)	(6,406.62)		(6,406.62)		(429,296)	
30, 520, 400, V, T	X	125,518	—	121,603	−3.12%	121,588	−3.13%
	(t)	(6,522.23)		(6,522.23)		(28,214)	
30, 520, 400, F, T	X	174,526	—	171,641	−1.65%	167,939	−3.77%
	(t)	(8,415.24)		(8,415.24)		(40,010.9)	
30, 700, 400, V, L	X	110,000	—	108,029	−1.79%	106,777	−2.93%
	(t)	(12,636.2)		(12,636.2)		(24,816.8)	
30, 700, 400, F, L	X	165,484	—	154,215	−6.81%	148,950	−9.99%
	(t)	(11,367.70)		(11,367.70)		(69,540.1)	
30, 700, 400, V, T	X	103,768	—	102,468	−1.25%	101,672	−2.02%
	(t)	(15,879.50)		(15,879.50)		(34,974.9)	
30, 700, 400, F, T	X	150,919	—	148,243	−1.77%	142,778	−5.39%
	(t)	(11,660.40)		(11,660.40)		(51,877.9)	

approaches, two results are displayed for the cycle-based metaheuristic. The TABU-CYCLE column displays solutions found by our approach using the same CPU time reported for the path-based method. Longer runs were performed to further illustrate and analyze the behaviour of the algorithm we propose. Column TC (400) displays computational results after 400 iterations. For the three metaheuristic results, the next column displays the optimality GAP relative to the branch-and-bound solution. When branch-and-bound has failed to identify a feasible solution, as in the last part of Table 3, the gap relative to the path-based tabu search is displayed instead. For all methods, the figures in parentheses represent total computational time in CPU seconds on the appropriate computers.

A first conclusion that emerges from the two tables is that fixed-cost, capacitated, multicommodity network design problems are indeed difficult to solve, as indicated by the performance of a state-of-the-art mixed-integer programming solver. Experimental results also indicate clearly that TABU-CYCLE outperforms TABU-PATH for an equivalent computational effort in almost all experiments. Thus, out of 43 problem instances, only for 9 relatively small problems are the solutions of TABU-PATH better than those of TABU-CYCLE. Most of these problems are "easy," however. That is, problem dimensions are sufficiently small for an advanced branch-and-bound method to find the optimal solution in very reasonable computing times. There is no need for metaheuristics using complex neighbourhoods to address such problems. One reaches very rapidly, however,

problem sizes and characteristics for which comparisons and the use of advanced metaheuristics are meaningful and necessary.

The superiority of TABU-CYCLE thus apears to be greater when fixed costs are high: The optimality gap of TABU-CYCLE for these problems is at most 14%, while it reaches 33% for TABU-PATH. The average percentage improvement of TABU-CYCLE compared to TABU-PATH is around 3.36%, with a maximum improvement of 18.13%. These results are a first indication of the effectiveness of the cycle-based neighbourhood structures to generate good solutions for the CMND.

The results displayed in Tables 2 and 3 also support the hypothesis that our algorithm may identify higher-quality solutions given longer search times. Using 400 iterations, improved solutions were found for 35 out of the 41 problems (two solutions were already optimal). Moreover, the benefits obtained from increasing the computational effort appear to become greater for larger instances. Thus, for example, for problems with 200 commodities the maximum optimality gap is decreased from 13.21% to 7.80%. Compared to TABU-PATH, the average improvement obtained by TC (400) is increased to 4.93% and the maximum improvement is increased to 22.30%. The results discussed in the rest of the section are obtained by using 400 iterations of the cycle-based tabu search.

A more in-depth study of the performance of the proposed methodology was conducted using the second set of problem instances, denoted **R**. These problems were generated to facilitate the study of the impact of particular problem characteristics on algorithmic performance. The number of nodes, design arcs, and commodities are systematically varied, one at a time, and for each of these networks, nine problem instances were created by combining three levels of fixed cost ratios and three levels of capacity ratios. The *fixed cost ratio* is computed as $|\mathcal{P}| \sum_{(ij)\in\mathcal{A}} f_{ij} / \sum_{p\in\mathcal{P}} w^p \sum_{(ij)\in\mathcal{A}} c_{ij}^p$, and the three values considered are $F01 = 0.01$, $F05 = 0.05$, and $F10 = 0.10$, corresponding to continuously higher levels of fixed costs compared to the routing costs. The *capacity ratio* is computed as $|\mathcal{A}| \sum_{p\in\mathcal{P}} w^p / \sum_{(ij)\in\mathcal{A}} u_{ij}$, and the values considered are $C1 = 1$, $C2 = 2$, and $C8 = 8$, indicating that the total capacity available becomes increasingly tighter relatively to the total demand. Thus, the easiest problems generally have F01 and C1 ratios, while the most difficult ones have F10 and C8 characteristics. Detailed results are reported in Ghamlouche et al. (2001a). Aggregated statistics are used in the following to support the discussion.

Table 4 displays the distribution of the optimality gap relative to CPLEX branch-and-bound for the two sets of problem instances, obtained by the cycle-based tabu search after 400 iterations. The first column identifies the problem set, the second the total number of instances in the set, the third indicates the number of problems where branch-and-bound did not find a feasible solution (in 10 hours), the forth indicates the number of optimal solutions found by the metaheuristic, while the fifth displays the number of problems where the heuristic solution is better than the one found by branch-and-bound after 10 hours of computation. The next six columns correspond to the indicated gap intervals. For problems in set **C**, the optimality gap never exceeds 8% and is often much smaller than 2%, for an average within 2.10% of the best solutions found by branch-and-bound. For the 153 problems of set **R**, 28 optimum solutions were found, 110 solutions are within 8%, 79 of which are within 4%, while only 7 solutions display a gap greater than 10%. The average gap over all **R** problems is 2.97%. This constitutes a very good performance, especially given the dimensions and difficulty of most problem instances treated.

Tables 5 and 6 summarize information on the behaviour of the cycle-based tabu search according to various problem characteristics. This information appears under the heading TABU-CYCLE (400). These results are also contrasted to the corresponding behaviour of the path-based tabu search. The statistics are computed over the problem instances of set **R**.

Table 5 displays the optimality gap distributions of TABU-PATH and TABU-CYCLE (400) according to the fixed cost and the capacity ratios. One notices, as expected, that more tightly constrained problems are more difficult to solve and thus, for the same computational effort, somewhat less good results are obtained. Compared to the path-based method, however, the cycle-based procedure displays a more consistent behaviour and appears significantly more robust with respect to variation in these two characteristics. Thus, for example, the average gap is not larger than 4.40% contrasted to the almost 20% of TABU-PATH. In fact, the cycle-based method cuts the optimality gaps in all cases, significantly more so (by at least a factor of 2) as problems become more difficult. Comparing no longer absolute results but rather the general trends displayed by the two approaches, it becomes apparent, however, that the cycle-based method seems to encounter more difficulties when capacities become tighter. A plausible explanation may follow from the fact that when capacities are tight, a large number of moves that consider all flows on arcs will result in infeasible solutions. Then, more time is spent to achieve feasibility and the actual search is less thorough. The superiority of the cycle-based method is not in doubt, yet these results point to the need to develop more elaborate search strategies to successfully address the full range of CMND problems. Such a development is beyond the scope of this paper, however.

Table 4. Distribution of relative gaps.

P. Set	Card.	X	Opt.	IMP	(0-2%]	(2-4%]	(4-6%]	(6-8%]	(8-10%]	>10%
C	43	7	3	2	15	6	4	6	—	—
R	153	—	28	—	46	33	20	11	8	7

Table 5. Gap distribution according to fixed cost and capacity level.

Tabu-Path				Tabu-Cycle (400)			
	C1	C2	C8		C1	C2	C8
F01	2.49%	2.21%	2.96%	F01	1.48%	1.31%	1.74%
F05	12.30%	9.16%	6.33%	F05	3.49%	3.69%	4.14%
F10	19.86%	14.45%	8.6%	F10	3.55%	4.27%	4.40%

Table 6 displays results for the distribution of optimality gaps according to problem dimensions. The same general behaviour as discussed previously is also observed in this case. For the same computation effort, larger problem instances are, as expected, more difficult to address. On the other hand, this difficulty increases slowly, much more slowly than for the path-based procedure. In fact, one observes again the robustness of the cycle-based metaheuristic with respect to problem dimensions, including the number of commodities. All these results confirm the interest of cycle-based neighbourhoods and the superiority of TABU-CYCLE in addressing CMND problems.

To complete our evaluation of the impact of the random seed in the selection of the candidate set, we solved all **C** problem instances three times. We observed some differences in solution values due to different seeds (no significant differences were observed in computing times). Differences were relatively small, however, and did not change any of the previous discussions or conclusions. Consequently, to avoid adding to the length of this paper, we do not include these results. The issue is addressed in more details in Ghamlouche et al. (2001a).

Summarizing the computational results after testing on 196 problem instances, we conclude that cycle-based neighbourhoods help build metaheuristics that yield very good solutions to the CMND. The TABU-CYCLE improves almost all TABU-PATH solutions. The average improvement is around 4.75%, while the maximum improvement is 33.7%. Compared to state-of-the-art branch-and-bound methods, the method finds very good quality solutions with a reasonable computation effort. More importantly, cycle-based tabu search displays a robust behaviour relative to variations in problem dimension, including number of commodities, fixed to routing cost ratios, and capacity tightness levels.

6. CONCLUSIONS

In this paper, we proposed new cycle-based neighbourhood structures for the fixed-charge capacitated multicommodity network design problem. These neighbourhoods are different from traditional ones (add/drop/swap) used in the literature for combinatorial optimization problems. They allow us to identify moves based on potential variations in the flow distribution of several commodities, and that impact the status of several design arcs simultaneously.

To evaluate the quality of these neighbourhoods, a simple tabu-based local search method has been developed. We realize that this is not a very advanced metaheuristic. Indeed, it does not include some of the very basic features characteristic of tabu search method, yet a simple design does not overshadow the contribution of the proposed neighbourhood structures. We were thus able to clearly analyze their performance and to identify desirable future developments. Numerical experiments have shown this method to display a consistently superior and robust performance. It generates solutions of high quality and outperforms other heuristics proposed in the literature. It is, in fact, the best current heuristic for fixed-charge, capacitated, multicommodity network design problems.

Table 6. Gap distribution according to problem dimensions.

$\|P\|$	$\|N\|, \|A\|$	Opt. Gap	$\|N\|, \|A\|$	Opt. Gap	$\|N\|, \|A\|$	Opt. Gap
			Tabu-Path			
10		0.61%		1.81%		2.04%
25	10,25	0.56%	10, 50	2.78%	10, 75	4.03%
50		1.54%		7.69%		8.96%
			Tabu-Cycle (400)			
10		0.00%		0.10%		0.41%
25	10, 25	0.78%	10, 50	0.48%	10, 75	0.91%
50		1.63%		2.47%		2.87%
			Tabu-Path			
40		5.00%		8.43%		6.47%
100	20, 100	9.03%	20, 200	16.87%	20, 300	20.8%
200		8.06%		24.4%		23.24%
			Tabu-Cycle (400)			
40		2.78%		3.50%		2.90%
100	20, 100	2.69%	20, 200	6.32%	20, 300	5.47%
200		5.12%		6.82%		8.19%

Several interesting research avenues are now before us. One avenue consists of studying the impact of the linear programming methodology used to solve the minimum cost network flow problem on the efficiency and quality of the method. This may be particularly important as increasingly larger problem instances are addressed. Parallelization strategies also contribute toward this goal. A second development direction consists of building more complete tabu search metaheuristics that use cycle-based neighbourhoods. Long-term memory-based strategies (e.g., path relinking and scatter search) will be investigated as efficient means to diversify the search and improve its performance. Preliminary results appear extremely promising (Ghamlouche et al. 2001b). Finally, the application of cycle-based neighbourhoods to other difficult combinatorial optimization problems opens up intriguing but challenging perspectives.

ACKNOWLEDGMENTS

Funding for this project was provided by the Natural Sciences and Engineering Council of Canada through its Research Grant program and through the Network for Mathematical Modelling and Computation, and by the Fonds F.C.A.R. of the Province of Québec.

REFERENCES

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows—Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Balakrishnan, A., T. L. Magnanti, P. Mirchandani. 1997. Network Design. M. Dell'Amico, F. Maffioli, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization.* John Wiley and Sons, New York, 311–334.

CPLEX. 1999. *ILOG CPLEX 6.5*. ILOG, Mountain View, CA.

Crainic, T. G., J. M. Rousseau. 1986. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Res. B: Methodology* **20B** 225–242.

——, A. Frangioni, B. Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated network design. *Discrete Appl. Math.* **112** 73–99.

——, M. Gendreau, J. M. Farvolden. 2000. A simplex-based tabu search method for capacitated network design. *INFORMS J. Comput.* **12**(3) 223–236.

——, ——, P. Soriano, M. Toulouse. 1993. A tabu search procedure for multicommodity location/allocation with balancing requirements. *Ann. Oper. Res.* **41** 359–383.

Farvolden, J. M., W. B. Powell. 1994. Subgradient methods for the service network design problem. *Transportation Sci.* **28**(3) 256–272.

Gendron, B., T. G. Crainic. 1994. Relaxations for multicommodity network design problems. Publication CRT-965, Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada.

——, ——. 1996. Bounding procedures for multicommodity capacitated network design problems. Publication CRT-96-06, Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada.

——, ——, A. Frangioni. 1998. Multicommodity capacitated network design. B. Sansó, P. Soriano, eds. *Telecommunications Network Planning*. Kluwer, Norwell, MA, 1–19.

Ghamlouche, I., T. G. Crainic, M. Gendreau. 2001a. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. Publication CRT-2001-01, Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada.

——, ——, ——. 2001b. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. Publication CRT-2002-01, Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **1**(3) 533–549.

——. 1989. Tabu search—Part I. *ORSA J. Comput.* **1**(3) 190–206.

——. 1990. Tabu search—Part II. *ORSA J. Comput.* **2**(1) 4–32.

——, M. Laguna. 1997. *Tabu Search*. Kluwer, Norwell, MA.

Holmberg, K., D. Yuan. 2000. A Lagrangean heuristic based branch-and-bound approach for the capacitated network design problem. *Oper. Res.* **48**(3) 461–481.

Jarvis, J. J., O. Mejia de Martinez. 1977. A sensitivity analysis of multicommodity network flows. *Transportation Sci.* **11**(4) 299–306.

Koskosidis, Y. A., W. B. Powell, M. M. Solomon. 1992. An optimization-based heuristic for vehicle routing and scheduling with soft time windows constraints. *Transportation Sci.* **26** 69–85.

Magnanti, T. L., R. T. Wong. 1986. Network design and transportation planning: Models and algorithms. *Transportation Sci.* **18**(1) 1–55.

Minoux, M. 1986. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks* **19** 313–360.

Powell, W. B. 1986. A local improvement heuristic for the design of less-than-truckload motor carrier networks. *Transportation Sci.* **20**(4) 246–357.