Learning Mechanisms and Local Search Heuristics for the Fixed Charge Capacitated Multicommodity Network Design

Ilfat Ghamlouche[1], Teodor Gabriel Crainic[2], Michel Gendreau[3] and Ihab Sbeity[4]

[1] Faculté des Sciences Économiques et de Gestion, Université Libanaise
Beirut, Hadath C.P: 6573-14 , Lebanon

[2] Département de management et technologie
Université du Québec à Montréal
and
CIRRELT, Université de Montréal
Montréal, Québec, Canada

[3] CIRRELT, Université de Montréal
Montréal, Québec, Canada

[4] Faculté des Sciences, Université Libanaise
Beirut, Hadath C.P: 6573-14 , Lebanon

Abstract

In this paper, we propose a method based on learning mechanisms to address the fixed charge capacitated multicommodity network design problem. Learning mechanisms are applied on each solution to extract meaningful fragments to build a pattern solution. Cycle-based neighborhoods are used both to generate solutions and to move along a path leading to the pattern solution by a tabu-like local search procedure. Within this concept, the method integrates important mechanisms such as intensification and diversification. Experimental results show that the proposed algorithm is effective for large structured instances with several commodities.

Keywords: *Adaptive memories, Tabu search, fixed charge capacitated multicommodity network design, Meta-heuristics, Cycle-based neighborhoods.*

1. Introduction

The fixed-charge capacitated multicommodity network design problem (CMND) has various applications in the field of transportation, telecommunication and production planning (Balakrishnan, Magnanti, and Mirchandani [9], Magnanti and Wong [1], Minoux [2]). In these applications, multiple commodities (goods, data, people, etc.) must be routed between different points of origin and destination over a network of limited capacities. 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) for 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 in order to minimize the total system cost, computed as the sum of the fixed and routing costs, while satisfying the demand for transportation.

The fixed-charge capacitated multicommodity network design problem is one of the most difficult NP-hard combinatorial optimization problems. Existing exact algorithms are not yet capable to handle problems of realistic sizes (Crainic, Frangioni, and Gendron[23], Gendron, Crainic, and Frangioni[11], Holmberg and Yuan[19], Sellmann, Kliewer et Koberstein [27]). Therefore, there is substantial interest in developing heuristic procedures for this problem (Crainic, Gendreau, and Farvolden [15], Crainic, Gendron and Hernu [10]). Currently, the best available heuristic procedures are the cycle-based tabu search and the path relinking algorithms developed by Ghamlouche, Crainic, and Gendreau ([25],[26]). In the first paper, the authors propose a new class of neighborhood structures for the CMND and evaluate these neighborhoods using a very simple tabu-based local search procedure. The approach appears robust in terms of solution quality and computing efficiency. However, it does not go beyond a rather local exploration of the search space.

Adaptive memories appear as important building blocks for designing a complete tabu search (Glover and Laguna [8], Glover [6], Glover, Taillard and de Werra [3]. Adaptive memories may be explicit or attributive. Explicit memory records complete solutions, typically consisting of elite solutions visited during the search while attributive memory records information about solution attributes that change in moving from one solution to another. Ghamlouche, Crainic and Gendreau [26] developed a path relinking method based on cycle-based tabu search that offers the best current performance among approximate solution methods for the CMND. The method makes use of explicit memory to record elite solutions. Then the process explores paths between elite solutions in order to generate improved new ones.

The motivation of this paper is to investigate effects of adding learning mechanisms to the cycle based tabu search introduced in [25]. We aim in particular, to develop more general guidelines for the neighborhood exploration by focusing on attributive memories. Our main contribution is the adaptation to the fixed-charge capacitated multicommodity network design problem, of concepts widely used in Tabu search, such as intensification and diversification mechanisms.

The outline of the paper is as follows. Section 2 describes the problem then Section 3 provides the necessary background and fundamentals. Section 4 details the implementation of our learning mechanisms.
Section 5 is dedicated to experimental results. We conclude in Section 6.

2. Problem Formulation and Notation

The goal of a CMND formulation is to find the optimal configuration - the links to include in the final design - of a network of limited capacity to satisfy the demand of transportation of different commodities sharing the network. The objective is to minimize the total system cost, computed as the sum of the link fixed and routing costs.

Given a set of commodities P, the CMND can be defined on a network $G=(N,A)$ where N is the set of nodes and A is the set of directed arcs. A cost C_{ij}^p is associated to each unit flow of commodity p on arc (i,j), and a fixed cost f_{ij} has to be paid in order to use arc (i,j) at all. Without loss of generality, we assume that 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

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

Subject to

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

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

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

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

Where y_{ij}, $(i,j)\in A$, 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 $p\in P$ on arc (i,j), and

$N^+(i)/N^-(i)$: *Set of outward / inward neighbors of node i*

u_{ij} : *Capacity applied on arc* (i,j)

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

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 (5) and (4) are the usual non-negativity and integrality constraints for decision variables. 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)

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

Subject to (2) *and*

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

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

where $A(\bar{y})$ stands for the set of arcs corresponding to the design $\bar{y}$. A solution to the CMND may then be viewed as 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 $x^*(\bar{y})$

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

3. Background and Fundamentals

The necessary background of the cycle-based tabu search is outlined here. For more details, see Ghamlouche, Crainic and Gendreau [25]. The class of neighborhood structures proposed by Ghamlouche, Crainic and Gendreau [25] for the CMND explores the space of the arc design variables by redirecting flow around cycles and closing and opening design arcs accordingly. The neighborhood 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.

The fundamental idea 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. Such neighborhoods are huge, however, 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 computation intensive. Thus, in order to select the best move out of a given solution, the method implements an efficient procedure that 1) avoids the complete evaluation of every examined move and 2) generates a limited number of cycles that include the "good" moves. Note that not all cycles are of equal interest. The method seeks 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, one must be able to deviate all its flow. The residual capacity of any cycle that includes that arc must then be at least equal to the total flow on the arc. Consequently, the cycles of interest are those that display a residual capacity equal to one of the values in the set of the total (strictly positive) volumes on the open arcs.

Cycles are thus to be identified on residual networks and the one leading to the network modification that yields the largest improvement (smallest deterioration, eventually) in the design objective function corresponds to the best move. To reduce the computational burden, cycles are identified and evaluated for a set of candidate links C. The "lowest" cost cycle for each candidate link is identified by an optimization heuristic based on a modification of the shortest path label-correcting algorithm that avoids getting trapped in negative directed cycles. The method thus progressively builds a set of good candidate neighbors (cycles) among which the best move is then selected.

To evaluate these concepts, Ghamlouche, Crainic, and Gendreau [25] developed a simple tabu search-based local search procedure that integrates two versions of the cycle-based neighborhood: 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.

Following an initialization phase, the tabu search procedure explores the design variables solution space using a simple local search framework: at each iteration, the best non-tabu move is determined and implemented regardless 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 neighborhoods that consider the flow distribution of one commodity only. A solution is considered particularly good when it improves the best overall solution or is close to it by at least a pre-defined percentage. The method terminates whenever a predefined stopping criterion (number of iterations, CPU time, etc.) is met. Computational results on a large set of instances, with various characteristics, show that the cycle-based tabu search produces superior solutions.

Ghamlouche, Crainic and Gendreau [26] explore the adaptation of path relinking to the (CMND). Path relinking (Glover [7]; see also Glover and Laguna [8] and Glover, Laguna, and Marti [16]), is a meta-heuristic that operates on a set of elite solutions, called the reference set, and generates paths between solutions in this set to create improved new ones. Starting from an "initial" solution, the primary goal of the search is to find a path to reach another, "guiding" solution, by performing moves that progressively introduce into the current solution attributes contained in the guiding solution. Thus, the method does not progress by choosing a "best" move from the

neighborhood set, but by selecting the "best" move from the restricted set of moves that incorporate some or all of the attributes of the guiding solution. This exploration allows the search to perform moves that may be considered unattractive according to the objective function value but which appear essential in reaching solutions with given characteristics.

To implement path relinking, Ghamlouche, Crainic and Gendreau [26] used the cycle-based neighborhoods both to move along a path between elite solutions and to generate the elite candidate set by a tabu-like local search procedure. The authors proposed and compared several strategies to build the reference set and to select initial and guiding solutions from this set. The best strategies for the fixed charge capacitated multicommodity network design problem is to build the reference set with improving local minima, that is local minimum solutions that offer a better evaluation of the objective function than those already in the reference set, and to build paths between the most distant solutions in this set, that is with solutions having the maximum Hamming distance.

Extensive computational experiments indicate that the path relinking procedure offers excellent results. It systematically outperforms the cycle-based tabu search method in both solution quality and computational effort and offers the best current metaheuristic for this difficult class of problems.

4. Learning Mechanisms

During the cycle-based tabu search, each solution found depends only on the previous one. To take into consideration the history of the search, we decided to modify the cycle-based tabu search by adding learning mechanisms performed at each iteration and use this knowledge to build further solutions. To this end, we use adaptive memories that give us an overview on each arc of the network. For each arc, two adaptive memories *ArcToOpen* and *ArcToClose* are used for intensification purposes and one adaptive memory, *ArcResidency*, is used for diversification. Intensification adaptive memories record how many times it was useful to have the arc opened and how many times it was not. In particular, $ArcToOpen(i, j)$ indicates the number of times arc (i, j) is useful to be opened while $ArcToClose(i, j)$ indicates the number of times arc (i, j) is useful to be closed. The third memory, $ArcResidency(i, j)$, stores the number of times arc (i, j) has been used in a solution. *ArcResidency* is later used to diversify the search by penalizing highly used arcs and favor not much used arcs (see section 4.6).

For each solution, to decide whether it is good or not to open each arc, we investigate two strategies: both study the contribution of the arc to the solution cost. However in the first one, identified as arc strategy, we evaluate each arc independently while in the second, identified as node strategy, we evaluate each arc within a subset of arcs having the same origin node.

4.1 Arc Strategy

Two measures are used to evaluate the status of arcs of each solution of the modified cycle-based tabu search: the flow and the fixed cost. Ideally, we would like to use arcs with low fixed cost; in addition, we would like those arcs to have a very high flow. Such arcs are the most attractive and thus one would like to open these arcs in the next solutions. Arcs with high fixed cost and low flow are poor candidates to be included in the next solutions and one would like to close all these arcs when building further solutions. Arcs with high fixed cost and high flow cannot be rejected (closed) unless we take a closer look to the fixed cost over capacity ratio. In fact, those arcs can be distributed in two subsets: those having high fixed cost over total flow ratio and those having low fixed cost over total flow ratio.

Arcs in the first set (high fixed cost, high flow and high fixed cost over total flow ratio) lead to a costly objective value and should be closed in the next solutions. On the other hand, arcs in the second set (high fixed cost, high flow and low fixed cost over total flow ratio) are less expensive than those in the first set but we still do not like to open them in the next solutions because of the high fixed cost. The same argument applies on arcs having low fixed cost and low flow. Section 4.3 gives an evaluation mechanism to determine, for each arc, whether the flow and the fixed cost are high or low. Adaptive memories are updated for each arc of the solution by incrementing *ArcToOpen* or *ArcToClose* depending on how we would like to have the status of the arc in the next solutions. Table 1 summarizes the arc strategy.

In order to identify at each solution $(\bar{x}, \bar{y})$ arcs with the highest or lowest fixed cost over total flow ratio, let H be the set of all arcs having high fixed cost and high flow and L be the set of all arcs having low fixed cost and positive low flow. An arc (i, j) belonging to H is considered to have a high fixed cost over its total flow ratio if this ratio exceeds the threshold t_1 (i.e. $f_{ij}\,\bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^{p} > t_1$) where:

$$t_1 = \sum_{(i,j) \in H} (f_{ij}\,\bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^{p}) / |H|$$

In the same way, an arc (i, j) belonging to L is considered to have a low fixed cost over its total flow ratio if this ratio is less than t_2 (i.e. $f_{ij}\,\bar{y}_{ij} / \sum_{p\in P} \bar{x}_{ij}^{p} < t_2$) where:

$$t_2 = \sum_{(i,j)\in H} (f_{ij}\,\bar{y}_{ij} / \sum_{p\in P} \bar{x}_{ij}^{p}) / |L|$$

Table 1: Arc Strategy

ARCS	HIGH FLOW	LOW FLOW
HIGH FIXED COST	Increment *ArcToClose* for arcs having the highest fixed cost over total flow ratio	Increment *ArcToClose* for all used arcs
LOW FIXED COST	Increment *ArcToOpen* for all used arcs	Increment *ArcToOpen* for arcs having the lowest fixed cost over total flow ratio

4.2 Node Strategy

In another attempt to use flow and fixed cost to build adaptive memories, information on arcs are grouped and transferred to their originated nodes as follow:

F_i = *The total fixed cost on outgoing arcs from node i having positive flow* $(\sum_{(i,j)\in A/\ j\in i^+} f_{ij}\bar{y}_{ij})$

X_i = *The total flow on outgoing arcs from node i* $(\sum_{p\in P}\sum_{(i,j)\in A/\ j\in i^+} \bar{x}_{ij}^{p})$

Where $(\bar{x}, \bar{y})$ denote the current solution and j is the node successor of i. Section 4.3 gives an evaluation mechanism to determine, for each node, whether the flow and the fixed cost are high or low.

Transferring information from arcs to nodes is based on the fact that flow is traveling on arcs; flow is then grouped on nodes and then redistributed on arcs. Consequently, at each node, the cost of this redistribution is the total cost on outgoing arcs. In node strategy, adaptive memories are updated according to the influence of this redistribution on the objective value as follows:

_ If node i has a high fixed cost and a low flow, the distribution of the flow is very costly and we want to avoid this redistribution in the next solutions. In fact, we want to push the flow back to use different distribution channels. This can be realized by closing all used arcs originating from node i and thus *ArcToClose* will be incremented for those arcs.

_ If node i has a low fixed cost and a high flow, the distribution cost of the flow is very low, thus we want to favor this distribution in next solutions. This can be realized by keeping all used arcs originating from node i opened. Consequently, *ArcToOpen* will be incremented for those arcs.

_ If node i has a high fixed cost and a high flow, the distribution cost is more or less acceptable. In this case, we might need to tighten the flow by closing some arcs. In fact we are interested in preventing part of the flow (i.e. causing high redistribution cost) to be redistributed on outgoing arcs in the next solutions. To identify the part of the flow to be pushed back, let H_i be the set of used arcs originating from node i. An arc (i, j) belonging to H_i should be closed (*ArcToClose* incremented) if its fixed cost over total flow ratio exceeds the threshold t_{ci}, computed as the average fixed cost over total flow of all used arcs originating at node i. Explicitly, *ArcToClose* (i, j) is incremented if:

$$\frac{f_{ij}\bar{y}_{ij}}{\sum_{p\in P}\bar{x}_{ij}^{p}} > t_{ci} = \frac{\sum_{(i,j)\in H_i}(f_{ij}\bar{y}_{ij} / \sum_{p\in P}\bar{x}_{ij}^{p})}{|H_i|}$$

_ If node i has a low fixed cost and a low flow, the distribution cost is also more or less acceptable. In fact, only the part of the flow leading to the lowest redistribution cost is attractive and we would like to keep this part redistributed on outgoing arcs in the next solutions. To identify the part of the flow to be pushed forward, let Li be the set of used arcs originated at node i. An arc (i, j) belonging to L_i should be kept opened (*ArcToOpen* incremented) if its fixed cost over total flow ratio is lower than a threshold t_{oi}, computed as the average fixed cost over total flow of all used arcs originated at node i. Explicitly, *ArcToOpen* (i, j) is incremented if:

$$\frac{f_{ij}\bar{y}_{ij}}{\sum_{p\in P}\bar{x}_{ij}^{p}} < t_{oi} = \frac{\sum_{(i,j)\in L_i}(f_{ij}\bar{y}_{ij} / \sum_{p\in P}\bar{x}_{ij}^{p})}{|L_i|}$$

4.3 Arc Strategy

Two thresholds are used to explicitly identify low/high fixed cost and low/high flow on arcs (or nodes) at each solution of the modified cycle-based tabu search. These thresholds depend on the current best solution found and thus are tighter when progresses the search. Those thresholds are calculated as follows:
Let x and f denote respectively the percentage of used capacity and the average used fixed cost of the current best solution $(\tilde{x}, \tilde{y})$:

$$x = \frac{\sum_{p \in P} \sum_{(i,j) \in A} \tilde{x}_{ij}^{p}}{\sum_{(i,j) \in A} u_{ij} \tilde{y}_{ij}}$$

$$f = \frac{\sum_{(i,j) \in A} f_{ij} \tilde{y}_{ij}}{\sum_{(i,j) \in A} \tilde{y}_{ij}}$$

In arc strategy, an arc (i, j) of the current solution $(\bar{x}, \bar{y})$ has a high fixed cost if its used fixed cost exceeds f ($f_{ij} \bar{y}_{ij} > f$) otherwise the arc has a low fixed cost ($f_{ij} \bar{y}_{ij} \leq f$). In the same way, arc (i, j) has a high flow if its total flow over total capacity ratio exceeds x ($\sum_{p \in P} \bar{x}_{ij}^{p} / u_{ij} > x$) otherwise the arc has a low flow.

Following the same analogy with node strategy, a node i of the current solution $(\bar{x}, \bar{y})$ has a high fixed cost if the average fixed cost on outgoing arcs from node i exceeds f . Explicitly:

$$\frac{F_i}{\sum_{(i,j) \in A / \; j \in i^{+}} \bar{y}_{ij}} > f$$

In the same way, node i has a high flow if the percentage of used capacity on outgoing arcs from node i exceeds x . Explicitly:

$$\frac{X_i}{\sum_{(i,j) \in A / \; j \in i^{+}} u_{ij} \bar{y}_{ij}} > x$$

4.4 Pattern solution

To move from a current solution to a neighboring one in our approach, we select a set of candidate arcs from the network and we perform cycle-based tabu search to get the best move. Remember that at least one of the candidate arcs will have its status changed after the move. In the original form of the cycle-based tabu search, the candidate arcs were selected randomly from the set of closed arcs. In our approach, we will use the learning mechanisms to decide which arcs should be included in the candidate set. Each closed arc is a candidate arc if *ArcToOpen* exceeds *ArcToClose* by a predefined value. This means that during the search, it was more useful to have the arc opened than to have it closed. However, since the arc is closed in the current solution, we want to direct the search to open the arc without forcing it to be opened. Similarly, each opened arc is a candidate arc if *ArcToClose* exceeds *ArcToOpen* by a predefined value.

Pattern solution is the result of our learning during the search. In *pattern* solution, an arc is opened if its associated *ArcToOpen* exceeds *ArcToClose* by a predefined value *OpenTheArc* and an arc is closed if its associated *ArcToClose* exceeds *ArcToOpen* by a predefined value *CloseTheArc*. Originally, all arcs in pattern solution have an undecided status. In this way, arcs with different status between current and pattern solution constitute the set of candidate arcs when performing a cycle-based tabu search.

4.5 Intensification

Building the set of candidate arcs as in section 4.4 constitute intensification in the already explored neighborhood since we seek, via the *pattern* solution, to open arcs found good during the search and to close arcs found to be costly. The intensification consists on using the cycle-based tabu search to find the best move starting from the current solution and building and maintaining the *pattern* solution as in section 4.4. The intensification phase ends after a given number of iterations, *MaxInt*, without improvement of the objective function value.
Note that, even if adaptive memories are updated after each iteration of the search, the *pattern* solution is only updated when no improvement to the current solution is noticed. This is to avoid disturbing the search when improvement is taking place.

4.6 Diversification

Intensification by itself is insufficient to yield the best outcome to our difficult problem. Diversification must be invoked to allow the most effective search over the solution space. To do this, we exploit additional memory means (i.e. residency based memories) to penalize frequently occurred arcs in visited solutions and consequently reach a new search trajectory over the solution space. As in the intensification phase, we use the *pattern* solution to build the set of candidate arcs and the cycle-based tabu search to move from one solution to another. However, the pattern solution is modified to introduce the residency based memory as follow: An arc (i, j) is set to be opened in *pattern* solution if its current status is not decided and its *ArcResidency* is less than a

predefined threshold *ResidMeasure*. An arc (i, j) is set to be closed in *pattern* solution if its current status is not decided and its *ArcResidency* exceeds a *ResidMeasure*. All other arcs of the *pattern* solution will receive a not decided status. The diversification is launched after the intensification phase and performed for *MaxDiv* iterations without improvement of the best solution.

4.7 Path relinking

In our approach, path relinking is implemented to explore trajectories connecting best solutions found during the intensification and the diversification phases.
In particular, when moving toward *pattern* solution in the intensification phase, we keep track of the best r solutions to build a first reference set, then, when the method switches to a diversification phase a second reference set is built with another r best solutions. The method then starts to explore trajectories connecting those solutions: at each iteration, two solutions, one from each reference set, that satisfy the maximum Hamming distance are chosen, the worst one is set to be the starting solution and is removed from its reference set. If during the path relinking exploration, we reach a solution that improves the best overall solution, this solution is added to the reference set of the starting solution. If one of the reference sets is empty, path relinking keeps exploring trajectories between solutions in the remaining reference set. Path relinking ends when both reference sets are empty.

4.8 Warming up

The method needs time to learn therefore a warming up phase is performed. It consists on identifying arcs to open or to close and applying cycle-based tabu search on these arcs to get the best move. At each iteration of the warming up phase, adaptive memories are updated and used to build *pattern* solution. However, at this stage, *pattern* solution is not yet mature to guide the search in the solution space and will be used only at the end of the warming up phase.

Two warming up phases are performed: the first one consists in closing arcs with high fixed cost over total flow ratio while the second consists in opening arcs with low fixed cost over capacity ratio. The percentage of arcs to be closed or opened is set to 50% of the total number of opened or closed arcs. This value was selected as the best during the experimental results reported in Ghamlouche, Crainic and Gendreau [25]. The warming up stops after a given number of iterations, *MaxWarmingUp*, without improvement in the objective function value.

4.9 The search strategy

After some initialization, the method performs a warming up phase to create *pattern* solution. When no improvement is observed, the search proceeds to an intensification phase until a number of iterations without improving the best overall solution is reached.

The method switches then to a diversification phase as indicated in Section 4.6. Path relinking is applied to explore paths connecting best solutions found during the intensification and the diversification phases. The overall process is repeated by starting with the best overall solution. Figure 1 summarized the structure of the search while figure 2 details the learning phase.

5. Experimentation and Computational Results

Experiments have been performed to evaluate the behavior and the performance of the learning algorithm proposed in this paper. To ensure meaningful comparisons, we employ the same two sets of problem instances as used in Ghamlouche, Crainic, and Gendreau ([25],[26]). The heuristic in this paper was implemented in C++. The exact evaluation of the capacitated multicommodity network flow problems is done using the same environment as in Ghamlouche, Crainic and Gendreau [26]. Computing times are reported in seconds.

5.1 Parameter settings

We first performed a calibration phase. An initial set of results (not shown here in order not to overcharge the paper) allowed us to fix the value of *MaxWarmingUp*, *MaxInt* and *MaxDiv* to 10, 40 and 40 respectively. Ten problems have been selected for calibration purposes. The ten problems cover networks sizes from 100 to 700 design arcs and from 10 to 400 commodities. They also display relatively high fixed cost compared to routing cost and are tightly capacitated. We tested the following combinations of parameters:

Initialization

Generate an initial feasible solution to initiate *BestSolution* and *CurrentSolution*.

Let $(\tilde{x}, \tilde{y})$, $(\bar{x}, \bar{y})$ and (x^t, y^t) denote the *BestSolution*, *CurrentSolution* and *PatternSolution* respectively. Set *WarmingUpStatus* = closed

Main search loop

Repeat the following until a stopping condition is met

- Initialize memories
- Repeat until *MaxWarmingUp* is reached
  - _ if (*WarmingUpStatus* = opened)
    - _ Sort closed arcs of the current solution according to f_{ij} / u_{ij}
    - _ Let $\Gamma = \{(i, j) / \bar{y}_{ij} = closed\ and\ f_{ij} / u_{ij}\ is\ low\}$
  - _ else if (*WarmingUpStatus* = closed)
    - _ Sort used arcs of the current solution according to $f_{ij} / \sum_{p \in P} \bar{x}_{ij}^p$
    - _ Let $\Gamma = \{(i, j) / \bar{y}_{ij} = opened\ and \frac{f_{ij}}{\sum_{p \in P} \bar{x}_{ij}^p} is\ high\}$
  - _ Perform one iteration of cycle-based neighborhood by considering arcs in Γ to get a new current solution $(\bar{x}, \bar{y})$
  - _ Perform a Learning phase
  - _ If *CurrentSolution* < *BestSolution* update *BestSolution*
- **Intensification Loop**
  Repeat until *MaxInt* is reached
  - _ Perform one tabu search iteration by introducing arcs present in Pattern solution to get a new solution $(\bar{x}, \bar{y})$
  - _ Perform a Learning phase
  - _ Save best solutions in the first reference set
- Perform a **Diversification phase** by changing Pattern solution
  Repeat until *MaxDiv* is reached
  - _ Perform one tabu search iteration by introducing arcs present in Pattern solution to get a new solution $(\bar{x}, \bar{y})$
  - _ Perform a Learning phase
  - _ Save best solutions in the second reference set
- **Perform a Path Relinking phase between best solutions found during both Intensification and diversification phases**
  - _ Set *CurrentSolution* to *BestSolution*
  - _ *WarmingUpStatus* = opened

- _ for each arc (i, j) of *CurrentSolution* $(\bar{x}, \bar{y})$ with $\bar{y}_{ij} = 1$ increment *ArcResid* (i, j)
- _ Calculate x and f as in section 4
- _ **If arc strategy**

  <u>Low fixed cost, high flow</u>

  - _ If $f_{ij} \bar{y}_{ij} < f$ and $\sum_{p \in P} \bar{x}_{ij}^p / u_{ij} > x$ then increment *ArcToOpen* (i, j)

  <u>High fixed cost, low flow</u>

  - _ If $f_{ij} \bar{y}_{ij} > f$ and $\sum_{p \in P} \bar{x}_{ij}^p / u_{ij} < x$ then increment *ArcToClose* (i, j)

  <u>High fixed cost, high flow</u>

  - _ Calculate the threshold t_1
  - _ Increment *ArcToClose* for each arc having $f_{ij} \bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^p > t_1$ **<u>Low fixed cost, low flow</u>**
  - _ Calculate the threshold t_2
  - _ Increment *ArcToOpen* for each arc having $f_{ij} \bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^p < t_2$
- _ **if node strategy**
  - _ for each node i of *CurrentSolution* $(\bar{x}, \bar{y})$
    - _ Calculate *fixedcost*(i) and flow(i) as in section 4.2

      <u>Low fixed cost, high flow</u>
    - _ If *fixedcost*(i) < f and flow(i) > x then for each arc (i, j) of *CurrentSolution* $(\bar{x}, \bar{y})$ with $\bar{y}_{ij} = 1$ and $j \in i^+$. Increment *ArcToOpen* (i, j)

      <u>High fixed cost, low flow</u>
    - _ If *fixedcost*(i) > f and flow(i) < x then for each arc (i, j) of *CurrentSolution* $(\bar{x}, \bar{y})$ with $\bar{y}_{ij} = 1$ and $j \in i^+$. Increment *ArcToClose* (i, j)

      <u>High fixed cost, high flow</u>
    - _ If *fixedcost*(i)> f and flow(i)> x then calculate the threshold t_{ci} Increment *ArcToClose* for each arc outgoing having $f_{ij} \bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^p > t_{ci}$ **<u>Low fixed cost, low flow</u>**
    - _ If *fixedcost*(i)< f and flow(i)< x then calculate the threshold t_{oi} Increment *ArcToOpen* for each arc outgoing having $f_{ij} \bar{y}_{ij} / \sum_{p \in P} \bar{x}_{ij}^p < t_{oi}$
  - _ if (Warming up) or (*CurrentSolution* ≥ *PreviousSolution*)
    - _ for each arc $(i, j) \in$ *PatternSolution* do
    - _ if *ArcToOpen* (i, j) - *ArcToClose* (i, j) ≥ *OpenTheArc* then y_{ij}^t = Opened
    - _ - if *ArcToClose* (i, j) - *ArcToopen* (i, j) ≥ *CloseTheArc* then y_{ij}^t = Closed

- *OpenTheArc*: This parameter indicates the threshold to exceed in order to open the arc in Pattern solution.
- *CloseTheArc*: This parameter indicates the threshold to exceed in order to close the arc in Pattern solution.

Three values 1, 2 and 3 were considered initially for these parameters. However, the value of 1 for *OpenTheArc* and

CloseTheArc and consequently all combinations (1, 1), (1, 2), (1, 3) as well as (2, 1) and (3, 1), were rapidly dropped since the quality of the solutions started to decrease.

- *ResidMeasure*: This parameter depends on the number of iterations and indicates how often the arc should be opened to be considered highly used. Three values 40%, 60% and 80% of the number of current iteration have been tested.

Node strategy was used for these tests. Each parameter combination was ranked for each problem instance according to the gap relative to the best known solution (that of the branch-and-bound procedure of CPLEX 7.5, when available, or that obtained by Ghamlouche, Crainic and Gendreau [26], otherwise). A score of 10... 1 is assigned to each of the first ten places, respectively. The performance of each parameter setting is then aggregated: gaps are averaged while scores are summed up. Table 2 displays these aggregated results for each parameter combination.

The first column holds the parameter setting; the second column presents the global average gaps while the last column displays the total score.

The results in Table 2 display one set of parameters that offers the most robust combination, *OpenTheArc* = 2, *CloseTheArc* = 3 and *ResidMeasure* = 60%. It offers the lowest average gap and the highest scores. This setting will be maintained in the remaining of this computational study.

Table 2: Parameter SettingPerformances

Parameter settings	Gap	Score
2, 2, 40%	2.35%	59
2, 2, 60%	2.67%	40
2, 2, 80%	2.80%	41
2, 3, 40%	2.36%	33
2, 3, 60%	1.62%	72
2, 3, 80%	1.85%	67
3, 2, 40%	2.16%	69
3, 2, 60%	2.44%	35
3, 2, 80%	2.03%	51
3, 3, 40%	2.42%	63
3, 3, 60%	2.37%	24
3, 3, 80%	2.80%	34

5.2 Performance Analysis

To evaluate the behavior and the performance of the learning algorithm proposed in this paper, we compare its output to the results of the cycle-based tabu search and to those of the path relinking algorithm ([25],[26]). To further characterize the quality of the solutions, we also include the optimal solutions obtained using the branch-and-bound algorithm of cplex 7.5 [12]. The same two data sets of networks used by Ghamlouche, Crainic, and Gendreau [25] were also used to test our learning algorithm. 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. Detailed description of problem instances is given in Crainic, Frangioni, and Gendron [23]; see also Gendron and Crainic ([4], [5]). The problem generators as well as the problem instances can be obtained from the authors.

Problems in the first set of network, denoted C, are defined respectively by the number of nodes, the number of arcs, the number of 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.

Computational results for the first set of networks are reported in Tables 3, 4 and 5. In these tables, the OPT column corresponds to the solution of the branch-and-bound algorithm solved using CPLEX 7.5 [12] on the same workstations. 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. The columns labeled TC and PR hold respectively the best solution, over 3 runs, of the cycle-based tabu search and the path relinking approach while AV.TC and AV.PR columns display respectively the average solution found by these two meta-heuristics. The column LS-NODE and LS-ARC gives the solutions obtained by our approach when using node strategy and arc strategy respectively. When our learning algorithm produces optimal solutions or solutions better than the best solutions found by path relinking, bold characters are employed. The figures in parentheses represent total computation time in CPU seconds. For Comparison purposes, gap is computed for solutions of our learning algorithm with respect to the average solution found by path relinking and displayed in percentage under the CPU time in columns LS-NODE and LS-ARC respectively.

Table 3 shows the results of our learning algorithm on smaller test cases (number of commodities up to 100) while Tables 4 and 5 show the results from runs of larger structured test cases (i.e. with 200 and 400 commodities). From the numerical results, a number of observations can be made. First, the use of adaptive memories is effective for realizing good quality solution for our difficult problem. The results of Table 4 show that our proposed algorithm improves the best solutions found by path relinking for 6 out 8 problems. Results are also encouraging for large real-world problems, such those with 400 commodities (see tables 4 and 5). For this class of difficult instances, CPLEX is unable to find the optimal solution, (not even a

Table 3: Computational Results, C problems

PROBLEM	OPT	TC	PR	AV.TC	AV.PR	LS-NODE		LS-ARC	
25,100,10,V,L	14712 (0.36)	14712 (19.08)	14712 (12.97)	14769.33 (19.38)	14712 (13.01)	14712 (12.36)	0%	14712 (12.26)	0%
25,100,10,F,L	14941 (53.64)	14941 (22.55)	14941 (15.2)	14941 (22.80)	15081.33 (16.3)	14941 (12.9)	-0.93%	14941 (13.6)	-0.93%
25,100,10,F,T	49899 (40.58)	50529 (31.39)	49899 (22.1)	50619.67 (32.10)	50154 (24.5)	50324 (18.75)	0.34%	51328 (21.3)	2.34%
25,100,30,V,T	365272 (16.62)	365385 (121.30)	365322 (91.9)	365385 (123.98)	365323.66 (93.2)	365322 (83.11)	-0.02%	365322 (81.81)	-0.02%
25,100,30,F,T	85530 (534.18)	87325 (123.88)	86428 (97.78)	88095.33 (125.69)	86492.33 (99.67)	86334 (89.31)	-0.18%	86815.3 (84.29)	0.37%
100,400,10,V,L	28423 (84.81)	28786 (208.58)	28485 (83.97)	28836.67 (225.12)	28529 (89.9)	28553 (83.11)	0.08%	28553 (83.31)	0.08%
100,400,10,F,L	24436 (t)	24022 (178.52)	24022 (109.66)	24022 (191.07)	24022 (112.45)	24104 (77.45)	0.34%	24104 (77.5)	0.34%
100,400,10,F,T	66364 (t)	67184 (425.68)	65278 (193.4)	68215.00 (432.01)	65153 (201.34)	66171 (340.63)	0.96%	66410 (218.58)	1.33%
100,400,30,V,T	385544 (t)	385508 (1161.06)	384926 (424.08)	385512.7 (1169.70)	385181.7 (450.76)	384951 (563.7)	-0.06%	384828 (631.31)	-0.09%
100,400,30,F,L	50496 (t)	51831 (730.10)	51325 (328.08)	52176.33 (648.47)	51875.67 (301.4)	53066 (322.31)	2.29%	52173 (293.71)	0.57%
100,400,30,F,T	141278 (t)	147193 (1208.95)	141359 (529.02)	147478 (1235.27)	143403.7 (579.32)	143552 (619.96)	0.10%	142411 (462.89)	-0.67%
30,520,100,F,T	98357 (t)	105130 (2863.76)	106130 (1336.9)	107885.3 (2418.28)	107575 (1405.5)	106912 (1134.01)	-0.62%	107266 (1257.51)	-0.28%
30,700,100,F,T	55709 (t)	57628 (3219.31)	56575.5 (1534.6)	58111.33 (3316.46)	56808.83 (1765.3)	57741 (1840.27)	1.64%	58032.7 (2266.3)	2.15%

Table 4: Computational Results, C problems

PROBLEM	OPT	TC	PR	AV.TC	AV.PR	LS-NODE		LS-ARC	
20,230,200,V,L	94386 (t)	100001 (2577.34)	100404 (2317.35)	101481.3 (2606.79)	101469.3 (2034.54)	102492 (3235.53)	1.01%	102492 (3323.53)	1.01%
20,230,200,F,L	141737.4 (t)	148066 (3143.76)	147988 (2893.49)	148975 (3158.22)	151352 (2760.73)	150617 (2955.02)	-0.49%	151961 (3229.97)	0.40%
20,230,200,V,T	97914 (t)	106868 (2595.35)	104689 (2304.25)	107589.3 (2361.57)	105598.7 (2304.63)	103700 (2921.11)	-1.80%	103700 (2934.62)	-1.80%
20,230,200,F,T	137271 (t)	147212 (3601.90)	147554 (3656.96)	147868 (3868.50)	148044.3 (3505.46)	144895 (4220.34)	-2.13%	149284 (4732.58)	0.84%
30,520,400,V,L	112997.5 (t)	122673 (55771.2)	119416 (29650.7)	123277.3 (55720.30)	119624 (33716.23)	115918 (82551.2)	-3.10%	115918 (95761.4)	-3.10%
30,520,400,F,L	X (t)	164140 (429296)	163112 (33641.2)	165458 (40922.57)	163377 (35671.23)	161205 (54686.2)	-1.32%	159084 (62757.9)	-2.63%
30,520,400,V,T	X (t)	122655 (46565.2)	120170 (31461.9)	123210 (50666.83)	120764.3 (25705.4)	118835 (44631.4)	-1.60%	118705 (60088.6)	-1.70%
30,520,400,F,T	X (t)	169508 (49886.9)	163675 (51400.1)	170301.3 (49476.67)	164921.3 (44862.3)	161102 (114120.4)	-2.32%	161102 (107664)	-2.32%

Table 5: Computational Results, C problems

PROBLEM	OPT	TC	PR	AV.TC	AV.PR	LS-NODE		LS-ARC	
20,300,200,V,L	74972.4 (t)	81367.7 (4247.14)	78184 (3487.2)	82187.23 (3974.91)	79095.33 (3560.5)	79791.5 (4899.92)	0.88%	81307 (4473.87)	2.80%
20,300,200,F,L	117306 (t)	122262 (3267.94)	123484 (3765.34)	123247.7 (4524.49)	124924.7 (3912.5)	128258 (4664.53)	2.67%	125421 (4046.51)	0.40%
20,300,200,V,T	74991 (t)	80344 (4235.15)	78866.8 (3691.24)	82187.23 (4396.22)	79212.27 (3860.3)	81453 (5158.17)	2.83%	81453 (4771.54)	2.83%
20,300,200,F,T	108252 (t)	113947 (4657.54)	113584 (3546.58)	115342 (4906.50)	114632.3 (4001.23)	114269 (4691.77)	-0.32%	114259 (4532.14)	-0.15%
30,700,400,V,L	X (t)	107727 (36332)	105116 (22314.6)	108459.3 (36282.63)	105403.3 (19733.4)	102906 (45637)	-2.36 %	102530 (54264.3)	-2.73%
30,700,400,F,L	X (t)	150256 (73030.8)	145026 (52360.2)	150909 (63170.20)	147887.3 (58761.8)	148862 (128777)	0.66%	146921 (125860)	-0.65%
30,700,400,V,T	(t)	101749 (49324.2)	101212 (26592.3)	103112.3 (49239.5)	102119.3 (28664.3)	98911 (99322.9)	-3.14%	98911 (102727)	-3.14%
30,700,400,F,T	X (t)	144852 (74796.3)	141013 (45179.1)	146705 (79053.40)	141446.7 (49824.83)	139055 (95330.1)	-1.69%	141096 (123233)	-0.25%

feasible solution for 7 out 8 problems) with available computational time. Our learning algorithm improves the best known solutions for 6 out 8 problems and the improvement relative to the average solution of path relinking ranges from 1.60% to 3.14%.

The second observation concerns the computing effort. Even if we account for the fact that we stop our learning algorithm on the same criterion as the path relinking procedure, 400 iterations, it appears that the learning algorithm requires longer computing times, especially when the number of commodities is high. This could be explained by the fact that good solutions are found earlier in the search (of the order of 40% of the total iteration limit), which yields more difficult multicommodity capacitated network flow problems to be solved by CPLEX at each iteration and consequently more time.
The interesting performance and behavior of the proposed learning algorithm is confirmed by the results obtained on the second set of problems instances, denoted R. There are 116 problems divided in 18 groups. Each group contains the same number of nodes, arcs, and commodities but with different combined level of fixed cost and capacity ratios. Three levels of fixed cost and capacity ratios are considered: F01 = 0.01, F05 = 0.05 and F10 = 0.10 indicating continuously higher levels of fixed costs compared to routing costs, and C1 = 1, C2 = 2, and C8 = 8 that signal that the total capacity available becomes increasingly tighter relatively to the total demand. The fixed cost ratio is computed as $|P|\sum_{(i,j)\in A} f_{ij} / \sum_{p\in P} w^p \sum_{(i,j)\in A} c_{ij}^p$, and the capacity ratio is computed as $|A|\sum_{p\in P} w^p / \sum_{(i,j)\in A} u_{ij}$. Only aggregated statistics are used in the following to support the discussion. Table 6 displays the optimality gap distribution, according to problem dimension, for both learning strategy and path relinking algorithms. As we can see, our learning algorithm achieves better results than the average solution of path relinking when the number of commodities increases versus the number of arcs. The results also indicate that our learning algorithm is certainly not interesting for small-sized problem instances with few commodities.

On the other hand, it helps to achieve better solutions, especially in the case of more difficult problems, with a large number of commodities.

According to the results, we observe that our learning algorithm is well fitted to a large number of instances and its performance increases with increasing the difficulty of problems.

6. Conclusion

The objective of our study has been to exploit and advance the knowledge associated with the implementation of learning mechanism with the use of adaptive memories structure for the fixed charge capacitated multicommodity network design problems. In particular, we have undertaken to examine the critical issue of what form of intensification and diversification proves more effective for this class of difficult problem. Our resulting algorithm is effective, and its benefits are particularly significant for large structured instances.

Table 6: Gap Distribution According to Problem Dimensions

|N|, |A|	|P|	PR	AV.PR	LS-NODE	LS-ARC
	10	0%	0%	0%	0%
10,25	25	0.23%	0.60%	0.43%	0.39%
	50	0.61%	0.72%	0.49%	0.44%
	10	0.08%	0.12%	0.25%	0.04%
10,50	25	0.36%	0.55%	0.46%	0.48%
	50	1.14%	1.93%	1.05%	1.15%
	10	0.04%	0.46%	0.89%	1.02%
10,75	25	0.41%	0.92%	0.72%	0.87%
	50	1.52%	2.30%	2.37%	2.40%
	40	1.37%	1.70%	3.02%	2.29%
20,100	100	2.05%	2.82%	2.22%	2.45%
	200	4.55%	4.86%	2.81%	3.26%
	40	3.59%	4.84%	6.29%	5.32%
20,200	100	4.93%	5.90%	7.79%	7.50%
	200	5.41%	6.42%	5.00%	5.07%
	40	2.08%	3.91%	6.89%	7.28%
20,300	100	4.68%	5.79%	7.96%	7.50%
	200	6.84%	8.97%	4.28%	5.12%

References

[1] Magnanti, T.L. andWong, R.T. Network Design and Transportation Planning: Models and Algorithms Transportation Science, Vol. 18(1), 1986, pp. 1-55.
[2] Minoux, M. (1986). Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications Networks, Vol.19, 1986, pp.313-360.
[3] Glover, F., Taillard, E.D., and de Werra, D. A user's guide to tabu search. Annals of Operations Research, Vol. 41, 1993, pp.:3-28.

[4] Gendron, B. and Crainic, T.G. Relaxations for Multicommodity Capacitated Network Design Problems. Publication CRT-965, 1994, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.
[5] Gendron, B. and Crainic, T.G. Bounding Procedures for Multicommodity Capacitated Network Design Problems. Publication CRT-96-06, 1996, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.
[6] Glover, F. Tabu Search and Adaptive Memory Programming-Advances, Applications and Challenges. In Barr, R., Helgason, R., and Kennington, J., editors, Interfaces in Computer Science and Operations Research, 1996, pp. 1-75. Kluwer Academic Publishers, Norwell, MA.
[7] Glover, F. A Template for Scatter search and Path Relinking. in Lecture Notes in Computer Science, 1363, J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. snyers (eds). 1997, pp. 13-54.
[8] Glover, F. and Laguna, M. 1997. Tabu Search. Kluwer, Norwell, MA.
[9] Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. Network Design. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, Annotated Bibliographies in Combinatorial Optimization, 1997, pp. 311-334. John Wiley & Sons, New York, NY.
[10]Crainic, T.G, Gendron, B, and Hernu, G. A slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design. Publication, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada
[11]Gendron, B., Crainic, T.G., and Frangioni, A. Multicommodity Capacitated Network Design. In Sanso, B. and Soriano, P., editors. Telecommunications Network Planning, 1998, pp. 1-19. Kluwer, Norwell, MA.
[12]ILOG CPLEX 7.5. ILOG, Mountain View, 2002, CA. U.S.A.
[13]Laguna, M., and Marti, R. GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization. INFORMS Journal on Com-puting, 1999, Vol. 11(1), pp. 44-52.
[14]Laguna, M., Marti, R., and Campos, V. Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem. Computers and Operations Research, 1999, vol. 22, pp. 1217-1230.
[15]Crainic, T.G., Gendreau, M., and Farvolden, J.M. (2000). A Simplex-Based Tabu Search Method for Capacitated Network Design. IN-FORMS Journal on Computing. forthcoming.
[16]Glover, F., Laguna, M., and Marti, R. Fundamentals of scatter search and path relinking. Control and Cybernetics, 2000, vol. 39(3), pp.653-684.
[17]Crainic, T.G., Frangioni, A., and Gendron, B. Bundle-Based Relaxation Methods for Multicommodity Capacitated Network Design. Discrete Applied Mathematics. forthcoming.
[18]Glover, F., Laguna, M., and Marti, R. Fundamentals of Scatter Search and Path Relinking. Control and Cybernetics, 2000, vol. 39(3), pp.653-684
[19]Holmberg, K., and Yuan, D. A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. Operations Research, 2000, vol. 48(3), pp. 461-481.
[20]Ribeiro, C.C., Uchoa, E., and Werneck, R. A Hybrid GRASP with Perturbations for the Steiner Problem in Graphs.
[21]Ghamlouche, I., Crainic, T.G., and Gendreau, M. Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. Publication CRT-2001-01, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.
[22]Laguna, M., and Armentano, V.A (2001). Lessons from Applying and Experimenting with Scatter Search.
[23]Crainic, T.G. and Frangioni, A. and Gendron, B. Bundle-Based Relaxation Methods for Multicommodity Capacitated Network Design. Discrete Applied Mathematics, 2001, vol. 112, pp. 73-99.
[24]Ghamlouche, I. and Crainic, T.G. and Gendreau, M. Cycle-based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. Publication: Centre de recherché sur les transports, Université de Montréal CRT-2001-01
[25]Ghamlouche, I. and Crainic, T.G. and Gendreau, M. Cycle-based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. Operations Research 2002.
[26]Ghamlouche, I. and Crainic, T.G. and Gendreau, M. Path Relinking, Cycle-based Neighbourhoods and Capacitated Multicommodity Network Design. Annals of Operations Research 2002.
[27]Sellmann, M., Kliewer, G., and Koberstein, A. Capacitated Network Design, Cardinality Cuts and Coupled Variable Fixing Algorithms based on Lagrangian Relaxations. Publication tr-ri-02-234, 2002, University of Paderborn, Department of Mathematics and Computer Science.

.