A SLOPE SCALING/LAGRANGEAN PERTURBATION HEURISTIC WITH LONG-TERM MEMORY FOR MULTICOMMODITY CAPACITATED FIXED-CHARGE NETWORK DESIGN

Teodor Gabriel Crainic[1,2] Bernard Gendron[2,3]
Geneviève Hernu[2]

1 Département de management et technologie,
Université du Québec à Montréal,
Montréal, Québec, Canada H3C 3J7

2 Centre de recherche sur les transports
Université de Montréal,
C.P. 6128, succursale Centre-ville,
Montréal, Québec, Canada H3C 3J7

3 Département d'informatique et de recherche opérationnelle,
Université de Montréal,
C.P. 6128, succursale Centre-ville,
Montréal, Québec, Canada H3C 3J7

February 2003

Abstract

This paper describes a slope scaling heuristic for solving the multicomodity capacitated fixed-charge network design problem. The heuristic integrates a Lagrangean perturbation scheme and intensification/diversification mechanisms based on a long-term memory. Although the impact of the Lagrangean perturbation mechanism on the performance of the method is minor, the intensification/diversification components of the algorithm are essential for the approach to achieve good performance. The computational results on a large set of randomly generated instances from the literature show that the proposed method is competitive with the best known heuristic approaches for the problem. Moreover, it generally provides better solutions on larger, more difficult, instances.

Key words: Slope scaling, Lagrangean heuristic, long-term memory, multicommodity capacitated fixed-charge network design.

Résumé

Nous décrivons dans cet article une heuristique d'ajustement de pente, développée pour résoudre le problème de conception de réseaux multiproduits avec coûts fixes et capacités. L'heuristique combine une methode de perturbation lagrangienne et des mécanismes d'intensification et de diversification basés sur des mémoires à long terme. Les mécanismes d'intensification et de diversification s'avèrent essentiels pour la bonne performance globale de la méthode, tandis que la contribution de la perturbation lagrangienne est modeste. Les résultats expérimentaux obtenus sur un grand ensemble de problèmes test montrent que la méthode proposée est compétitive avec les meilleures heuristiques pour le problème et obtient de meilleures solutions pour les problèmes difficiles de grande taille.

Mots-clés : Ajustement de pente, heuristique lagrangienne, mémoire à long terme, conception de réseaux multiproduits avec coûts fixes et capacités.

We consider the *multicommodity capacitated fixed-charge network design problem (MCFP)*, which can be described as follows. We denote by $G = (N, A, K)$ a directed network, where N is the set of nodes, A is the set of arcs, and K is the set of commodities, or origin-destination pairs. For each commodity $k \in K$, we denote by d^k the positive demand that must flow between the origin $O(k) \in N$ and the destination $D(k) \in N$. We associate a positive capacity u_{ij} to each arc $(i, j) \in A$ and assume that $u_{ij} \leq \sum_{k \in K} d^k$. A nonnegative fixed cost f_{ij} is charged when arc (i, j) is used. A nonnegative transportation cost c^k_{ij} has to be paid for each unit of commodity k moving through arc (i, j). The problem consists in minimizing the sum of all costs, while satisfying demand requirements and capacity constraints.

MCFP is NP-hard and is usually formulated as a mixed-integer programming (MIP) model (for surveys on complexity results and MIP techniques applied to MCFP, and other related network design problems, see Magnanti and Wong [14], Minoux [15], Balakrishnan, Magnanti and Mirchandani [1] and Gendron, Crainic and Frangioni [6]). In particular, some efforts have been devoted to design efficient solution techniques for MCFP based on Lagrangean relaxation [2, 4, 5, 6, 10]. To complement these Lagrangean bounding procedures, effective heuristics should be used to derive good feasible solutions. Crainic, Gendreau and Farvolden [3] propose a tabu search heuristic based on a path formulation of the problem, where simplex pivots define the neighborhoods, and new paths are dynamically added to the formulation using a column generation approach (we assume familiarity of the reader with the principles of tabu search; for further details, see Glover and Laguna [9]). Ghamlouche, Crainic and Gendreau [7] present a different tabu search heuristic based on an arc formulation of the problem, where the neighborhoods are obtained by moving flows around cycles. This approach has recently been improved by adding a path relinking search [8]. The resulting heuristic is currently the most effective for MCFP, since for a large set of randomly generated instances, it generally displays the smallest gap with respect to the optimal solution or, when the latter is unknown, it generally identifies the best known solution.

In this paper, we propose a different heuristic based on the idea of slope scaling (see Kim and Pardalos [11, 12, 13] for recent implementations of this idea for solving non-convex piecewise linear network flow problems and special cases, and Yaged [16] for an earlier approach based on similar ideas). The *Slope Scaling (SS)* procedure is an iterative scheme that consists in solving a linear approximation of the original formulation at each iteration. The costs of each linear approximation are adjusted in order to reflect the exact costs (both linear and fixed) incurred by the solution at the previous iteration. The iterations proceed until two successive solutions are identical. At this point, the linear approximation costs of the final solution correspond to the true objective function value, given by the sum of the original transportation and fixed costs. Although this procedure allows to identify fairly good solutions in a short amount of time, it might stop relatively far from an optimal solution. In order to improve its performance, we propose to combine it with two features, inspired from the literature on heuristics: Lagrangean perturbation and long-term memory.

The *Lagrangean perturbation (LP)* scheme assumes that the SS heuristic is performed concurrently with a Lagrangean bounding procedure, which provides dual values and reduced cost information. At every iteration of the SS procedure, the formula for computing the linear approximation costs is modified by taking into account the current values of the Lagrangean multipliers. The SS heuristic and the Lagrangean bounding procedure alternate at regular intervals (determined by a parameter), which allows to produce more variability in the process than, for example, performing SS iterations only after the Lagrangean bounding procedure has converged to an optimal value.

The resulting *Slope Scaling/Lagrangean perturbation (SS/LP)* heuristic explores more solutions, and generally identifies better ones, than the simple SS procedure. However, a careful examination of the solutions visited by the SS/LP heuristic reveals that the trajectories it produces remain around those obtained by the simple SS procedure (see Section 6 for computational evidence). As a result, even though the procedure usually continues to progress while the SS heuristic has already stopped, its final solution might also remain far from an optimal one for many instances. To improve its performance, we decompose the heuristic into multiple phases, each phase corresponding to one execution of the SS/LP procedure, which is stopped when no significant progress is obtained. Each phase starts with linear approximation costs modified by using a *long-term memory* that stores information gathered from all past iterations. We propose various intensification and diversification mechanisms based on this long-term memory. The experimental results demonstrate that the overall approach explores effectively the set of feasible solutions, particularly so for problem instances with large number of commodities. Indeed, on the largest and most difficult instances, the proposed heuristic is competitive with the tabu search/path relinking approach of Ghamlouche, Crainic and Gendreau [8], and it even identifies the best known solution for some instances.

To sum up, the main contribution of this work is to propose a new heuristic approach for MCFP that combines slope scaling, Lagrangean relaxation, and learning capabilities inspired by metaheuristics. It thus contributes to the emerging and promising field of hybrid heuristics that bring together mathematical programming approaches and metaheuristic methodologies.

The paper is organized as follows. Section 1 presents the mathematical formulation of MCFP, which is used throughout the paper to define the heuristic framework. In Section 2, we describe the SS procedure, while in Section 3, we provide the details of the SS/LP procedure. Section 4 is dedicated to the long-term memory approach and its intensification and diversification mechanisms. The overall procedure, which we denote *SS/LP/LM*, is then summarized in Section 5. Results of extensive experiments on a large set of randomly generated test problems with various characteristics are presented and analyzed in Section 6. We conclude the paper with a summary of its main contributions and a discussion of avenues for future research.

1 Formulation and Heuristic Framework

The arc formulation of MCFP uses two types of variables. First, nonnegative flow variables x_{ij}^k represent the flow of commodity $k \in K$ on arc $(i,j) \in A$. Second, binary design variables y_{ij} assume value 1 whenever arc $(i,j) \in A$ is used, and value 0 otherwise. The problem is then formulated as follows, where $b_{ij}^k = \min\{u_{ij}, d^k\}$, $\forall (i,j) \in A$, $k \in K$:

$$Z(MCFP) = \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij}, \tag{1}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{ji}^k = \begin{cases} d^k, & if\ i = O(k), \\ -\ d^k, & if\ i = D(k), \\ 0, & if\ i \neq O(k), D(k), \end{cases} \qquad \forall\, i \in N,\ k \in K, \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \quad \forall\ (i,j) \in A, \tag{3}$$

$$x_{ij}^k \leq b_{ij}^k y_{ij}, \quad \forall\ (i,j) \in A,\ k \in K, \tag{4}$$

$$x_{ij}^k \geq 0, \quad \forall\ (i,j) \in A,\ k \in K, \tag{5}$$

$$y_{ij} \in \{0,1\}, \quad \forall\ (i,j) \in A. \tag{6}$$

Equations (2) are the usual flow conservation constraints for multicommodity networks. The capacity constraints, (3), ensure that no flow circulates when an arc is not used. The same is achieved by relations (4), which are therefore redundant, but are added to the formulation in order to improve the quality of the Lagrangean relaxations derived from it (see Section 3, for further details).

The heuristic procedures we present are based on solving a succession of linear multicommodity minimum cost network flow problems, each defined by a vector of linearization factors ρ and denoted MMCF(ρ):

$$Z(MMCF(\rho)) = \min \sum_{k \in K} \sum_{(i,j) \in A} (c_{ij}^k + \rho_{ij}^k) x_{ij}^k, \tag{7}$$

subject to flow conservation constraints, (2), non negativity constraints, (5), and capacity constraints

$$\sum_{k \in K} x_{ij}^k \leq u_{ij}, \quad \forall\ (i,j) \in A. \tag{8}$$

When a feasible solution $\widetilde{x}$ to MMCF(ρ) is obtained, one can easily derive a feasible solution to MCFP by setting the design variables to

$$\widetilde{y}_{ij} = \begin{cases} 1, & if\ \sum_{k \in K} \widetilde{x}_{ij}^k > 0, \\ 0, & otherwise, \end{cases} \qquad \forall\, (i,j) \in A. \tag{9}$$

An upper bound on $Z(MCFP)$ is then computed as

$$Z(\tilde{x}, \tilde{y}) = \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k \tilde{x}_{ij}^k + \sum_{(i,j) \in A} f_{ij} \tilde{y}_{ij}. \tag{10}$$

The heuristics that we describe next only differ in the way they define ρ, the vector of linearization factors.

2 Slope Scaling (SS)

The SS procedure begins by solving a multicommodity minimum cost network flow problem, MMCF($\rho(0)$), with some initial value of $\rho = \rho(0)$. The approximation $\rho_{ij}^k(0) = f_{ij}/u_{ij}$, $\forall (i,j) \in A$, $k \in K$, provides an effective initial solution [11].

Let t denote the iteration counter. Given an optimal solution $\tilde{x}$ to problem MMCF($\rho(t)$), the next value $\rho(t+1)$ is determined so as to reflect the exact costs. More precisely, for any arc (i,j), if $\sum_{k \in K} \tilde{x}_{ij}^k > 0$, we must satisfy

$$\sum_{k \in K} (c_{ij}^k + \rho_{ij}^k(t+1)) \tilde{x}_{ij}^k = \sum_{k \in K} c_{ij}^k \tilde{x}_{ij}^k + f_{ij}. \tag{11}$$

On the other hand, when $\sum_{k \in K} \tilde{x}_{ij}^k = 0$, $\rho_{ij}^k(t+1)$ must equal some large value, for each $k \in K$. In this case, we propose to use the cost at the previous iteration, which was clearly large enough for that arc and commodity pair to incur no flow. In general, it is not desirable to set the cost to a very large value, because this would virtually forbid that particular arc-commodity pair to be subsequently used, thus reducing the set of solutions explored [11].

To satisfy these conditions, we use the following update of ρ at every iteration $t > 0$ (similar rules in a single-commodity setting are proposed in [11]):

$$\rho_{ij}^k(t) = \begin{cases} f_{ij} / \sum_{k \in K} \tilde{x}_{ij}^k, & \textit{if } \tilde{x}_{ij}^k > 0, \\ \rho_{ij}^k(t-1), & \textit{otherwise}, \end{cases} \quad \forall\, (i,j) \in A,\, k \in K. \tag{12}$$

Problem MMCF($\rho(t)$) is then solved again, and the two steps, adjustment of ρ and solution of the corresponding multicommodity minimum cost network flow problem, are repeated until a maximum number of iterations, $t_{\max} > 0$, is achieved. Note that Kim and Pardalos [11] propose to stop the procedure when two successive solutions are identical, which is indeed a sound stopping criterion. Since the SS heuristic is a special case of the overall SS/LP/LM approach presented in Section 5, we defer until then the discussion on other stopping rules. To summarize, the SS heuristic proceeds as follows, where Z^* is the best known upper bound on $Z(MCFP)$:

1. Set $Z^* = +\infty$, $t = 0$ and $\rho_{ij}^k(t) = f_{ij}/u_{ij}$ $\forall (i,j) \in A$, $k \in K$.
2. Solve MMCF($\rho(t)$); let $\widetilde{x}$ be the optimal solution and $Z(t) = Z(\widetilde{x}, \widetilde{y})$ the corresponding upper bound, computed using (9) and (10).
3. If $Z^* > Z(t)$, then $Z^* = Z(t)$.
4. $t = t + 1$.
5. If $t = t_{\max}$, stop the procedure.
6. Update $\rho(t)$ using formula (12), and go to step 2.

3 Lagrangean Perturbation (LP)

When the SS procedure identifies the same solution on two consecutive iterations, it makes no further progress, since ρ retains the same values. In order to allow more variability in the adjustment of ρ, we propose to perturb the linearization factors using dual information available from some relaxation of MCFP. More precisely, the SS iterations will alternate with some relaxation procedure that provides updated dual information, which is then used to adjust the linearization factors.

Denote by π, $\alpha \geq 0$ and $\beta \geq 0$ the vectors of dual variables associated to constraints (2), (3) and (4), respectively. The dual information is then given by:

$$\Delta_{ij}^k = \pi_i^k - \pi_j^k + \alpha_{ij} + \beta_{ij}^k, \quad \forall (i,j) \in A,\, k \in K. \tag{13}$$

Note that $c_{ij}^k + \Delta_{ij}^k$ corresponds to the reduced cost associated to x_{ij}^k in the standard linear programming (LP) relaxation of MCFP. To obtain this dual information, we use a Lagrangean relaxation procedure where the Lagrangean dual is optimized with a bundle method, an approach that has shown its superiority when compared to subgradient methods [2].

Two different Lagrangean relaxations are compared (see Section 6). The first is the *shortest path (SP) relaxation*, which consists in relaxing constraints (3) and (4), yielding the following Lagrangean subproblem:

$$Z(\alpha, \beta) = \min \sum_{(i,j)\in A} \sum_{k\in K} (c_{ij}^k + \alpha_{ij} + \beta_{ij}^k) x_{ij}^k \tag{14}$$

$$+ \sum_{(i,j)\in A} (f_{ij} - u_{ij}\alpha_{ij} - \sum_{k\in K} b_{ij}^k \beta_{ij}^k) y_{ij}, \tag{15}$$

subject to constraints (2), (5), and (6). This subproblem decomposes into a shortest path problem for each commodity, and a problem expressed with the design variables only, which is solvable by simple inspection of the signs of the costs. When using this relaxation, note that the current values of α and β are provided by the bundle algorithm, which is in charge of the adjustment of the Lagrangean multipliers, while the current values of π are given by the shortest path subproblem, i.e., π_i^k then represents the length of the shortest path (with respect to the Lagrangean costs) from $O(k)$ to i.

The second Lagrangean relaxation consists in relaxing the flow conservation constraints, (2), which yields the following Lagrangean subproblem:

$$Z(\pi) = \min \sum_{(i,j)\in A} \sum_{k\in K} (c_{ij}^k + \pi_i^k - \pi_j^k) x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} d^k (\pi_{D(k)}^k - \pi_{O(k)}^k), \qquad (16)$$

subject to constraints (3) to (6). This problem decomposes into $|A|$ subproblems, one for each arc, which can be easily solved by considering the two possible alternatives, $y_{ij} = 0$ and $y_{ij} = 1$, and by selecting the cheapest one. We note that if $y_{ij} = 0$, the problem is trivially solved (in this case, $x_{ij}^k = 0$, $\forall k \in K$), while if $y_{ij} = 1$, the problem reduces to a continuous knapsack problem, which is easily solvable by sorting the costs. Hence, we give the name *continuous knapsack (CK)* to this relaxation. Note that the current values of π are provided by the bundle method, while those of α and β are derived from the solution of the last Lagrangean subproblem.

At every iteration $t > 0$ of the slope scaling procedure, given the solution $\widetilde{x}$ computed at the previous iteration, we incorporate the dual information provided by any of these two relaxations using the following formula:

$$\rho_{ij}^k(t) = \begin{cases} \Delta_{ij}^k + (f_{ij} - \sum_{k\in K} \Delta_{ij}^k \widetilde{x}_{ij}^k) / \sum_{k\in K} \widetilde{x}_{ij}^k, & \textit{if } \widetilde{x}_{ij}^k > 0, \\ \rho_{ij}^k(t-1), & \textit{otherwise}, \end{cases} \qquad \forall\,(i,j) \in A,\ k \in K. \qquad (17)$$

It is easy to verify that this update of the linearization factors satisfies relation (11), i.e., it maintains the same interpretation of reflecting the exact costs if the solution to MMCF(ρ) is $\widetilde{x}$.

Two additional parameters are needed to characterize the SS procedure that incorporates Lagrangean perturbation, which we denote SS/LP. To decide when to update the dual information, we count the number of SS/LP iterations performed without improving the best upper bound found since the last update of the dual information. When this value is equal to $t_{\max}^{LP} > 0$, the Lagrangean bounding procedure is called to update the dual information. The rationale behind this rule is to use the current dual information as long as some progress is made, and to perturb the linearization factors only when it becomes clear that no further improvement can be performed. The second parameter, $t_{\max}^{BI}$, represents the number of bundle iterations (each corresponding to the solution of a Lagrangean subproblem) performed before returning to SS/LP iterations with updated dual information. Subsequently, every time the SS/LP procedure updates the dual values, the bundle method is restarted using the same information available to it the last time it was stopped. To summarize, the SS/LP procedure proceeds as follows:

1. Set $Z^* = +\infty$, $t = 0$ and $\rho_{ij}^k(t) = f_{ij}/u_{ij}$ $\forall (i,j) \in A$, $k \in K$.
2. Set $\Delta = 0$, $Z_{LP}^* = +\infty$ and $t^{LP} = 0$.
3. Solve MMCF($\rho(t)$); let $\widetilde{x}$ be the optimal solution and $Z(t)$ the corresponding upper bound.
4. If $Z^* > Z(t)$, then $Z^* = Z(t)$.
5. If $Z_{LP}^* = +\infty$, then $Z_{LP}^* = Z(t)$; go to step 7.
6. If $Z_{LP}^* > Z(t)$, then $Z_{LP}^* = Z(t)$ and $t^{LP} = 0$; otherwise, $t^{LP} = t^{LP} + 1$.
7. $t = t + 1$.
8. If $t = t_{\max}$, stop the procedure.
9. If $t^{LP} = t_{\max}^{LP}$, then set $Z_{LP}^* = +\infty$ and $t^{LP} = 0$, call the bundle method for $t_{\max}^{BI}$ iterations, and then update the dual information Δ according to (13).
10. Update $\rho(t)$ using formula (17), and go to step 3.

Note that by setting $t_{\max}^{LP}$ to $t_{\max}$, the SS/LP procedure simulates the behavior of the SS procedure, since then Δ is never updated and remains at value 0. In this case, formula (17) reduces to (12), and the two procedures are identical.

4 Long-Term Memory (LM)

Although the Lagrangean perturbation allows the heuristic to search the solution space more thoroughly than the simple SS procedure, the two variants, SS and SS/LP, tend to explore solutions that remain around the same regions of the solution space (Section 6 presents computational evidence). Hence, to further improve the performance of the heuristic, one needs additional mechanisms to intensify the search into more promising regions, and to diversify the search when it is believed that no significant progress can be made by looking further around the most recently visited regions (see [9] for additional explanations on the notions of intensification and diversification).

The mechanisms we propose are based on a long-term memory which stores informations gathered during the whole history of the search. More specifically, at iteration $t \geq 0$, the memory contains three informations for each triplet (i,j,k), where $(i,j) \in A$ and $k \in K$: n_{ij}^k, the number of iterations where $\widetilde{x}_{ij}^k > 0$; $\overline{x}_{ij}^k$, the average flow, defined as $\sum_{0 \leq T \leq t} \widetilde{x}_{ij}^k(T)/(t+1)$, where $\widetilde{x}(T)$ is the solution of MMCF($\rho(T)$); and $\widehat{x}_{ij}^k$, the maximum flow defined as $\max_{0 \leq T \leq t} \widetilde{x}_{ij}^k(T)$.

If the objective is to perform intensification, the corresponding mechanism attempts to make more "interesting" the triplets (i, j, k) that attract flow frequently (measured by n_{ij}^k) and to make less "interesting" the triplets that are rarely used. Conversely, the diversification mechanism attempts to make less "interesting" the triplets that are frequently used and to make more "interesting" the triplets that rarely attract flow. To make a given triplet more or less "interesting", the linearization factors are modified using the ratio $v_{ij}^k = \overline{x}_{ij}^k / \widehat{x}_{ij}^k$, which measures the "variability" of the flow activity related to triplet (i, j, k) (to complete the definition, we set $v_{ij}^k = 0$, when $\widehat{x}_{ij}^k = 0$): if v_{ij}^k approaches 1, the amounts of flow attracted by (i, j, k) show small variations along the iterations, while if v_{ij}^k approaches 0, the amounts of flow attracted by (i, j, k) vary significantly along the iterations. When performing intensification, we favor the triplets that show less "variability" and penalize those with more "variability". On the contrary, when performing diversification, we favor the triplets with more "variability" and penalize those with less "variability".

When performing an intensification or a diversification, the linearization factors are modified in two ways. First, since the update of the linearization factors given by (17) implies that some may have a negative value (because π is unrestricted in sign), they are normalized in such a way that $\rho(t) \geq 0$: if $\rho_{\min}(t) = \min\{0, \min_{(i,j,k)} \rho_{ij}^k(t)\}$, we set $\rho_{ij}^k(t) = \rho_{ij}^k(t) - \rho_{\min}(t)$, $\forall (i, j) \in A, k \in K$. Second, if we want to make more "interesting" a given triplet (i, j, k), we decrease its linearization factor by multiplying it with some value dependent of v_{ij}^k and smaller than 1 (this is why we need to ensure first that $\rho \geq 0$). Conversely, if we want to make less "interesting" a given triplet (i, j, k), we increase its linearization factor by multiplying it with some value dependent of v_{ij}^k and larger than 1.

More precisely, when performing an intensification step, we apply the following rules:

1. Normalize ρ to satisfy $\rho \geq 0$.
2. If $n_{ij}^k \geq \delta^+$, then
$$\rho_{ij}^k = \rho_{ij}^k (1 - v_{ij}^k), \quad (i, j) \in A,\ k \in K. \tag{18}$$
3. If $n_{ij}^k < \delta^-$, then
$$\rho_{ij}^k = \rho_{ij}^k (2 - v_{ij}^k), \quad (i, j) \in A,\ k \in K. \tag{19}$$

Conversely, when performing a diversification step, we apply the following rules:

1. Normalize ρ to satisfy $\rho \geq 0$.
2. If $n_{ij}^k \geq \delta^+$, then
$$\rho_{ij}^k = \rho_{ij}^k (1 + v_{ij}^k), \quad (i, j) \in A,\ k \in K. \tag{20}$$

3. If $n_{ij}^k < \delta^-$, then

$$\rho_{ij}^k = \rho_{ij}^k(v_{ij}^k), \quad (i,j) \in A,\, k \in K. \tag{21}$$

The parameters δ^+ and δ^- measure the triplets that frequently and rarely attract flow, respectively. We compute them as follows: $\delta^+ = \overline{n} + \omega^+ s_n$ and $\delta^- = \overline{n} - \omega^- s_n$, where $\overline{n}$ and s_n are, respectively, the average and standard deviation measures for n_{ij}^k, $\forall (i,j) \in A$, $k \in K$, while ω^+ and ω^- are parameters. Typically, most triplets (i,j,k) are very rarely used, so $\overline{n}$ is close to 0. Consequently, setting ω^- to a significant positive value would imply a negative value of δ^-, and in this case, no triplet would be identified as rarely used. This is why we use $\omega^- = 0$ in all tests reported in Section 6. Similarly, we should not use too large values for ω^+, but typically values in the interval $[0,1]$ show good performance.

The decisions related to when to perturb the linearization factors according to the informations stored in the long-term memory and whether to perform an intensification or a diversification are explained in the next section.

5 Overall Procedure (SS/LP/LM)

The overall procedure organizes the computations into multiple phases, where each phase corresponds to the execution of several iterations of procedure SS/LP, until some stopping criteria are met. These stopping criteria are based on storing the value of the best overall solution found during the current phase, Z^*_{local}. Two parameters are used: $t^>_{\max}$, the maximum number of consecutive iterations without improving Z^*_{local} ($Z(t) > Z^*_{local}$ for $t^>_{\max}$ consecutive iterations); $t^=_{\max}$, the maximum number of consecutive iterations without modifying $Z(t)$ ($Z(t) = Z(t-1)$ for $t^=_{\max}$ consecutive iterations).

Once a phase has stopped based on these criteria, we perform an LM perturbation of the linearization factors. The decision as to whether use an intensification or a diversification depends on comparing the value of the best solution found during that phase, Z^*_{local}, to the value of the best solution found prior to the current phase, Z^*. We apply intensification if $Z^*_{local} < Z^*$ and diversification otherwise. The rationale is to apply intensification when the best solution at the current phase improves upon Z^*. If for several consecutive phases, the condition $Z^*_{local} < Z^*$ is satisfied, no diversification will take place. Similarly, if $Z^*_{local} \geq Z^*$ for several successive phases, no intensification will take place. To allow for both types of perturbation, intensification and diversification, to happen at regular intervals, we limit the number of consecutive applications of intensification or diversification, using the parameters $T^{inten}_{\max}$ and $T^{diver}_{\max}$. Finally, we apply intensification or diversification on the linearization factors corresponding to the best solution found during the current phase.

At the end of each phase, it is possible to improve Z^*_{local} by applying the following local improvement step. Given the solution $(\widetilde{x}, \widetilde{y})$ corresponding to Z^*_{local}, one can set the linearization factors corresponding to "closed" arcs (with $\widetilde{y}_{ij} = 0$) to very high values (thus forbidding the flows to pass through these arcs) and to 0 the linearization factors corresponding to "open" arcs (with $\widetilde{y}_{ij} = 1$). We then solve the corresponding MMCF(ρ), which provides a solution that is optimal with respect to the original transportation costs, given the arc configuration defined by $\widetilde{y}$. If the new solution obtained improves upon Z^*_{local}, which is often the case, it replaces it. This local improvement step can be seen as a form of intensification.

The overall procedure can be summarized as follows:

1. Set $Z^* = +\infty$, $t = 0$, $T^{inten} = 0$, $T^{diver} = 0$ and $\rho^k_{ij}(t) = f_{ij}/u_{ij}$ $\forall (i,j) \in A$, $k \in K$.
2. Set $\Delta = 0$, $Z^*_{LP} = +\infty$ and $t^{LP} = 0$.
3. Set $Z^*_{local} = +\infty$, $t^{>} = 0$ and $t^{=} = 1$.
4. Solve MMCF($\rho(t)$); let $\widetilde{x}$ be the optimal solution and $Z(t)$ the corresponding upper bound.
5. If $Z^*_{local} > Z(t)$, then $Z^*_{local} = Z(t)$ and $t^{>} = 0$; otherwise, $t^{>} = t^{>} + 1$.
6. If $Z^*_{LP} = +\infty$, then $Z^*_{LP} = Z(t)$; go to step 8.
7. If $Z^*_{LP} > Z(t)$, then $Z^*_{LP} = Z(t)$ and $t^{LP} = 0$; otherwise, $t^{LP} = t^{LP} + 1$.
8. $t = t + 1$.
9. If $t = t_{\max}$, go to step 16.
10. If $Z(t) = Z(t-1)$, then $t^{=} = t^{=} + 1$; otherwise, $t^{=} = 1$.
11. If $t^{LP} = t^{LP}_{\max}$, then $Z^*_{LP} = +\infty$ and $t^{LP} = 0$, call the bundle method for $t^{BI}_{\max}$ iterations, and then update the dual information Δ according to (13).
12. Update $\rho(t)$ using formula (17).
13. If $t^{>} < t^{>}_{\max}$ and $t^{=} < t^{=}_{\max}$, go to step 4.
14. Perform local improvement.
15. If ($Z^*_{local} < Z^*$ or $T^{diver} \geq T^{diver}_{\max}$) and $T^{inten} < T^{inten}_{max}$, perform intensification, set $T^{inten} = T^{inten} + 1$, and if $T^{diver} \geq T^{diver}_{\max}$, set $T_{diver} = 0$; otherwise, perform diversification, set $T^{diver} = T^{diver} + 1$, and if $T^{inten} \geq T^{inten}_{\max}$, set $T_{inten} = 0$.
16. If $Z^* > Z^*_{local}$, then $Z^* = Z^*_{local}$.

17. If $t = t_{\max}$, stop the procedure.

18. Go to step 3.

It is worth noting that by appropriately setting the parameters, the procedure can simulate the behavior of the SS/LP procedure. Indeed, when $t^{>}_{\max} = t^{=}_{\max} = t_{\max}$, only one phase is performed, which corresponds to the execution of procedure SS/LP. Since the latter can also be used to simulate the behavior of the SS procedure, we will use only an implementation of the SS/LP/LM procedure to measure the performance of the three procedures in the next section.

6 Computational Experiments

The SS/LP/LM procedure has been implemented in C++, using CPLEX (version 6.6) to solve the linear multicommodity minimum cost network flow problem at each iteration and during the local improvement step. The optimal basis at the last iteration is used to initialize the dual simplex algorithm of CPLEX. Although probably not the most efficient code for solving multicommodity minimum cost network flow problems, CPLEX provides a robust environment to test the effectiveness of the SS/LP/LM procedure and to investigate its ability to identify good solutions using a "reasonable" computing effort. To define such a "reasonable" effort, we limit the number of iterations to $t_{\max} = 400$, which corresponds roughly to the number of multicommodity minimum cost network flow problems solved by the tabu cycle/path relinking method [8], in the tests reported by the authors. Since they used the same set of instances as ours, this will allow for a fair comparison between the two approaches.

The experiments were performed on a Sun Enterprise 10000 with 64 CPUs (with each CPU operating at 450 MHz) and 64 GBs of RAM memory. The code is compiled with the g++ compiler using the -O option. The initial multiplier adjustment and parameter setting of the bundle method for each of the two tested relaxations (SP and CK) are documented in [2].

We have run our tests on 196 problem instances obtained from a network generator similar to the one described in [4, 5]. When provided with target values for $|N|$, $|A|$, and $|K|$, this generator creates arcs by connecting two randomly selected nodes (no parallel arcs are allowed). It proceeds similarly to create commodities. Costs, capacities, and demands are then generated, uniformly distributed over user-provided intervals. Capacities and costs can then be scaled to obtain networks with various degrees of capacity tightness and relative importance of fixed costs. Two ratios are used for this purpose: the *capacity ratio* $C = |A|T / \sum_{(i,j)\in A} u_{ij}$ and the *fixed cost ratio* $F = |K| \sum_{(i,j)\in A} f_{ij} / T \sum_{k\in K} \sum_{(i,j)\in A} c^k_{ij}$, where $T = \sum_{k\in K} d^k$. Capacities and fixed costs

Class I	(31)	Class II	(12)	Class III			(153)
20,230,40	(3)	25,100,10	(3)	10,35,10	(6)	20,120,40	(9)
20,300,40	(4)	100,400,10	(3)	10,60,10	(6)	20,220,40	(9)
30,520,100	(4)	25,100,30	(3)	10,85,10	(6)	20,320,40	(9)
30,700,100	(4)	100,400,30	(3)	10,35,25	(9)	20,120,100	(9)
20,230,200	(4)			10,60,25	(9)	20,220,100	(9)
20,300,200	(4)			10,85,25	(9)	20,320,100	(9)
30,520,400	(4)			10,35,50	(9)	20,120,200	(9)
30,700,400	(4)			10,60,50	(9)	20,220,200	(9)
				10,85,50	(9)	20,320,200	(9)

Table 1: Classification of Instances According to Problem Dimension

are adjusted so that these ratios come close to user-provided values. In general, when C approaches 1, the network is lightly capacitated and becomes more congested as C increases. When F is close to 0, the fixed costs are low compared to the transportation costs, while their relative importance increases with F.

The instances are divided into three classes. Class I consists of instances with a large number of commodities, especially relative to the number of nodes, while Class II is made of instances with number of nodes approximately equal or larger than the (small) number of commodities. Class III instances have been specifically generated to make problem characteristic versus performance analyses easier. Nine networks are generated in each subclass, by combining three arc-densities, roughly 25%, 50%, and 75%, with three commodity-densities, roughly 10%, 25%, and 50% (the density is the ratio with respect to $|N||N-1|$). For each of these nine networks, nine problem instances are created by combining three values of F – 0.01, 0.05, and 0.1 – with three values of C: 1, 2 and 8. Several problem instances were also created for each network in classes I and II to represent various F and C ratio values. Table 1 summarizes the characteristics of the 196 problem instances in the three classes according to problem dimension represented by a triplet $|N|$,$|A|$,$|K|$. The number of instances is displayed between parentheses (infeasible problem instances have been discarded).

The next subsection presents the result analysis of the SS/LP/LM procedure and its variants on the three classes of instances, while the second subsection is dedicated to comparative analyses with a state-of-the-art commercial solver and several tabu search-based metaheuristics:

- *CPLEX*: This is the branch-and-bound method implemented in CPLEX (version 7.1), used to solve the MIP formulation presented in Section 1. The experiments were performed on the Sun Enterprise 10000 for a maximum of 10 hours of CPU time. Although this is sufficient to identify the optimal solution for many instances, only feasible solutions are obtained for some other instances, while no feasible solutions are found for a few instances.

- *Tabu Path*: This is the tabu search heuristic described in [3]. Although this paper does not provide the detailed results on each of the 196 instances, we have obtained them from the authors. Results were obtained on SUN UltraSparc 1/140 Workstations with 64 MB of RAM memory. In this approach, there is no separate resolution of multicommodity minimum cost network flow subproblems.

- *Tabu Cycle*: This is the tabu search-based heuristic described in [7]. The detailed results are derived from [8] where the approach is compared to the path relinking method. The multicommodity minimum cost network flow problems are solved with CPLEX (version 6.5). The machine used is the Sun Enterprise 10000, as in our experiments, and the stopping criterion is a maximum of 400 iterations (which roughly corresponds to the solution of the same number of multicommodity minimum cost network flow problems).

- *Path Relinking*: This is the path relinking method described in [8], which extends the tabu cycle approach (by adding some diversification capabilities), and as such, exhibits similar implementation characteristics.

6.1 Analysis of SS/LP/LM and Variants

We tested and compared the SS/LP/LM procedure and its variants, and analyzed the impact of a number of key parameters. In the following, we summarize the main findings and present aggregated results that support them (detailed results can be obtained from the authors).

For this phase of the computational experiments, we use as a performance measure the GAP $(Z^* - Z_{best})/Z_{best}$ between the upper bound Z^* obtained by the procedure and the best available upper bound Z_{best}. The latter corresponds to the smallest upper bound identified by the four methods *CPLEX*, *Tabu Path*, *Tabu Cycle*, and *Path Relinking*. The aggregated results are obtained through averaging the performance measure over all instances in a class. Note that, although averaging over such large sets of instances with so varied characteristics might appear coarse, it proves remarkably reliable, as more detailed instance-by-instance analyses have revealed similar results and tendencies.

We first study the impact of the Lagrangean perturbation approach. We compare the pure slope scaling procedure (SS) to the slope scaling procedure with Lagrangean perturbation (SS/LP) using either the shortest path (SP) or the continuous knapsack (CK) relaxation method. The parameters of the SS/LP procedure assume values $t_{\max}^{LP} = 10$ and $t_{\max}^{BI} = 10$. Other values were tried, but the results did not vary significantly. Table 2 displays the average GAP (%) obtained by the three approaches. On each problem

	SS	SS/LP (SP)	SS/LP (CK)
20,230,40 (3)	1.02	1.09	1.31
20,300,40 (4)	1.57	1.58	1.59
30,520,100 (4)	11.03	11.49	11.90
30,700,100 (4)	10.57	11.39	12.12
20,230,200 (4)	15.04	13.22	13.68
20,300,200 (4)	6.45	4.94	6.10
30,520,400 (4)	13.37	12.90	11.70
30,700,400 (4)	7.68	5.44	7.16
Average (Class I)	8.58	7.97	8.42
25,100,10 (3)	6.70	6.37	6.37
100,400,10 (3)	15.98	16.81	14.38
25,100,30 (3)	3.06	3.58	3.07
100,400,30 (3)	8.32	7.68	7.67
Average (Class II)	8.52	8.61	7.87
10,35,10 (6)	0.92	0.91	0.91
10,60,10 (9)	2.34	2.34	2.34
10,85,10 (9)	2.50	2.34	2.23
10,35,25 (6)	0.66	0.57	0.63
10,60,25 (9)	4.76	4.91	5.34
10,85,25 (9)	7.88	6.42	7.89
10,35,50 (6)	0.75	0.54	0.54
10,60,50 (9)	6.80	6.62	5.65
10,85,50 (9)	9.59	9.02	9.02
20,120,40 (9)	6.47	6.38	6.63
20,220,40 (9)	15.43	16.65	15.29
20,320,40 (9)	11.93	14.60	13.08
20,120,100 (9)	3.56	3.39	3.51
20,220,100 (9)	10.99	10.94	11.17
20,320,100 (9)	17.74	16.83	17.66
20,120,200 (9)	2.69	2.65	2.65
20,220,200 (9)	6.34	6.62	5.81
20,320,200 (9)	8.83	9.96	9.20
Average (Class III)	7.02	7.12	6.99

Table 2: Comparison of Lagrangean Perturbation Methods (GAP %)

class, at least one of the variant of the SS/LP procedure generally slightly improves, on average, over the SS procedure. On Class I, the SP variant is generally superior to the CK relaxation, while the reverse is observed on the two other classes. A detailed instance-by-instance analysis also reveals that the SS procedure and the best variant of SS/LP for each problem class found the same solution for 34% of instances. For the remaining instances, the SS/LP procedure finds better solutions than the SS procedure for 60% of them.

Since the Lagrangean perturbation only marginally improves upon the pure slope scaling approach, one might ask whether the long-term memory perturbation would introduce enough diversification, so that the effect of the Lagrangean perturbation would be further diminished. Indeed, when the SS/LP/LM procedure is performed (see below), with and without Lagrangean perturbation, the two approaches find the same solution for 48% of the instances. Yet, for the remaining instances, the Lagrangean perturbation identifies better solutions than the pure slope scaling approach for 56% of them. Given the slight superiority of the Lagrangean perturbation, we have decided to use it for the remaining tests, the SP relaxation being used for Class I instances, and the CK one for the other classes of instances. It should be noted, however, that the results would not vary significantly if the Lagrangean perturbation were not be used.

We now turn to the long-term memory and examine the parameters that impact on the performance of the global procedure. Four parameters determine the procedure: $T_{\max}^{inten}$ and $T_{\max}^{diver}$, indicating the maximum number of consecutive applications of intensification and diversification steps, respectively; ω^- and ω^+, defining the triplets that frequently and rarely attract flow, respectively (see Section 4).

Setting $T_{\max}^{inten} = t_{\max}$ and $T_{\max}^{diver} = 0$ implies that the procedure performs only intensification. Symmetrically, $T_{\max}^{inten} = 0$ and $T_{\max}^{diver} = t_{\max}$ forces the procedure to perform diversification steps only. In the metaheuristic literature, smaller number of repetitions and more balanced approaches are generally used. We observed, in fact, that setting either $T_{\max}^{inten}$ or $T_{\max}^{diver}$ to values greater than 2 implies very long intensification or diversification phases that, in practice, forbid the other one from being executed. A (1,1) value has the procedure pass from one phase to the other irrespective whether the current phase is improving the solution or not. A more balanced approach is offered by setting the two parameters at 2: an improving trend will not be immediately interrupted, but there is sufficient time to execute both phases.

Turning to the ω parameters, as indicated in Section 4, ω^- must be set to 0 and values for ω^+ should not be too large. Actually, values in the $[0, 1]$ interval show good performance, combinations with $\omega^+ = 1$ usually displaying better results than other settings. Yet the case $\omega^+ = 0$ is also interesting, since in this case all costs are modified at every intensification or diversification step. Table 3 displays the results for the four sets of parameters that best illustrate the impact of the intensification/diversification alternation

	(400,0,0,1)	(0,400,0,1)	(2,2,0,1)	(2,2,0,0)
	SS/LP(SP)/LM			
20,230,40(3)	0.86	0.70	0.84	0.84
20,300,40(4)	1.07	1.07	1.07	1.07
30,520,100(4)	7.14	8.57	7.23	7.74
30,700,100(4)	7.83	8.55	7.60	8.40
20,230,200(4)	7.65	7.50	7.37	7.79
20,300,200(4)	6.54	9.08	5.82	6.15
30,520,400(4)	2.64	2.66	2.51	2.84
30,700,400(4)	-2.03	13.06	-2.03	-2.11
Average (Class I)	4.06	6.58	3.89	4.19
	SS/LP(CK)/LM			
25,100,10(3)	5.37	5.61	3.28	3.17
100,400,10(3)	11.04	12.34	10.37	7.78
25,100,30(3)	2.46	1.93	1.99	1.43
100,400,30(3)	6.33	4.54	4.29	4.56
Average (Class II)	6.30	6.10	4.98	4.24
	SS/LP(CK)/LM			
10,35,10 (6)	0.59	0.23	0.41	0.41
10,60,10 (9)	1.64	0.75	0.89	0.54
10,85,10 (9)	1.54	1.79	1.58	1.46
10,35,25 (6)	0.53	0.03	0.03	0.03
10,60,25 (9)	3.25	1.05	3.05	3.12
10,85,25 (9)	4.81	2.67	3.34	3.39
10,35,50 (6)	0.50	0.14	0.25	0.42
10,60,50 (9)	3.24	2.34	1.42	1.61
10,85,50 (9)	5.21	4.34	3.65	3.62
20,120,40 (9)	4.80	4.10	3.74	3.50
20,220,40 (9)	9.26	11.37	9.03	9.45
20,320,40 (9)	9.01	10.03	8.36	8.10
20,120,100(9)	2.40	1.95	2.11	2.08
20,220,100(9)	5.07	5.06	4.77	4.86
20,320,100(9)	9.62	16.15	7.92	9.04
20,120,200(9)	1.98	1.81	1.86	1.96
20,220,200(9)	2.48	2.93	2.63	2.94
20,320,200(9)	2.33	17.49	2.23	2.21
Average (Class III)	3.98	4.95	3.36	3.44

Table 3: Analysis of Long-Term Memory Utilization (GAP %)

strategy, where each parameter set is represented by a quadruplet $(T_{\max}^{inten}, T_{\max}^{diver}, \omega^-, \omega^+)$.

From the results displayed in the table, it appears that performing only diversification is detrimental to the performance of the method for the instances in Class I and Class III (surprisingly, on the instances of Class II, this combination is, on average, slightly better than performing intensification only). It is clear, however, that a combined and balanced utilization of intensification and diversification (third parameter setting) outperforms extreme procedure designs (intensification or diversification only). This observation is consistent with the state-of-the-art in the metaheuristic literature (see for example [9]). In the last column, we show the results obtained with a combination similar to the third one, but with $\omega^+ = 0$ instead of $\omega^+ = 1$. The results of this combination are, on average, worse than those observed with the third setting, except for the instances of Class II, which confirms that a high degree of diversification is indicated for these instances.

It should also be noted that the results in Table 3 are significantly better than those displayed in Table 2. This indicates that the long-term memory and the intensification and diversification mechanisms added to the procedure are instrumental in achieving high performance for this type of heuristic. Based on these results, we have used the long-term memory variant in the remaining experiments, that is, the SS/LP/LM procedure with the third parameter setting, (2,2,0,1), for Class I and Class III instances, and the fourth parameter setting, (2,2,0,0), for Class II instances.

To complete the analysis, we include two graphs displaying the evolution of the bound over time for the three variants, SS, SS/LP, and SS/LP/LM, for a relatively easy problem (100, 400, 10), Figure 1, and probably the most difficult instance in the whole set (30, 700, 400), Figure 2. For each graph, the x axis gives the iteration count, while the y axis gives the solution value. Circles indicate solutions found by the local improvement step. In the first case, easy problem (Figure 1), *Path Relinking* gives the best solution, and SS/LP slightly improves upon SS. Intensification and diversification mechanisms work well: the local improvement step slightly improves the bound, and the long-term memory perturbation drives the search to find new, better solutions, even though diversification initially deteriorates the solution value significantly. For the difficult problem, Figure 2, *Path Relinking* is still better than SS and SS/LP, but the full hybrid SS/LP/LM procedure yields the best solution. Again, SS/LP is slightly better than SS. The long-term memory mechanisms perform well, and the local improvement step significantly improves the bound. These examples, typical of what might be observed over the entire problem set, emphasize the central role of the long-term memory, and of the intensification and diversification mechanisms, in establishing the good performance of the method.

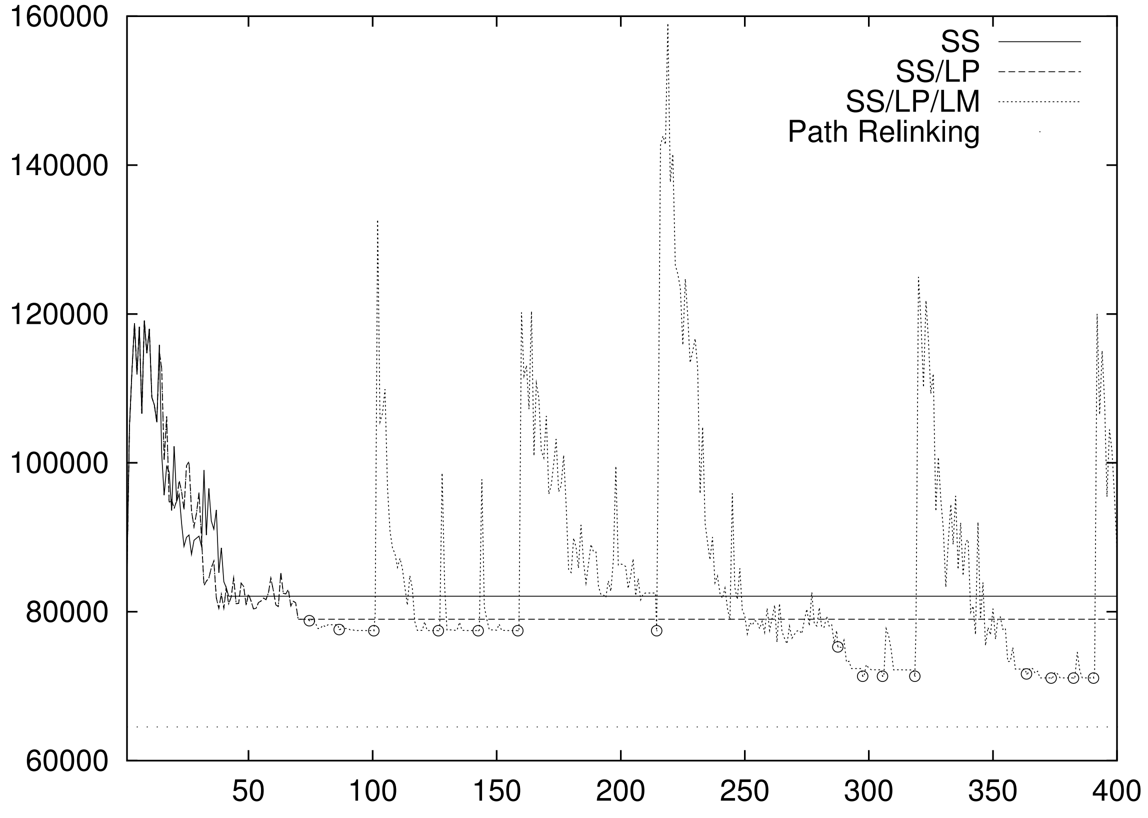

Figure 1: Problem (100, 400, 10)

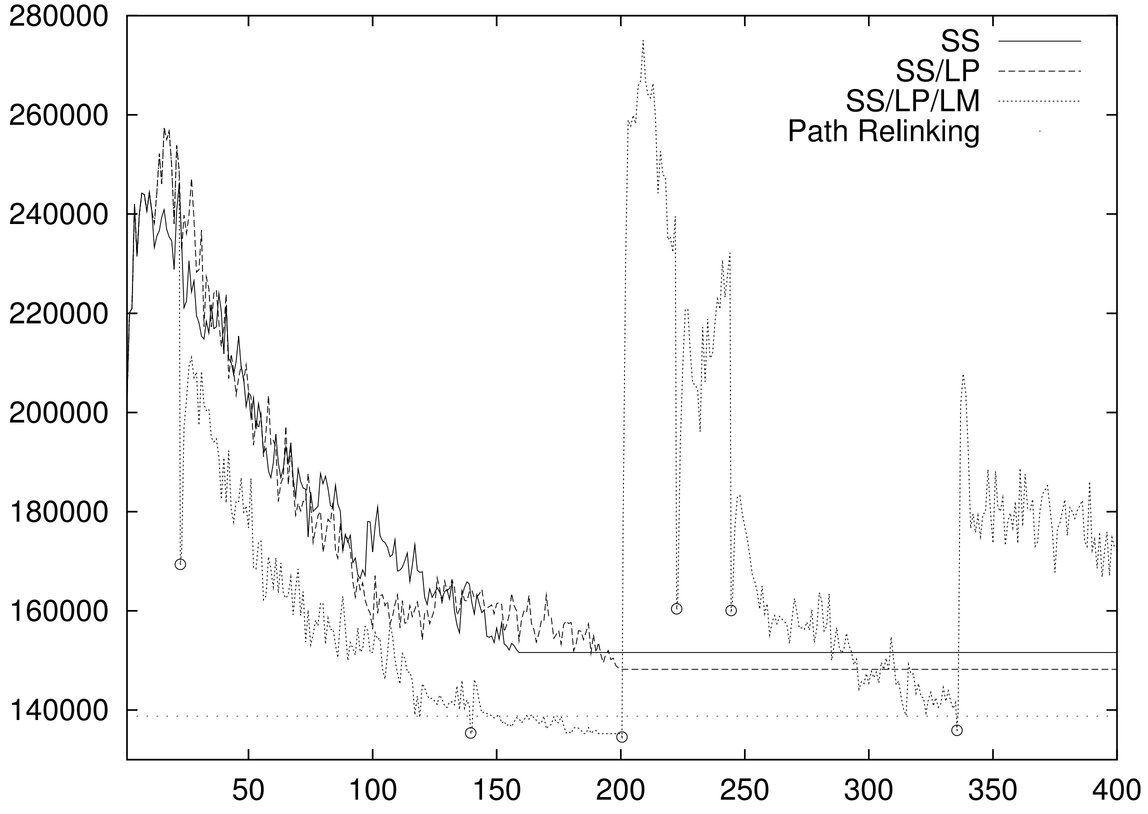

Figure 2: Problem (30, 700, 400)

6.2 Comparisons with Other Methods

We now compare the results of the SS/LP/LM procedure with those obtained by the four other methods, *CPLEX*, *Tabu Path*, *Tabu Cycle* and *Path Relinking*. In Table 4, we display two measures for each of these four methods: 1) GAPSS, the gap $(Z_{SS} - Z^*)/Z_{SS}$ between the upper bound Z^* obtained by the respective method and the upper bound Z_{SS} found by the SS/LP/LM procedure (thus, a negative value indicates that the SS/LP/LM procedure has identified a better solution than that found by the method; the result is an average over all problem instances in each class); 2) In between parentheses, *NBest*, the number of instances where the SS/LP/LM procedure has identified a better solution than the corresponding method.

We observe that the SS/LP/LM procedure provides solutions that are, on average, within 5% of optimality, as indicated by the gap with respect to *CPLEX*. It is also competitive with the other heuristics. It generally outperforms *Tabu Path*, and displays solutions that are close, or even better, than those found by *Tabu Cycle* and *Path Relinking*. Since the latter is the most effective of the three competing heuristic methods, we now focus on the comparison with this approach.

Path Relinking remains, on average, the most effective heuristic. We note, however, that SS/LP/LM generally obtains better solutions for instances with a large number of commodities (200 and more): in Class I, SS/LP/LM finds better solutions for 9 out of 16 instances, while in Class III, SS/LP/LM identifies better solutions for 21 out of 27 instances. A detailed instance-by-instance analysis also reveals that, for the 20-node instances in Class III with the highest fixed cost ratio ($F = 0.1$), SS/LP/LM obtains better solutions for 16 out of 27 instances. These results indicate that SS/LP/LM is generally more effective than *Path Relinking* on larger, more difficult, instances. For seven of these difficult instances, SS/LP/LM even identifies the best known solution, outperforming not only *Path Relinking*, but also the branch-and-bound method of *CPLEX* executed for 10 hours of CPU time (note that for the four Class I instances of dimension (30, 700, 400), *CPLEX* could not identify a feasible solution, hence we display an average gap of $-\infty$ for this subclass, and we do not consider these instances when computing the average gap for Class I). These results emphasize that the SS/LP/LM heuristic is not only competitive with the current best heuristics for MCFP, but also that it might become the algorithm of choice for problems with a large number of commodities.

7 Conclusion

We have presented a slope scaling heuristic for solving the multicomodity capacitated fixed-charge network design problem. The heuristic integrates a Lagrangean perturba-

	CPLEX	*Tabu Path*	*Tabu Cycle*	*Path Relinking*
20,230,40 (3)	0.83 (0)	0.64 (0)	0.66 (0)	0.71 (0)
20,300,40 (4)	1.05 (0)	0.80 (0)	0.56 (1)	0.72 (0)
30,520,100 (4)	6.64 (0)	0.70 (2)	3.26 (0)	3.36 (0)
30,700,100 (4)	7.01 (0)	2.38 (0)	4.77 (0)	5.36 (0)
20,230,200 (4)	6.78 (0)	-20.54 (4)	0.78 (2)	1.56 (1)
20,300,200 (4)	5.43 (0)	-14.21 (4)	-0.34 (3)	1.30 (1)
30,520,400 (4)	3.41 (0)	-7.06 (4)	-1.63 (3)	0.16 (3)
30,700,400 (4)	$-\infty$(4)	-10.79 (4)	-4.74 (4)	-2.08 (4)
Average (Class I)	4.68 (4)	-6.22 (18)	0.40 (13)	1.41 (9)
25,100,10 (3)	3.01 (0)	-0.11 (2)	2.55 (0)	3.01 (0)
100,400,10 (3)	5.59 (0)	2.70 (2)	5.59 (1)	6.61 (1)
25,100,30 (3)	1.40 (0)	-0.49 (1)	-0.70 (2)	0.37 (2)
100,400,30 (3)	4.21 (1)	-2.61 (3)	2.95 (1)	3.78 (0)
Average (Class II)	3.55 (1)	-0.13 (8)	2.60 (4)	3.44 (3)
10,35,10 (6)	0.41 (0)	-0.21 (2)	0.41 (0)	0.41 (0)
10,60,10 (9)	0.87 (0)	-0.9 (5)	0.65 (1)	0.79 (1)
10,85,10 (9)	1.53 (0)	-0.47 (4)	0.85 (1)	1.49 (0)
10,35,25 (6)	0.03 (0)	-0.54 (4)	-1.38 (4)	-0.21 (2)
10,60,25 (9)	2.82 (0)	0.18 (6)	2.04 (2)	2.47 (1)
10,85,25 (9)	3.19 (0)	-0.67 (5)	1.34 (2)	2.80 (0)
20,120,40 (9)	3.56 (0)	-1.17 (6)	1.18 (2)	2.26 (0)
20,220,40 (9)	8.01 (0)	0.48 (3)	4.05 (2)	4.80 (1)
20,320,40 (9)	7.59 (0)	1.72 (2)	4.02 (1)	5.69 (0)
10,35,50 (6)	0.25 (0)	-0.28 (2)	-1.22 (5)	-0.35 (5)
10,60,50 (9)	1.38 (0)	-6.16 (7)	-0.92 (7)	0.26 (5)
10,85,50 (9)	3.47 (0)	-5.12 (6)	0.89 (2)	2.02 (1)
20,120,100 (9)	2.01 (0)	-6.68 (9)	-0.86 (7)	0.05 (4)
20,220,100 (9)	4.47 (0)	-11.61 (7)	-0.67 (5)	-0.25 (5)
20,320,100 (9)	7.21 (0)	-11.94 (8)	0.72 (4)	2.86 (3)
20,120,200 (9)	1.80 (0)	-6.02 (9)	-2.85 (9)	-2.60 (8)
20,220,200 (9)	2.53 (0)	-21.10 (9)	-4.05 (9)	-2.74 (7)
20,320,200 (9)	2.01 (3)	-20.88 (9)	-5.22 (7)	-4.71 (6)
Average (Class III)	3.11 (3)	-5.35 (103)	-0.02 (70)	0.89 (49)

Table 4: Comparison with Other Methods (GAPSS % (*NBest*))

tion scheme and intensification/diversification mechanisms based on long-term memories. Computational experiments performed on a large set of problem instances have shown that the intensification and diversification components of the algorithm are essential for the approach to be effective. This emphasizes the gains that may be achieved in combinatorial optimization by bringing together mathematical programming and metaheuristic elements into comprehensive solution algorithms.

The computational experiments have also demonstrated that the proposed method is competitive with the best known heuristic approaches for the problem, and that it generally provides better solutions on larger, more difficult, instances. It even identifies the best known solution for some instances, providing solutions that are better than those obtained by *CPLEX* after 10 hours of CPU time.

Several fascinating research avenues are open following this work. In particular, improvements in the efficiency of the procedure could be achieved by substituting the general-purpose solver (CPLEX) with an adaptation of a decomposition method to solve the multicommodity minimum cost network flow problems at each iteration (e.g., a bundle method, a resource-decomposition approach, or a column generation algorithm). This might impact not only on the efficiency of the procedure, but also on its effectiveness. Also, since the SS/LP/LM and *Path Relinking* approaches appear complementary, a hybrid algorithm that integrates them both seems indicated. Its implementation in a parallel environment opens the way for several challenging, but promising research issues.

Acknowledgments

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Research Grant programs, and by the Fonds F.C.A.R. of the Province of Québec. We also thank the RQCHP - Réseau Québecois de Calcul de Haute Performance - for its assistance in providing computing facilities and Mr. François Guertin's time. The RQCHP has been created through the financial support of the Canada Foundation for Innovation, the Ministry of Education of the Province of Québec and the industrial partners SGI, SUN, NEC, and IBM.

References

[1] Balakrishnan A., Magnanti T.L., and Mirchandani P. (1997), "Network Design", *Annotated Bibliographies in Combinatorial Optimization*, Dell'Amico M., Maffioli F. and Martello S. (eds.), John Wiley & Sons, New York, NY, 311-334.

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

[3] Crainic T.G., Gendreau M., and Fravolden J. (2000), "A simplex-Based Tabu Search Method for Capacitated Network Design", *INFORMS Journal on Computing* 12, 223-236.

[4] Gendron B., and Crainic T.G. (1994), "Relaxations for Multicommodity Capacitated Network Design Problems", Publication CRT-965, Centre de recherche sur les transports, Université de Montréal.

[5] Gendron B., and Crainic T.G. (1996), "Bounding Procedures for Multicommodity Capacitated Fixed Charge Network Design Problems", publication CRT-96-06, Centre de recherche sur les transports, Université de Montréal.

[6] Gendron B., Crainic T.G., and Frangioni A. (1999), "Multicommodity Capacitated Network Design", Chapter 1 in *Telecommunications Network Planning*, Sansò B., and Soriano P. (eds.), Kluwer Academics Publishers, Norwell, MA, 1-19.

[7] Ghamlouche I., Crainic T.G., and Gendreau M. (2001), "Cycle Based Neighborhood Structures for Fixed-Charge Capacitated Multicommodity Network Design", forthcoming *Operations Research*.

[8] Ghamlouche I., Crainic T.G., and Gendreau M. (2002), "Path Relinking for Fixed-Charge Capacitated Multicommodity Network Design", Publication CRT-2001-01, Centre de recherche sur les transports, Université de Montréal.

[9] Glover F., and Laguna M. (1997), *Tabu Search*, Kluwer Academic Publishers, Norwell, MA.

[10] Holmberg K., and Yuan D. (2000), "A Lagrangian Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem", *Operations Research* 48, 461-481.

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

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

[13] Kim D., and Pardalos P.M. (2000), "A Dynamic Domain Contraction Algorithm for Nonconvex Piecewise Linear Network Flow Problems", *Journal of Global Optimization* 17, 225-234.

[14] Magnanti T.L., and Wong R.T. (1984), “Network Design and Transportation Planning: Models and Algorithms”, *Transportation Science*, 18, 1-55.

[15] Minoux M. (1989), “Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications”, *Networks* 19, 313-360.

[16] Yaged B. (1971), “Minimum Cost Routing for Static Network Models”, *Networks* 1, 139-172.