Tabu Search for a Network Loading Problem with Multiple Facilities

David Berger[*] Bernard Gendron[†] Jean-Yves Potvin[‡]
S. Raghavan[§] Patrick Soriano[¶]

January 1999

Abstract

This paper examines a network design problem that arises in the telecommunications industry. In this problem, communication between a gateway vertex and a number of demand vertices is achieved through a network of fiber optic cables. Since each cable has an associated capacity (bandwidth), enough capacity must be installed on the links of the network to satisfy the demand, using possibly different types of cables. Starting with a network with no capacity or some capacity already installed, a tabu search heuristic is designed to find a solution that minimizes the cost of installing any additional capacity on the network. This tabu search applies a k-shortest path algorithm to find alternative paths from the gateway to the demand vertices. Numerical results are presented on different types of networks with up to 200 vertices and 100 demand vertices.

[*]Centre Universitaire des Sciences et Techniques, Université Blaise Pascal, B.P. 206, 63174 Aubière Cedex, France

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

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

[§]The Robert H. Smith School of Business and the Institute for Systems Research, Van Munching Hall, University of Maryland, College Park, MD 20742, U.S.A.

[¶]Centre de recherche sur les transports et Ecole des Hautes Etudes Commerciales, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7

1 Introduction

Network design problems are pervasive in the real-world and find applications in computer networks, transportation, manufacturing and telecommunications. The early literature in this area has mostly focused on uncapacitated problems [4, 22, 24]. More recently, several authors have tackled capacitated network design problems, with either a single or a very limited number of capacity options on each link [3, 5, 6, 7, 8, 10, 11, 12, 13, 18, 19, 20, 21, 25]. The capacitated problems are more difficult to solve and raise considerable algorithmic challenges. Heuristic methods are thus particularly indicated to address instances of realistic sizes.

This paper introduces a tabu search heuristic for a variant of the network loading problem [21] where facilities of fixed capacity can be installed on the links of the network to carry the flow from a central vertex to a set of demand vertices. No bifurcation of the flow is allowed: a single path must be used to satisfy the demand at each vertex. The motivation for this study comes from an application in the telecommunications area, where the central vertex is a gateway (or backbone vertex), the demand vertices are customer sites and the facilities to be installed or loaded are fiber optic cables. Since several types of cables are available, each with a corresponding capacity, the problem is to determine the number and type of cables that should be installed on each link of the network to satisfy the demand at minimum cost. The objective function to be minimized is simply the cost of loading the network through the acquisition and installation of cables, as no routing costs are present. The cost of a cable increases with its capacity with, typically, substantial economies of scale.

The bifurcated version of the network loading problem has been studied in [6, 21]. The problem in [21] relates to the design of a private communication network where only two types of facilities are offered. A Lagrangean relaxation strategy and a cutting plane approach are developed for this particular case. A branch-and-cut algorithm for a similar problem is also reported in [6]. Due to the exact methodologies adopted in these two papers, solutions have been found for relatively small problem instances with at most 15 vertices.

The nonbifurcated version of the problem has been studied in [5] for a single type of facility. Here, subsets of vertices are aggregated into "supervertices" in order to reduce the problem size. Once an exact solution is found for the aggregated network, a heuristic solution for the original network is derived. Although solutions to problems with up to 64 vertices have been obtained, the aggregated networks never contained more than 15 vertices. Other related problems, where both loading and routing costs are minimized simultaneously, may also be found in [10, 11, 25].

In the following, Section 2 first presents a formal definition of our problem. The tabu search heuristic that addresses this problem is described in Section 3. A k-shortest path algorithm, used to generate the neighborhood structure of the tabu search, is then presented in Section 4. Finally, computational results are reported in Section 5 on typical telecommunications data.

2 Problem Definition

Let $G = (V, E)$ be an undirected graph where V is the vertex set of cardinality n, E is the edge set of cardinality m and $D \subseteq V$ is the subset of demand vertices of cardinality r. The path-based formulation of our network loading problem is then:

Minimize

$$\sum_{e \in E} l_e \left(\sum_{t \in T} c_t y_{te} \right)$$

subject to

$$\sum_{i \in D} \left(\sum_{j \in P_i} x_{ij} \delta_{ej} \right) d_i \leq w_e + \sum_{t \in T} w_t y_{te}, \ \forall e \in E, \tag{1}$$

$$\sum_{j \in P_i} x_{ij} = 1, \ \forall i \in D, \tag{2}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \ \forall j \in P_i, \tag{3}$$

$$y_{te} \text{ integer}, \forall t \in T, \ \forall e \in E, \tag{4}$$

where

T is the set of cable types,

P_i is the set of paths from the central vertex to demand vertex i,

d_i is the demand at vertex i,

c_t is the cost of a cable of type t (per unit of length),

w_t is the capacity of a cable of type t,

l_e is the length of edge e,

w_e is the initial capacity on edge e,

δ_{ej} is equal to 1 if edge e is on path j, 0 otherwise.

This formulation contains two sets of variables: the variables x_{ij} which are equal to 1 if path $j \in P_i$ is used to service demand vertex i, 0 otherwise; and the design variables y_{te} which are equal to the number of cables of type t to be installed on edge e. In the model, the first set of constraints specifies that the capacity installed on the network must be sufficient to handle the flow on each edge. The second set of constraints requires nonbifurcating flows, that is, the use of a single path from the central vertex to each demand vertex. The objective is to minimize the acquisition and installation costs of any additional capacity on the edges of the network to satisfy the demand at each vertex (in case of a tie between two or more alternative solutions, the one that minimizes the number of edges carrying the flow is chosen).

This problem is NP-hard, as it contains the fixed charge network design problem (and thus, the Steiner tree problem) as a special case. It thus provides opportunities for the application of heuristic methods. In the next section, a tabu search heuristic is proposed to address this problem.

3 Solution Procedure

In recent years, tabu search (TS) has been applied with a high degree of success to a variety of NP-hard problems [16]. It is basically an iterative neighborhood search strategy that allows moves that degrade the objective function. Through such moves, the method can escape from bad local optima (as opposed to a pure descent approach). To avoid cycling, a short term memory, known as the tabu list, stores previously visited solutions or components of previously visited solutions. It is then forbidden or tabu to come back to these solutions for a certain number of iterations. Our tabu search heuristic follows the general guidelines provided in [14, 15]. It also includes a long term adaptive memory (AM) aimed at providing new starting points for the search [27].

We first briefly sketch the general algorithmic structure of our tabu search heuristic. Each component is then presented in greater detail in the following subsections.

```
1. while *stopping criterion of AM loop* is not met do:
    a. if the adaptive memory is empty (at the start of the algorithm) then
           generate initial solution $s$ through heuristic means;
           $s^{best} = s$;
       otherwise
           generate $s$ by combining paths found in the adaptive memory;
    b. $s^* = s$;
    c. while *stopping criterion of TS loop* is not met do:
           generate a neighborhood of $s$ through non tabu moves, or tabu
           moves that lead to solutions that improve $s^*$, and select the best
           solution $s'$;
           store $s'$ in the adaptive memory, if indicated (see subsection 3.2);
           if $s'$ is better than $s^*$ then $s^* = s'$;
           $s = s'$;
       d. if $s^*$ is better than $s^{best}$ then $s^{best} = s^*$;
2. output $s^{best}$
```

In this algorithm, the variables s^* and s^{best} are used to store the best solution of the current tabu search restart and best overall solution, respectively.

3.1 Initialization

At the outset, an initial solution is produced by individually computing the path of minimum cost from the central vertex to each demand vertex. These paths are then combined to provide a starting point for the tabu search. Although each path is by itself the best way to reach the associated demand vertex, the initial solution is not globally optimal, because these paths typically share common edges. In the remainder of the algorithm, the adaptive memory is used to produce the starting solutions, as it is explained below.

3.2 Adaptive Memory

The adaptive memory stores the paths associated with elite solutions in order to create good starting points for the tabu search. This memory is divided into r fixed size "compartments", one for each demand vertex. A given compartment contains paths leading from the central vertex to the associated demand vertex. A path has a score which corresponds to the objective value of the best visited solution that used that path. The memory is managed as follows.

When a new current solution s is produced by the tabu search, the paths in this solution are stored in their associated compartment if: (1) the compartment is not full or (2) the path is better than the worst path in the compartment, in which case the new path takes its place.

When a new starting solution is asked for by the tabu search, a path is selected from each compartment (i.e., a path leading to each demand vertex is chosen). The selection in each compartment is biased towards the paths with the best scores. To this end, the paths are ranked from best to worst in each compartment. The path with the best score is associated with some *Max* value, while the path with the worst score is associated with some *Min* value. The values for the other paths are equally spaced between *Min* and *Max*. More precisely, assuming d paths in a compartment (with $d > 1$), the value v_i of a path of rank i is computed according to the formula:

$$v_i = Max - (Max - Min) \times \frac{i-1}{d-1}, \quad 1 \leq i \leq d.$$

The selection probability p_i of the path of rank i is then:

$$p_i = \frac{v_i}{\sum_{j=1}^{d} v_j}, \quad 1 \leq i \leq d.$$

By imposing that $Min + Max = 2$, the selection bias in favor of the best paths can be increased by setting the Max value closer to 2, or reduced by setting its value closer to 1. To obtain a good compromise between exploitation of elite solutions and exploration of new solutions, the values $Min = 0.5$ and $Max = 1.5$ are typically used [2, 30]. Once the selection process is completed in each compartment, the selected paths are simply put together to form the new starting solution. This strategy for creating solutions is an instance of what is often referred to as "vocabulary building": fragments of solutions are assembled to create larger fragments, until ultimately producing complete solutions (see chapter 7 in [17]).

3.3 Neighborhood

The neighborhood structure is the most important issue in the development of a tabu search heuristic. Here, a neighborhood of solutions is constructed by considering, for each demand vertex, the best alternative path to reach that vertex, using the k-shortest path algorithm presented in section 4 with k=2. For r demand vertices, r new solutions can be produced, where each solution differs from the original one by a single path. The cost of each alternative path is computed *using the flow already present on the paths to the other* $r - 1$ *demand vertices in the current solution.* This can be done, as a single path is involved and the costs incurred for any additional capacity are easily computed (see subsection 5.1). By adjusting the needed capacity to the flow, the solutions considered are always feasible.

Once the best solution in the neighborhood is chosen, the associated demand vertex is "tabu" for a number of iterations randomly chosen in the interval $[tabu_{min}, tabu_{max}]$. That is, the path leading to that vertex cannot be changed for that number of iterations. By associating a tabu status to the vertices, rather than to the paths themselves, a conservative approach is favored as many moves may be forbidden (and thus, many new solutions may be overlooked) to reduce the risks of cycling. However, a tabu move may still be applied if it leads to a solution that is better than the best solution visited thus far.

3.4 Stopping Criteria

At the upper level (AM loop) of the algorithm, the number of tabu search restarts from the adaptive memory *rst* is set to some a priori value. Within the tabu search (TS loop), the followings are used: maximum number of consecutive iterations $iter^*$ without improving s^* or maximum number of iterations per restart $iter_{max}$.

In the next section, the k-shortest path algorithm used to generate the neighborhood structure of the tabu search heuristic is presented.

4 k-Shortest Paths

At each iteration of the tabu search heuristic, the best alternative path leading to each demand vertex must be computed. A k-shortest path algorithm is used to this end (where "shortest" refers to the path of minimum cost). Many methods for identifying the k shortest paths are extensions of algorithms developed for the classical 1-shortest path problem [1]. The latter algorithms may be classified as label-setting or label-correcting methods. In both cases, a label is associated with each vertex: this label corresponds to an upper bound on the shortest distance to reach that vertex. In label-setting methods, a vertex label is made permanent at each iteration of the algorithm, when a shortest path to that vertex has been found. In label-correcting methods, the labels are temporary until the final step, when they all become permanent. In this work, an adaptation of Dijkstra's label-setting algorithm [9] is used to solve the k-shortest path problem. This algorithm is indicated when (1) the edges of the network have non negative costs and (2) the shortest paths from a central vertex to all other vertices must be computed. Furthermore, networks found in telecommunications applications are sparse and label setting algorithms are particularly efficient in this case [28].

In this adaptation, the scalar label associated with every vertex i is replaced by a vector Π_i of size k whose p-th element π_i^p is an upper bound on the length of the p-th shortest path to vertex i (i.e., the labels are sorted in nondecreasing order) [23]. Two pointers are also associated with every vertex i: pointer p_i indicates the position of the first temporary label in the vector Π_i and q_i indicates the position of the last label with finite value. When position k is reached by p_i, this pointer becomes $k+1$ while q_i remains k. The status of each vertex is then determined as follows:

1. if $p_i = k+1$, the labels of vertex i are all permanently set, that is, its k shortest paths have been found;

2. if $p_i \leq k$ and $p_i > q_i$, then vertex i has no finite temporary labels;

3. if $p_i \leq k$ and $q_i \geq p_i$, then vertex i has finite temporary labels from position p_i to q_i in Π_i.

The algorithm also maintains a priority queue Q that contains candidate vertices with a finite minimum temporary label. The next permanent label is always chosen among them.

Assuming that k shortest paths are to be found from a central vertex 0 to each vertex in a graph $G = (V, E)$, and denoting l_{ij} the length of edge (i, j) and S_i the set of vertices adjacent to vertex i, the algorithm can be summarized as follows:

```
0. Initialization
    $\Pi_0 = (0, \infty, \infty, \ldots, \infty)$;
    $\Pi_i = (\infty, \infty, \infty, \ldots, \infty), \forall i \in V$;
    $p_i = 1, \forall i \in V; q_0 = 1; q_i = 0, \forall i \in V - \{0\}$;
    $Q = \{0\}$;
1. Main procedure
    a. find $j$ such that $\pi_j^{p_j} = \min\{\pi_q^{p_q}, q \in Q\}$;
    b. $\forall j' \in S_j$ such that $p_{j'} \leq k$ do
        $U = \pi_j^{p_j} + l_{jj'}$;
        if $p_{j'} > q_{j'}$ then
            $\pi_{j'}^{p_{j'}} = U$; $q_{j'} = q_{j'} + 1$; insert $j'$ into $Q$;
        otherwise
            $\hat{p} = \min\{q_{j'} + 1, k\}$;
            for $p = p_j$ to $\hat{p}$ do
                if $U < \pi_{j'}^p$ then
                    if $p < \hat{p}$ then push elements one position down in $\Pi_{j'}$
                            from $p$ to $\hat{p}$;
                    $\pi_{j'}^p = U$;
                    if $q_{j'} < k$ then $q_{j'} = q_{j'} + 1$;
                    exit the for loop;
    c. $p_j = p_j + 1$;
    d. if $q_j < p_j$ then remove $j$ from $Q$;
    e. if $Q \neq \emptyset$ then go to step a, otherwise stop.
```

The k-shortest paths obtained with this algorithm may contain cycles (e.g., in our context, it may well happen that the best alternative path to a demand vertex is the current one plus a small cycle). Clearly, loopless paths are looked for. One way to deal with this problem is to generate a reasonably large set of paths among which the desired, loopless, paths may be found. Apart from the fact that "reasonably large" may be difficult to quantify and may vary from one problem to another, this approach is too computationally expensive because it is used at each iteration of our tabu search heuristic. Hence, the algorithm has been modified in a simple way to forbid paths with cycles: once the minimum temporary label $\pi_j^{p_j}$ is chosen in Step a, any vertex j' that is adjacent to j, but is already found on the path, is excluded from further consideration in Step b. Clearly, this approach always produces loopless paths. However, it does not necessarily produce the shortest ones, as some paths may be overlooked. This modified version thus constitutes a heuristic approach to the k shortest simple path problem.

5 Computational Results

In order to evaluate our tabu search, we performed computational experiments on randomly generated networks with characteristics frequently observed in practice. We first introduce the cost data and network topologies used for the experiments. Then, we present numerical results obtained with the tabu search and two other neighborhood search heuristics.

5.1 Cost Data

The acquisition and installation costs for different types of fiber optic cables come from a real-world application. Since these costs are confidential, they are shown in generic form in Table 1. As observed in this table, it is not always optimal to use the smallest single cable that can accommodate the flow. For a flow of 100, for example, it is better to use cables 7 and 1 with a total capacity of 102 and a total cost of 3.58, rather than cable 8 with capacity 144 and a cost of 4.37. This additional complication can be alleviated through the following observations:

- except for cable 9, it is never advantageous to use the same cable twice (for example, installing cable 1 twice provides 12 fibers, while the same capacity is obtained at lower cost using cable 2);
- cable 9 is mandatory for flows of 216 or more, as it is less expensive than any other combination of smaller cables.

The special structure of these costs, typically found in practice, can thus be exploited to a priori determine the cables to be installed for any flow between 0 and $\sum_{i=1}^{r} d_i$:

Cable type	capacity (number of fibers)	cost (per unit of length)
1	6	0.55
2	12	0.73
3	24	1.03
4	36	1.39
5	48	1.67
6	72	2.31
7	96	3.03
8	144	4.37
9	216	6.33

Table 1: Cable types

- for flows over 216, cable 9 is installed until a residual flow under 216 is obtained;
- for flows between 1 and 215, all possible combinations of *different* cables are considered (except cable 9), and the combination of lowest cost is kept.

This preprocessing is done only once at the start of the algorithm. The results are then used to associate the appropriate combination of cables with the required additional capacity.

5.2 Test Problems

General undirected networks with $n = 50$, 100 and 200 vertices were randomly generated to test our algorithms. These networks are sparse and quasi-planar, like those found in practice, with edge densities (i.e., fraction of all possible edges) varying between .1 and .2. More precisely, three different types of problems were considered in the computational experiments, namely:

- networks with 50 vertices, 25 demand vertices and edge density of 0.2;
- networks with 100 vertices, 50 demand vertices and edge density of 0.15;
- networks with 200 vertices, 100 demand vertices and edge density of 0.1;

A demand is thus associated with 50% of the vertices. The demand vertices fall into three categories: category 1 with 1 to 20 units of demand; category 2 with 40 to 60 units; category 3 with 80 to 100 units. The demand vertices are evenly distributed among the three categories. The problems come either with no equipment at all or some equipment already installed. In the latter case, the total capacity of the network, which is the sum of the capacities over all links of the network, is twice the total demand. This capacity is then uniformly distributed over all links, with a random perturbation on each link (i.e. $\pm$ 0-20%).

5.3 Parameter Settings

Preliminary experiments have been conducted to determine appropriate parameter values for the tabu search heuristic. Extensive computations have been performed, in particular, to fine tune the adaptive memory size, tabu tenure and stopping criterion. The values tested in each case, are summarized below:

- adaptive memory: r compartments, each of size
  - $M_1 = \lceil \sqrt{\frac{n}{2}} \rceil$
  - $M_2 = \lceil \sqrt{n} \rceil$
  - $M_3 = \lceil 2\sqrt{n} \rceil$
  - $M_4 = \lceil 4\sqrt{n} \rceil$
- tabu tenure ($tabu_{min}$, $tabu_{max}$):
  - $T_1 = (\lceil 0.25\sqrt{n} \rceil, \lceil 0.5\sqrt{n} \rceil)$
  - $T_2 = (\lceil 0.5\sqrt{n} \rceil, \lceil \sqrt{n} \rceil)$
  - $T_3 = (\lceil \sqrt{n} \rceil, \lceil 1.5\sqrt{n} \rceil)$
  - $T_4 = (\lceil 1.5\sqrt{n} \rceil, \lceil 2\sqrt{n} \rceil)$
- stopping criterion (rst, $iter^*$, $iter_{max}$):
  - $I_1 = (1, 5n, 50n)$
  - $I_2 = (5, n, 10n)$
  - $I_3 = (10, \lceil n/2 \rceil, 5n)$
  - $I_4 = (20, \lceil n/4 \rceil, 2.5n)$

This leads to 64 different combination of values. We show here the results for only a few selected combinations on the largest problems with 200 vertices. Tables 2 and 3 first show the average solution value obtained with the "intermediate" values M_2, M_3, T_2, T_3, I_2 and I_3 on ten different problem instances (with and without initial capacity). These results indicate that there is not much difference in solution quality from one setting to another. However, slightly better solutions are obtained when more tabu search restarts are performed (c.f., I_3 versus I_2).

When the "extreme" parameter values M_1, M_4, T_1, T_4, I_1 and I_4 are used, solution quality deteriorates. This is illustrated in Tables 4 and 5. When intermediate values are combined with extreme values, an improvement is obtained

Initial solution	M2,T2,I2	M2,T2,I3	M2,T3,I2	M2,T3,I3
12065.9	11418.6	11412.9	11414.0	11406.5
	M3,T2,I2	M3,T2,I3	M3,T3,I2	M3,T3,I3
	11417.3	11412.4	11415.8	11406.2

Table 2: Parameter calibration on problems with 200 vertices; no initial capacity

Initial solution	M2,T2,I2	M2,T2,I3	M2,T3,I2	M2,T3,I3
11160.6	10658.3	10648.5	10653.7	10649.9
	M3,T2,I2	M3,T2,I3	M3,T3,I2	M3,T3,I3
	10658.9	10644.4	10649.1	10646.6

Table 3: Parameter calibration on problems with 200 vertices; with initial capacity

Initial solution	M1,T1,I1	M1,T1,I4	M1,T4,I1	M1,T4,I4
12065.9	11457.9	11419.1	11433.9	11427.8
	M4,T1,I1	M4,T1,I4	M4,T4,I1	M4,T4,I4
	11457.7	11418.6	11432.7	11421.7

Table 4: Parameter calibration on problems with 200 vertices; no initial capacity

Initial solution	M1,T1,I1	M1,T1,I4	M1,T4,I1	M1,T4,I4
11160.6	10685.0	10665.5	10668.8	10662.2
	M4,T1,I1	M4,T1,I4	M4,T4,I1	M4,T4,I4
	10686.2	10660.7	10662.1	10666.7

Table 5: Parameter calibration on problems with 200 vertices; with initial capacity

in a few cases, but solution quality never reaches the numbers found in Tables 2 and 3.

The parameter setting M_3, T_3, I_3 came out as one of the best in our calibration experiments and was used in the following. Note also that the number of paths k is always set to 2, because we only need to find the best path, other than the one currently in use, that leads to any given vertex.

5.4 Two Alternative Heuristics

Given that lower bounds for the network loading problem are not tight enough to provide insights about the performance of our tabu search heuristic, comparisons are provided in the following with the 1-opt and 2-opt neighborhood search heuristics [26], which both stop at the first local minimum. Using a pool of shortest paths (leading from the central vertex to each demand vertex) these heuristics look for the best combination of such paths. In contrast with the adaptive memory of the tabu search heuristic, the pool is fixed and is not enriched with new paths as these algorithms unfold. The neighborhood structure of the 1-opt heuristic is similar to the one used by tabu search. That is, a path leading to a particular vertex in the current solution is replaced by an alternative path found in the pool. The size of the neighborhood is thus $O(rk)$, where r is the number of demand vertices and k is the number of paths associated with each demand vertex in the pool. The 2-opt heuristic explores a larger neighborhood: for each pair of paths leading to demand vertices in the current solution, a new pair of paths is looked for in the pool. The complexity of this neighborhood is $O\left(k^2r^2\right)$ and includes 1-opt as a special case, when one of the two paths remains in place. To compare 1-opt and 2-opt with tabu search, the pool of paths was produced with the k-shortest path algorithm presented in Section 4, using $k = \lceil 2\sqrt{n} \rceil$ (i.e., the size of a compartment in the adaptive memory). Both methods also start with the same initial solution as the tabu search, which consists in the best individual path leading to each demand vertex. At each iteration, these methods select the best solution in the neighborhood of the current solution; the procedure is then repeated with the new solution until a local minimum is reached.

5.5 Experiments

This subsection compares the tabu search with the 1-opt and 2-opt heuristics. Tables 6 and 7 compare these methods on problems with 50, 100 and 200 vertices, with and without initial capacity, respectively. The numbers found in each entry associated with "Tabu", "1-opt" and "2-opt" are average solution value and average computation time in seconds on a Sun UltraSparc 1 workstation (140 MHz), over ten different instances.

These results indicate that Tabu outperforms 1-opt and 2-opt with respect to solution quality. In particular, Tabu finds better solutions than the 2-opt heuristic, even though it explores a more restricted 1-opt neighborhood. Based on these numbers, the average percentage of improvement of Tabu over the initial solution, defined as

$$\frac{\mathit{Initial\ Solution} - \mathit{Tabu}}{\mathit{Initial\ Solution}} \times 100,$$

is 3.3% on the problems of size 50, 4.2% on the problems of size 100 and 4.8% on the problems of size 200. Over 1-opt, the percentages of improvement are 0.6% on the problems of size 50, 1.0% on the problems of size 100 and 0.7% on the problems of size 200. Over 2-opt, these percentages drop to 0.4%, 0.4% and 0.3%, respectively. In all cases, the percentages of improvement are lower when no initial capacity is present. That is, the initial solution made of the best individual path leading to each vertex is closer to the optimal solution in these cases. When some initial capacity is already installed, the solutions tend to exploit as much as possible the available capacity, thus leading to paths that are often quite different from the initial ones.

It is worth noting that a variant of the tabu search heuristic with continuous diversification, obtained through the addition of a penalty term in the objective value for frequently used paths, did not provide any further improvement, even for larger values of k (note that increasing the value of k only makes sense for this variant, as the cost of a path is perturbed with the frequency penalty after its computation; thus, the ranking of the paths with the perturbed costs is likely to be different from the one with the original costs). The exploitation of an adaptive memory within our tabu search heuristic, which is also aimed at favoring the exploration of interesting new areas of the search space, explains the redundancy of the continuous diversification approach.

Tables 8 and 9 show detailed results on five instances with 200 vertices, with and without initial capacity, respectively. The tabu search was executed 3 times on each instance to provide some insight on the stability of the final solution (note that the tabu search contains stochastic features, as opposed to the 1-opt

Problem size	Initial solution	1-opt	2-opt	Tabu
50	3856.8	3756.6 0.3s	3753.0 2.8s	3744.5 25.2s
100	6922.4	6725.2 2.5s	6703.7 95.9s	6694.9 275.5s
200	11602.0	11106.4 33.2s	11048.1 4402.3s	11034.8 4099.6s

Table 6: Problems with no initial capacity

Problem size	Initial solution	1-opt	2-opt	Tabu
50	3400.7	3307.9 0.3s	3291.0 3.3s	3274.2 24.8s
100	6463.6	6228.2 2.5s	6174.4 110.9s	6134.6 260.9s
200	11019.2	10593.1 32.4s	10543.3 4193.4s	10504.9 3830.1s

Table 7: Problems with initial capacity

and 2-opt heuristics). The two numbers in each entry under "Tabu" are the average solution value and best solution value over the three runs, respectively. The average gap between the best and average solution is 0.05% when no initial capacity is present and 0.1% otherwise. The gap never goes beyond 0.12%. These percentages are even smaller for the problems with 100 and 50 vertices. This indicates that the tabu search is quite robust and that its performance does not vary widely due to its stochastic features.

Finally, although Tabu finds better solutions than the other two methods, it is more computationally expensive for $n = 50$ and 100. For n=200, however, Tabu runs faster than 2-opt, as the neighborhood of the latter grows quickly with problem size. In general, any additional computation time leading to even small improvements can be easily justified since the costs involved in telecommunications application are in the order of several million dollars.

6 Conclusion

A tabu search heuristic was developed and applied to network loading problems with up to 200 vertices and 100 demand vertices. Heuristics are particularly

Problem instance	Initial solution	1-opt	2-opt	Tabu
1	12870.7	12123.2	12064.6	12055.0 12049.7
2	12671.5	11979.8	11900.6	11902.8 11891.6
3	12389.8	11856.2	11820.5	11785.1 11774.1
4	12643.3	11932.0	11885.4	11865.0 11862.4
5	9754.2	9501.2	9470.9	9471.1 9467.4

Table 8: Problem instances with 200 vertices; no initial capacity

Problem instance	Initial solution	1-opt	2-opt	Tabu
1	11993.5	11347.2	11243.7	11178.8 11167.6
2	11576.2	11144.9	11083.1	11081.2 11073.6
3	11162.9	10914.3	10891.1	10867.1 10855.4
4	11759.5	11259.5	11218.6	11202.9 11189.2
5	9310.9	8978.3	8955.2	8909.4 8905.0

Table 9: Problem instances with 200 vertices; with initial capacity

indicated in this case, since exact methods cannot currently address instances of this size. Tabu search has produced better results than a descent heuristic based on the same 1-opt neighborhood structure. This result underlines the benefits associated with the mechanisms at the core of tabu search to escape from local optima. Tabu search has also outperformed a descent heuristic based on an extended 2-opt neighborhood structure that includes the former. To the best of our knowledge, this is the first application of a tabu search heuristic to this type of problem. Other types of telecommunications problems, however, could benefit as well from this methodology (see, for example, chapter 8 in [17]).

Acknowledgments. Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Quebec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). This support is gratefully acknowledged. Thanks also to François Guertin for running the computational experiments

References

[1] Ahuja R.K., T.L. Magnanti and J.B. Orlin (1993), *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall: Englewood Cliffs, NJ.

[2] Baker J. (1985), "Adaptive Selection Methods for Genetic Algorithms", in *Proceedings of the First Int. Conf. on Genetic Algorithms and their Applications*, J.J. Grefenstette ed., Lawrence Erlbaum Associates: Hillsdale, NJ, 101–111.

[3] Balakrishnan A., T.L. Magnanti and P. Mirchandani (1998), "Network Design", in Annotated Bibliographies in Combinatorial Optimization, M. Dell'Amico, F. Maffioli and S. Martello eds, Wiley, 311–334.

[4] Balakrishnan A., T.L. Magnanti and R.T. Wong (1989), "A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design", *Operations Research 37*, 716–740.

[5] Barahona F. (1996), "Network Design Using Cut Inequalities" *SIAM J. on Optimization 6*, 823–837.

[6] Bienstock D. and O. Günlük (1996), "Capacitated Network Design – Polyhedral Structure and Computation" *INFORMS J. on Computing 8*, 243–259.

[7] Crainic T.G., A. Frangioni and B. Gendron (1998), "Bundle-Based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems", Technical Report CRT-98-45, Centre de recherche sur les transports, Université de Montréal.

[8] Crainic T.G., M. Gendreau and J.M. Farvolden (1996), "Simplex-Based Tabu Search for the Multicommodity Capacitated Fixed Charge Network Design Problem", Technical Report CRT-96-07, Centre de recherche sur les transports, Université de Montréal.

[9] Dijkstra E.W. (1959), "A Note on Two Problems in Connection with Graphs", *Numerische Mathematik 1*, 269–271.

[10] Gavish B. and I. Neuman (1989), "A System for Routing and Capacity Assignment in Computer Communication Networks", *IEEE Trans. Commun.*, 360–366.

[11] Gavish B. and A. Altinkemer (1990), "Backbone Network Design Tools with Economic Tradeoffs", *ORSA J. on Computing 2*, 236–252.

[12] Gendron B., T.G. Crainic and A. Frangioni (1998), "Multicommodity Capacitated Network Design", forthcoming in Telecommunications Network Planning, B. Sansó and P. Soriano eds, Kluwer.

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

[14] Glover F. (1989), "Tabu Search - Part I", *ORSA J. on Computing 1*, 190–206.

[15] Glover F. (1990), "Tabu Search - Part II", *ORSA J. on Computing 2*, 4–32.

[16] Glover F. (1997), "Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges", in *Advances in Metaheuristics, Optimization and Stochastic Modeling Technologies*, R.S. Barr, R.V. Helgason and J.L. Kennington eds, Kluwer:Boston, 1–75.

[17] Glover F. and M. Laguna (1997), *Tabu Search*, Kluwer:Boston.

[18] Holmberg K. and D. Yuan (1996), "A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem", Research Report LiTH-MAT-R-1996-23, Dept. of Mathematics, Linkoping Institute of Technology.

[19] Magnanti T.L. and P. Mirchandani (1993), "Shortest Paths, Single Origin-Destination Network Design and Associated Polyhedra", *Networks 23*, 103–121.

[20] Magnanti T.L., P. Mirchandani and R. Vachani (1993), "The Convex Hull of Two Core Capacitated Network Design Problems", *Mathematical Programming 60*, 233–250.

[21] Magnanti T.L., P. Mirchandani and R. Vachani (1995), "Modeling and Solving the Two-Facility Capacitated Network Loading Problem" *Operations Research 43*, 142–157.

[22] Magnanti T.L. and R.T. Wong (1984), "Network Design and Transportation Planning: Models and Algorithms", *Transportation Science 18*, 1–55.

[23] Miaou S.P. and S.M. Chin (1991), "Computing k-Shortest Path for Nuclear Spent Fuel Highway Transportation", *European J. of Operational Research 53*, 64–80.

[24] Minoux M. (1989), "Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications", *Networks 19*, 313–360.

[25] Ng T. and D. Hoang (1987), "Joint Optimization of Capacity and Flow Assignment in a Packet-Switched Communication Network", *IEEE Trans. Comput. 35*, 202–209.

[26] Raghavan S. (1995), "A Heuristic for the IOF Routing Problem", Working Paper, US West Advanced Technologies, Boulder, CO.

[27] Rochat, Y. and É.D. Taillard (1995), "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics 1*, 147–167.

[28] Shier D.R. (1979), "On Algorithms for Finding the k-Shortest Paths in a Network", *Networks 9*, 195–214.

[29] Taillard É.D. (1993), "Parallel Iterative Search Methods for Vehicle Routing Problems", *Networks 23*, 661–673.

[30] Whitley D. (1989), "The GENITOR Algorithm: Why Rank-Based Allocation of Reproductive Trials is Best", in *Proceedings of the Third Int. Conf. on Genetic Algorithms*, J.D. Schaffer ed., Morgan Kaufmann: San Mateo, CA, 116–121.