A GRASP WITH PATH-RELINKING FOR PRIVATE VIRTUAL CIRCUIT ROUTING

MAURICIO G.C. RESENDE AND CELSO C. RIBEIRO

Abstract. A frame relay service offers virtual private networks to customers by provisioning a set of long-term private virtual circuits (PVCs) between customer endpoints on a large backbone network. During the provisioning of a PVC, routing decisions are made without any knowledge of future requests. Over time, these decisions can cause inefficiencies in the network and occasional offline rerouting of the PVCs is needed. In this paper, the offline PVC routing problem is formulated as an integer multicommodity flow problem with additional constraints and with an objective function that minimizes delays and network overload. We propose variants of a GRASP with path-relinking heuristic for this problem. Experimental results for realistic-size problems are reported, showing that the proposed heuristics are able to improve the solutions found with standard routing techniques. Moreover, the structure of our objective function provides a useful strategy for setting the appropriate value of its weight parameter, to achieve some quality of service (QoS) level defined by a desired maximum utilization rate.

1. Introduction

A frame relay service offers virtual private networks to customers by provisioning a set of permanent (long-term) private virtual circuits (PVCs) between endpoints on a large backbone network. During the provisioning of a PVC, routing decisions are made either automatically by the frame relay switch or by the network designer, through the use of preferred routing assignments and without any knowledge of future requests. Over time, these decisions usually cause inefficiencies in the network and occasional rerouting of the PVCs is needed. The new routing scheme is then implemented on the network through preferred routing assignments. Given a preferred routing assignment, the switch will move the PVC from its current route to the new preferred route as soon as this move becomes feasible.

One possible way to create the preferred routing assignments is to appropriately order the set of PVCs currently in the network and apply an algorithm that mimics the routing algorithm used by the frame relay switch to each PVC in that order. However, more elaborate routing algorithms, that take into account factors not considered by the switch, could further improve the efficiency of network resource utilization.

Typically, the routing scheme used by the frame relay switch to automatically provision PVCs is also used to reroute them in the case of trunk or card failures. Therefore, this routing algorithm should be efficient in terms of running time, a

Date: June 15, 2001.

Key words and phrases. Multicommodity network flows, frame relay, private virtual circuit, routing, bandwidth packing, GRASP, path-relinking, metaheuristics, offline rerouting.

AT&T Labs Research Technical Report.

requirement that can be traded off for improved network resource utilization when building preferred routing assignments offline.

In this paper, we propose variants of a GRASP (greedy randomized adaptive search procedure) with path-relinking algorithm for the problem of routing offline a set of PVC demands over a backbone network, such that delays are minimized and network load is balanced. This problem and its variants are also known in the literature as bandwidth packing problems. The set of PVCs to be routed can include all or a subset of the PVCs currently in the network, and/or a set of forecast PVCs. The explicit handling of delays, as opposed to just handling the number of hops (as in the routing algorithm implemented in Cisco switches) is particularly important in international networks, where distances between backbone nodes vary considerably. Network load balancing is important for providing the maximum flexibility to handle the following situations:

- overbooking, which is typically used by network designers to account for non-coincidence of traffic;
- PVC rerouting, due to link or card failures; and
- bursting above the committed rate, which is not only allowed but sold to customers as one of the attractive features of frame relay.

In Section 2, we formulate the offline PVC routing problem as an integer multicommodity flow problem with additional constraints and a hybrid objective function, which takes into account delays and network load. Minimum cost multicommodity network flow problems are characterized by a set of commodities flowing through an underlying network, each commodity having an associated integral demand which must flow from its source to its destination. The flows are simultaneous and the commodities share network resources. If the cost function in each edge is convex, then this problem can be solved in polynomial time [21]. The problem is NP-hard if the flows are required to be integral [11] or if each commodity is required to follow a single path from its source to its destination [9]. In Section 3, we propose variants of a GRASP with path-relinking heuristic for this problem. Experimental results, reported in Section 4, show that the proposed heuristics are able to improve the solutions found with standard routing techniques on realistic-size problems. Concluding remarks are made in Section 5.

2. Problem formulation

Let $G = (V, E)$ be an undirected graph representing the frame relay network. We denote by $V = \{1, \ldots, n\}$ the set of backbone nodes where switches reside, while E is set of trunks (or edges) that connect the backbone nodes, with $|E| = m$. Parallel trunks are allowed. Since G is an undirected graph, flows through each trunk $(i, j) \in E$ have two components to be summed up, one in each direction. However, for modeling purposes, costs and capacities will always be associated only with the ordered pair (i, j) satisfying $i < j$. For each trunk $(i, j) \in E$, we denote by b_{ij} its maximum allowed bandwidth (in kbits/second), while c_{ij} denotes the maximum number of PVCs that can be routed through it and d_{ij} is the delay associated with the trunk. Each commodity $k \in K = \{1, \ldots, p\}$ is a PVC to be routed, associated with an origin-destination pair and with a bandwidth requirement (or demand, also known as its effective bandwidth) r_k. It takes into account the actual bandwidth required by the customer in the forward and reverse directions, as well as an overbooking factor.

The delay on each trunk is associated with propagation or hopping. The ultimate objective of the offline PVC routing problem is to minimize PVC delays while balancing trunk loads, subject to several technological constraints. Network load balancing is achieved by minimizing the load on the most utilized trunk. Routing assignments with minimum delays may not achieve the best trunk load balance. Likewise, routing assignments having the best trunk load balance may not minimize delays. A compromising objective is to route the PVCs such that a desired point in the tradeoff curve between PVC delays and trunk load balancing is achieved.

The upper bound on the number of PVCs allowed on a trunk depends on the port card used to implement it. A set of routing assignments is feasible if and only if for every trunk $(i,j) \in E$ the total PVC effective bandwidth requirements routed through it does not exceed its maximum bandwidth b_{ij} and the number of PVCs routed through it is not greater than c_{ij}.

Let x_{ij}^k be a 0-1 variable such that $x_{ij}^k = 1$ if and only if trunk $(i,j) \in E$ is used to route commodity $k \in K$ from node i to node j. The following linear integer program models the problem:

$$\min \phi(x) = \sum_{(i,j)\in E, i<j} \phi_{ij}(x_{ij}^1, \cdots, x_{ij}^p, x_{ji}^1, \cdots, x_{ji}^p) \tag{1}$$

subject to

$$\sum_{k\in K} r_k(x_{ij}^k + x_{ji}^k) \leq b_{ij}, \quad \forall (i,j) \in E, i<j, \tag{2}$$

$$\sum_{k\in K} (x_{ij}^k + x_{ji}^k) \leq c_{ij}, \quad \forall (i,j) \in E, i<j, \tag{3}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(i,j)\in E} x_{ji}^k = a_i^k, \quad \forall i \in V, \forall k \in K, \tag{4}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in E, \forall k \in K. \tag{5}$$

Constraints of type (2) limit the total flow on each trunk to at most its capacity. Constraints of type (3) enforce the limit on the number of PVCs routed through each trunk. Constraints of type (4) are flow conservation equations stating that the flow associated with each PVC cannot be split, where $a_i^k = 1$ if node i is the source for commodity k, $a_i^k = -1$ if node i is the destination for commodity k, and $a_i^k = 0$ otherwise.

The cost function $\phi_{ij}(x_{ij}^1, \cdots, x_{ij}^p, x_{ji}^1, \cdots, x_{ji}^p)$ associated with each trunk $(i,j) \in E$ with $i<j$ is the linear combination of a delay component and a load balancing component. The delay component is defined as

$$\phi_{ij}^d(x_{ij}^1, \cdots, x_{ij}^p, x_{ji}^1, \cdots, x_{ji}^p) = d_{ij} \cdot \sum_{k\in K} \rho_k(x_{ij}^k + x_{ji}^k), \tag{6}$$

where coefficients ρ_k are used to model two plausible delay functions:

- If $\rho_k = 1$, then this component leads to the minimization of the number of hops weighted by the delay on each trunk.
- If $\rho_k = r_k$, then the minimization takes into account the effective bandwidth routed through each trunk weighted by its delay.

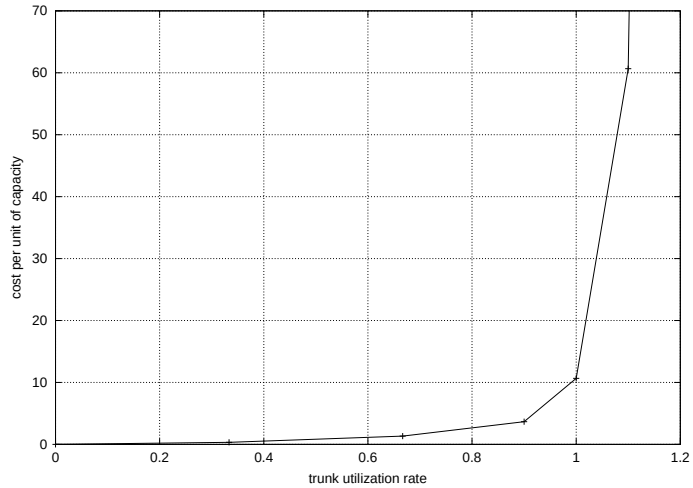

FIGURE 1. Piecewise linear load balance cost component associated with each trunk.

Let $y_{ij} = \sum_{k \in K} r_k(x^k_{ij} + x^k_{ji})$ be the total flow through trunk $(i,j) \in E$ with $i < j$. The load balance component depends on the utilization rates $u_{ij} = y_{ij}/b_{ij}$ of each trunk $(i,j) \in E$ with $i < j$. It is taken as the piecewise linear function proposed by Fortz and Thorup [13] and depicted in Figure 1, which increasingly penalizes flows approaching or violating the capacity limits:

(7)
$$\phi^b_{ij}(x^1_{ij}, \cdots, x^p_{ij}, x^1_{ji}, \cdots, x^p_{ji}) = b_{ij} \cdot \begin{cases} u_{ij}, & u_{ij} \in [0, 1/3) \\ 3 \cdot u_{ij} - 2/3, & u_{ij} \in [1/3, 2/3), \\ 10 \cdot u_{ij} - 16/3, & u_{ij} \in [2/3, 9/10), \\ 70 \cdot u_{ij} - 178/3, & u_{ij} \in [9/10, 1), \\ 500 \cdot u_{ij} - 1468/3, & u_{ij} \in [1, 11/10), \\ 5000 \cdot u_{ij} - 16318/3, & u_{ij} \in [11/10, \infty). \end{cases}$$

The value $\Omega = \max_{(i,j) \in E, i<j}\{u_{ij}\}$ gives a global measure of the maximum congestion in the network.

In this paper, we use the cost function

(8)
$$\begin{aligned} &\phi_{ij}(x^1_{ij}, \cdots, x^p_{ij}, x^1_{ji}, \cdots, x^p_{ji}) = \\ &\quad = (1-\delta) \cdot \phi^d_{ij}(x^1_{ij}, \cdots, x^p_{ij}, x^1_{ji}, \cdots, x^p_{ji}) + \delta \cdot \phi^b_{ij}(x^1_{ij}, \cdots, x^p_{ij}, x^1_{ji}, \cdots, x^p_{ji}) \end{aligned}$$

associated with each trunk $(i,j) \in E$ with $i < j$, where weights $(1-\delta)$ and δ correspond respectively to the delay and the load balance components, with $\delta \in [0,1]$. We show in Section 4 that small values, such as $\delta = 0.1$, lead to feasible solutions minimizing the overall delays (measured in terms of either hops or propagation) and with balanced loads on the trunks, characterized by reduced values of the maximum congestion index Ω. Other choices for the objective functions are possible, see e.g. Awduche et al. [3].

Several heuristics have been proposed for different variants of the bandwidth packing problem. One of the first algorithms for routing virtual circuits in communication networks was proposed by Yee and Lin [28]. Their problem is formulated as a nonlinear multicommodity flow problem with integer decision variables. Their heuristic applies Lagrangean relaxation and a multiplier adjustment procedure to solve a sequence of restricted problems. Computational results are illustrated for problems in three different networks. Their largest problem had 61 nodes and 148 links. Sung and Park [26] have also developed a Lagrangean heuristic for a similar variant of this problem. They limited its application to six small networks, the largest of which had 20 nodes and 52 links. Laguna and Glover [18] considered a bandwidth packing problem in which they want to assign calls to paths in a capacitated graph, such that capacities are not violated and some measure of the total profit is maximized. They develop a tabu search algorithm which makes use of an efficient implementation of the k-shortest path algorithm. Computational results for small problems involving up to 31 nodes and 50 calls are reported. Amiri et al. [2] proposed another formulation for the bandwidth packing problem. They consider both revenue losses and costs associated with communication delays as part of the objective. A heuristic procedure based on Lagrangean relaxation is applied for finding bounds and solutions. Computational results are reported for problems with up to 50 nodes, with the number of calls ranging from 50 to 90% of the maximum number of all possible calls. Resende and Resende [23] proposed a GRASP for offline PVC rerouting in which a different objective function is considered. Their construction and local search procedures are different than those proposed in this paper. In particular, the construction procedure often encountered difficulties in finding feasible solutions for tightly constrained instances. Also, their local search procedure is much more time-consuming, limiting its application to small instances. Shyur and Wen [25] proposed a tabu search algorithm for optimizing the system of virtual paths. The objective function consists in minimizing the maximum link load, by requiring that each route visits the minimum number of hubs. The load of a link is defined as the sum of the virtual path capacities, summed over the virtual paths that traverse the link. Computation results for problems with up to 64 nodes, 112 links, and 2048 demand pairs are given.

A number of exact approaches for solving variants of the bandwidth packing problem have also appeared in the literature. Parker and Ryan [22] described a branch and bound procedure for optimally solving a bandwidth packing problem. Their objective is to allocate bandwidth so as to maximize the total revenue. The linear relaxation of the associated integer programming problem is solved using column generation. Computational results for 14 different networks with up to 29 nodes, 61 links, and 93 calls are presented. LeBlanc et al. [20] addressed packet switched telecommunication networks, considering restrictions on paths and flows: hop limits, node and link capacity constraints, and high- and low-priority flows. They minimize the expected queueing time and do not impose integrality constraints on the flows. Dahl et al. [10] studied a network configuration problem in telecommunications, searching for paths in a capacitated network to accommodate a given traffic demand matrix. Their model also involves an intermediate pipe layer. The problem is formulated as an integer linear program, where the 0-1 variables represent different paths. An associated integral polytope is studied and different classes of facets are described. These are embedded in a cutting plane algorithm. Computational results for realistic-size problems, with up to 62 nodes, 81

links, and 33 origin-destination pairs are presented. Barnhart et al. [4] proposed a branch-and-cut-and-price algorithm for origin-destination integer multicommodity flow problems. This problem is a constrained version of the linear multicommodity network flow problem, in which each flow may use only path from its origin to its destination. Because this model contains one variable for each origin-destination path, for every commodity, the linear programming relaxations are solved using column generation. New branching rules allow columns to be efficiently generated at each node of the branch and bound tree. New cuts can also be generated at each node of the branch and bound tree, helping to strengthen the linear programming relaxation. Implementation details, together with computational results for problems with at most 50 nodes, 130 edges, and 585 commodities, are reported.

The model (1)–(5) proposed in this section has two distinctive features with respect to other formulations. First, it takes into account a two component objective function which is able to handle both delays and load balance. Second, it enforces constraints that limit the maximum number of PVCs that can be routed through any trunk. An approximate algorithm for its solution is described in the next section.

3. Approximate algorithm for PVC routing

A GRASP is a multistart or iterative process, in which each GRASP iteration consists of two phases, a construction phase, in which a feasible solution is produced, and a local search phase, in which a local optimum in the neighborhood of the constructed solution is sought [12]. The best overall solution is kept as the result. The pseudo-code in Figure 2 illustrates a general GRASP procedure for the minimization of an objective function $f(x)$ under constraints $x \in X$, in which `Max_Iterations` GRASP iterations are done.

```
procedure GRASP
1   f* ← ∞;
2   for k = 1, ... , Max_Iterations do
3       Construct a greedy randomized solution x ∈ X;
4       Find y by applying local search to x;
5       if f(y) < f* do
6           x* ← y;
6           f* ← f(x*);
7       end if;
8   end for;
9   return x*;
end GRASP;
```

Figure 2. Pseudo-code of a general GRASP procedure.

A feasible solution is iteratively constructed in the first phase, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list with respect to its contribution to the objective function. The list of best candidates is called the *restricted candidate list* (RCL).

The random selection of an element from the RCL allows for different solutions to be obtained at each GRASP iteration.

Another construction mechanism, called *heuristic-biased stochastic sampling*, was introduced by Bresina [6]. In the construction procedure of the basic GRASP, the next element to be introduced in the solution is chosen at random from the candidates in the RCL. The elements of the RCL are assigned equal probabilities of being chosen. However, any probability distribution can be used to bias the selection towards some particular candidates. Bresina [6] introduced a family of such probability distributions. In Bresina's selection procedure, the candidates are ranked according to the greedy function. Binato et al. [5] use Bresina's selection procedure, but restricted to elements of the RCL.

Since solutions generated by a GRASP construction are not guaranteed to be locally optimal, it is almost always beneficial to apply a local search to attempt to improve each constructed solution.

In the remainder of this section, we customize a GRASP for the offline PVC routing problem. We describe construction and local search procedures, as well as a path-relinking intensification strategy.

3.1. **Construction phase.** In the construction phase, the routes are determined, one at a time. A new PVC is selected to be routed in each iteration. To reduce the computation times, we used a combination of the strategies usually employed by GRASP and heuristic-biased stochastic sampling. We create a restricted candidate list with a fixed number of elements n_c. At each iteration, it is formed by the n_c unrouted PVCs pairs with the largest demands. An element ℓ is selected at random from this list with probability $\pi(\ell) = r_\ell / \sum_{k \in \mathrm{RCL}} r_k$.

Once a PVC $\ell \in K$ is selected, it is routed on a shortest path from its origin to its destination. The capacity constraints (2) are relaxed and handled via the penalty function introduced by the load balance component (7) of the edge weights. The constraints of type (3) are explicitly taken into account by forbidding routing through trunks already using its allowable number of PVCs. The weight $\Delta\phi_{ij}$ of each edge $(i,j) \in E$ is given by the increment of the cost function value $\phi_{ij}(x^1_{ij}, \cdots, x^p_{ij}, x^1_{ji}, \cdots, x^p_{ji})$, associated with routing r_ℓ additional units of demand through edge (i,j).

More precisely, let $\underline{K} \subseteq K$ be the set of previously routed PVCs and $\underline{K}_{ij} \subseteq \underline{K}$ be the subset of PVCs that are routed through trunk $(i,j) \in E$. Likewise, let $\overline{K} = \underline{K} \cup \{\ell\} \subseteq K$ be the new set of routed PVCs and $\overline{K}_{ij} = \underline{K}_{ij} \cup \{\ell\} \subseteq \overline{K}$ be the new subset of PVCs that are routed through trunk (i,j). Then, we define $\underline{x}^\ell_{ij} = 1$ if PVC $\ell \in \underline{K}$ is routed through trunk $(i,j) \in E$ from i to j, $\underline{x}^\ell_{ij} = 0$ otherwise. Similarly, we define $\overline{x}^\ell_{ij} = 1$ if PVC $\ell \in \overline{K}$ is routed through trunk $(i,j) \in E$ from i to j, $\overline{x}^\ell_{ij} = 0$ otherwise. According with (8), the cost associated with each edge $(i,j) \in E$ in the current solution is given by $\phi_{ij}(\underline{x}^1_{ij}, \cdots, \underline{x}^p_{ij}, \underline{x}^1_{ji}, \cdots, \underline{x}^p_{ji})$. In the same manner, the cost associated with each edge $(i,j) \in E$ after routing PVC ℓ will be $\phi_{ij}(\overline{x}^1_{ij}, \cdots, \overline{x}^p_{ij}, \overline{x}^1_{ji}, \cdots, \overline{x}^p_{ji})$. Then, the incremental edge weight $\Delta\phi_{ij}$ associated with routing PVC $\ell \in K$ through edge $(i,j) \in E$, used in the shortest path computations, is given by

$$\Delta\phi_{ij} = \phi_{ij}(\overline{x}^1_{ij}, \cdots, \overline{x}^p_{ij}, \overline{x}^1_{ji}, \cdots, \overline{x}^p_{ji}) - \phi_{ij}(\underline{x}^1_{ij}, \cdots, \underline{x}^p_{ij}, \underline{x}^1_{ji}, \cdots, \underline{x}^p_{ji}). \tag{9}$$

The enforcement of type (3) constraints may lead to unroutable demand pairs. In this case, the current solution is discarded and a new construction phase starts.

3.2. **Local search.** Each solution built in the first phase may be viewed as a set of routes, one for each PVC. Our local search procedure seeks to improve each route in the current solution. For each PVC $k \in K$, we start by removing r_k units of flow from each edge in its current route. Next, we compute incremental edge weights $\Delta\phi_{ij}$ associated with routing this demand through each trunk $(i, j) \in E$ according to (9), as described in Section 3.1. A tentative new shortest path route is computed using the incremental edge weights. If the new route improves the solution, it replaces the current route of PVC k. This is continued until no improving route can be found.

3.3. **Path-relinking.** Path-relinking was originally proposed by Glover [14] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [15, 16, 17]. Starting from one or more elite solutions, paths in the solution space leading towards other elite solutions are generated and explored in the search for better solutions. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions. Path-relinking may be viewed as a strategy that seeks to incorporate attributes of high quality solutions, by favoring these attributes in the selected moves.

The use of path-relinking within a GRASP procedure as an intensification strategy applied to each locally optimal solution was first proposed by Laguna and Martí [19], being followed by several extensions, improvements, and successful applications [1, 8, 24].

In this context, path-relinking is applied to pairs $x_1 - x_2$ of solutions, where x_1 is the locally optimal solution obtained after local search and x_2 is one of a few elite solutions randomly chosen from a pool with a limited number `Max_Elite` of elite solutions found along the search. The pool is originally empty. Each locally optimal solution obtained by local search is considered as a candidate to be inserted into the pool if it is different (by at least one trunk in one route, in the case of the bandwidth packing problem) from every other solution currently in the pool. If the pool already has `Max_Elite` solutions and the candidate is better than the worst of them, then the former replaces the latter. If the pool is not full, the candidate is simply inserted.

The algorithm starts by computing the symmetric difference $\Delta(x_1, x_2)$ between x_1 and x_2, resulting in a set of moves which should be applied to one of them (the initial solution) to reach the other (the guiding solution). Starting from the initial solution, the best move still not performed is applied to the current solution, until the guiding one is attained. The best solution found along this trajectory is also considered as a candidate for insertion in the pool and the incumbent is updated. Several alternatives have been considered and combined in recent implementations to explore trajectories connecting x_1 and x_2:

- do not apply path-relinking at every GRASP iteration, but only periodically;
- explore two different trajectories, using first x_1, then x_2 as the initial solution;
- explore only one trajectory, starting from either x_1 or x_2; and
- do not follow the full trajectory, but instead only part of it.

All these alternatives involve the trade-offs between computation time and solution quality. Ribeiro et al. [24] observed that exploring two different trajectories for each pair $x_1 - x_2$ takes approximately twice the time needed to explore only one of them, with very marginal improvements in solution quality. They have also observed that if only one trajectory is to be investigated, better solutions are found

when path-relinking starts from the best among x_1 and x_2. Since the neighborhood of the initial solution is much more carefully explored than that of the guiding one, starting from the best of them gives to the algorithm a better chance to investigate with more details the neighborhood of the most promising solution. For the same reason, the best solutions are usually found closer to the initial solution than to the guiding one, allowing pruning the relinking trajectory before the latter is reached.

Computational results illustrating the trade-offs between these strategies for the bandwidth packing problem are reported later in Section 4. In this case, the set of moves corresponding to the symmetric difference $\Delta(x_1, x_2)$ between any pair $x_1 - x_2$ of solutions is the subset $K_{x_1,x_2} \subseteq K$ of PVCs routed through different routes in x_1 and x_2. Without loss of generality, let us suppose that path-relinking starts from any elite solution z in the pool and uses the locally optimal solution y as the guiding solution.

The best solution $\overline{y}$ along the new path to be constructed is initialized with z. For each PVC $k \in K_{y,z}$, the same shortest path computations described in Sections 3.1 and 3.2 are used to evaluate the cost of the new solution obtained by rerouting the demand associated with PVC k through the route used in the guiding solution y instead of that used in the current solution originated from z. The best move is selected and removed from $K_{y,z}$. The new solution obtained by rerouting the above selected PVC is computed, the incumbent $\overline{y}$ is updated, and a new iteration resumes. These steps are repeated, until the guiding solution y is reached. The incumbent $\overline{y}$ is returned as the best solution found by path-relinking and inserted into the pool if it satisfies the membership conditions.

The pseudo-code with the complete description of the procedure `GRASP+PR_BPP` for the bandwidth packing problem arising in the context of offline PVC rerouting is given in Figure 3. This description incorporates the construction, local search, and path-relinking phases.

4. Computational experiments

The experiments were performed on an SGI Challenge computer (28 196-MHz MIPS R10000 processors) with 7.6 Gb of memory. Each run used a single processor. The algorithms were coded in Fortran and were compiled with the SGI MIPSpro F77 compiler using flags `-O3 -64 -static`. CPU times were measured with the system function `etime`.

The experiments were run on two groups of test instances. The first one is formed by some of the test problems from three of the classes used by Fortz and Thorup [13]. The first class is the *AT&T Worldnet backbone* with projected demands, a real-world network with 90 nodes and 274 links. The other two classes are formed by synthetic networks. More specifically, *2-level hierarchical graphs* are generated using the GT-ITM generator [29], based on a model of Calvert et al. [7] and Zegura et al. [30]. Edges are of two types: local access trunks and long distance trunks. The capacities of edges of the same type are equal. Local access trunks have lower capacities than long distance trunks. On *Waxman graphs*, the nodes are uniformly distributed points in the unit square. The probability of having an edge between two nodes u and v is given by $\eta e^{-\delta(u,v)/2\theta}$, where η is a parameter used to control the density of the graph, $\delta(u, v)$ is the Euclidean distance between u and v, and θ is the maximum distance between any two nodes [27]. All trunk capacities are equal. The demands are such that different nodes have different levels of activity,

```
procedure GRASP+PR_BPP;
  φ* ← ∞;
  Pool ← ∅;
  for k = 1, ... , Max_Iterations do
      Construct a greedy randomized solution x;
      Find y by applying local search to x;
      if y satisfies the membership conditions then insert y into Pool;
      Randomly select an elite solution z ∈ Pool with uniform probability;
      Compute K_{y,z};
      Let ȳ be the best solution found by applying path-relinking to y − z;
      if ȳ satisfies the membership conditions then insert ȳ into Pool;
      if φ(ȳ) < φ* do
          x* ← ȳ;
          φ* ← φ(x*);
      end if;
  end for;
  return x*;
end GRASP+PR_BPP;
```

FIGURE 3. Pseudo-code of the GRASP with path-relinking procedure for the bandwidth packing problem

modeling hot spots on the network. They are relatively larger between closer pairs of nodes. We have used another problem generator to create the second group of test instances, with characteristics more similar to those of a frame-relay network. This problem generator and the test instances are available from the authors.

Five problems were selected from each of these groups, whose characteristics are summarized in Table 1. These ten instances are among the largest, to date, to appear in the literature. They are available for download [1] from the authors. The table shows, for each instance, its name, network type, number of nodes, number of trunks, number of demand pairs, and the value Φ_{uncap}, which is the same normalizing scaling factor used by Fortz and Thorup [13]. This normalization allows comparing costs across different network sizes and topologies. This uncapacitated measure is defined as

$$\Phi_{uncap} = \sum_{k \in K} r_k \cdot h_k,$$

where r_k is the bandwidth requirement associated with pair $k \in K$ and h_k is the minimum distance measured with unit weights (hop count) between the origin and destination nodes of demand pair k.

4.1. **Algorithm variants.** In the first set of experiments, we considered four variants of the GRASP and path-relinking schemes proposed in Section 3.3:

- `G`: This variant is a pure GRASP with no path-relinking.
- `GPRf`: This variant adds to `G` a one-way path-relinking starting from a locally optimal solution and using a randomly selected elite solution as the guiding solution.

[1] `http://www.research.att.com/~mgcr/data/pvc-routing.tar.gz`

TABLE 1. Problem characteristics.

Instance	Network type	$\|V\|$	$\|E\|$	$\|K\|$	Φ_{uncap}
`att`	AT&T Worldnet backbone	90	274	272	92,607
`hier50a`	2-level hierarchical	50	148	2450	113,976,500
`hier100a`	2-level hierarchical	100	360	9900	435,618,300
`wax50a`	Waxman	50	476	9900	47,719,429
`wax100a`	Waxman	100	230	2220	198,827,455
`fr250`	Frame-relay	60	344	250	173,194
`fr500`	Frame-relay	60	453	500	288,086
`fr750`	Frame-relay	60	498	750	448,220
`fr1000`	Frame-relay	60	518	1000	603,362
`fr1250`	Frame-relay	60	535	1250	955,568

- `GPRb`: This variant adds to `G` a one way path-relinking starting from a randomly selected elite solution and using a locally optimal solution as the guiding solution.
- `GPRfb`: This variant combines `GPRf` and `GPRb`, performing path-relinking in both directions.

We evaluate the effectiveness of the above variants in terms of the tradeoffs between computational time and solution quality. The parameter δ was set to 1 in the objective function, i.e. only the load balancing component is used.

To study the effect of path-relinking on GRASP, we compared the four variants on two instances. The first is instance `att` from Table 1. The second is instance `fr750a`, derived from instance `fr750` from Table 1 by scaling all demands by a factor of $1/1.3 = 0.76923$. Two hundred independent runs for each variant were done for each problem. Execution was terminated when a solution of value less than or equal to `look4` was found. We used `look4` values of 129400 and 479000 for `att` and `fr750a`, respectively. These are sub-optimal values chosen such that the slowest variant could terminate in a reasonable amount of computation time. Empirical probability distributions for time to target solution are plotted in Figures 4 and 5. To plot the empirical distribution for each algorithm and each instance, we associate with the i-th smallest running time t_i a probability $p_i = (i - \frac{1}{2})/200$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 200$. Due to the time taken by the pure GRASP procedure, we limited its plot in Figure 5 to 60 points.

These plots show a similar relative behavior of the four variants on the two instances. Since instance `fr750a` is harder for all variants and computation times are longer, its plot is more discerning. For a given computation time, the probability of finding a solution at least as good as the target value increase from `G` to `GPRf`, from `GPRf` to `GPRfb`, and from `GPRfb` to `GPRb`. For example, there is 9.25% probability for `GPRfb` to find a target solution in less than 100 seconds, while this probability increases to 28.75% for `GPRb`. For `G`, there is a 8.33% probability of finding a target solution within 2000 seconds, while for `GPRf` this probability increases to 65.25%. `GPRb` finds a target solution in at most 129 seconds with 50% probability. For the same probability, this time increases to 172, 1727, and 10933 seconds, respectively, for variants `GPRfb`, `GPRf`, and `G`.

In accordance with these results, variant `GPRb`, which does path-relinking backwards from an elite solution to a locally optimal solution, is the most effective.

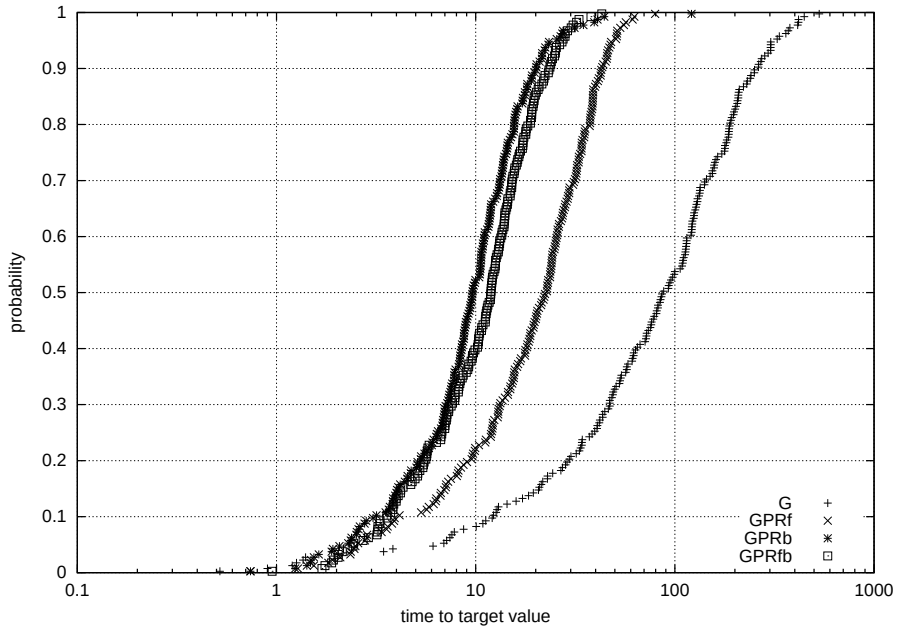

FIGURE 4. Empirical distributions of time to target solution for GRASP, GRASP with forward path-relinking, GRASP with backward path-relinking, and GRASP with back and forward path-relinking for instance `att`.

Because of this, we limit ourselves to only this GRASP with path-relinking variant in the remaining experiments.

4.2. **Comparison with other heuristics.** We now compare `GPRb` using a relatively small number of iterations, fixed at 200, with other simpler heuristics, one of them (heuristic `H1` described below) used in traffic engineering by network planners:

- Heuristic `H1` starts by sorting the pairwise demands in decreasing order and sequentially routes each pair in this order. Each pair is assigned to a minimum hop path (a path minimizing the number of links between the origin and destination nodes).
- Heuristic `H2` also starts by sorting the pairwise demands in decreasing order and sequentially routes each pair in this order. Each pair is assigned to a route minimizing the same cost function ϕ used in `GPRb`.
- Heuristic `H3` adds to `H2` the same local search procedure used in `GPRb`.

The heuristics above have been implemented in Fortran using the same components used to implement the GRASP with path-relinking variants.

We considered the test problems listed in Table 1. We also wanted to compare our heuristics on other test problems, and with other algorithms, described in [4, 25]. Unfortunately, data for these problems were not available from the authors.

Table 2 summarizes the numerical results. For each algorithm and for each instance, we give the normalized value $\Phi^* = \Phi/\Phi_{uncap}$ (where Φ is the cost function value of the best solution found), the corresponding maximum edge utilization rate Ω, and the distribution of the number of edges that have flow in each of the intervals

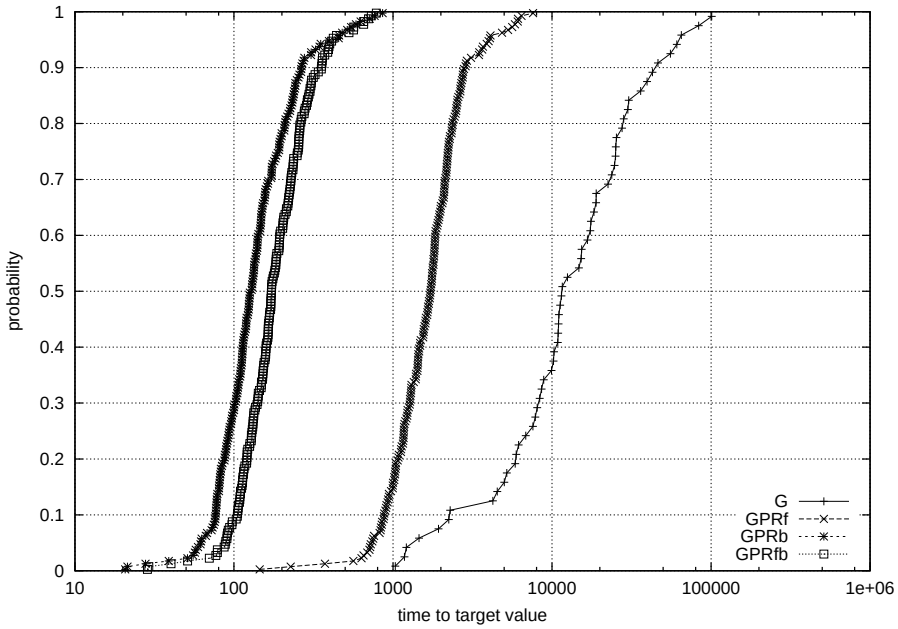

FIGURE 5. Empirical distributions of time to target solution for GRASP, GRASP with forward path-relinking, GRASP with backward path-relinking, and GRASP with back and forward path-relinking for instance `fr750a`.

defining each function ϕ^b_{ij}. The distribution is represented by a sequence of integers, separated by slashes. Right trailing zeroes are omitted. For example, 0/88/416/14 (results obtained by `GPRb` for instance `fr1000`) corresponds to a solution in which 88 edges have their utilization rates in the interval $[1/3, 2/3)$, 416 edges have their utilization rates in the interval $[2/3, 9/10)$, and 14 edges have their utilization rates in the interval $[9/10, 1)$. Better solutions, in general, will be characterized by smaller cost values, smaller maximum utilization rates, and distributions skewed to the left.

Heuristic `H1` does not take into account the cost function ϕ. As expected, for the other heuristics, `H3` systematically finds solutions with smaller costs than those found by `H2`, while `GPRb` further improves upon `H3`. Though none of the heuristics considers explicitly the minimization of the maximum utilization rate, this rate is systematically reduced by going from `H1` to `H2`, to `H3`, and to `GPRb`.

In general, both the local search in `H3` and, more strongly `GPRb`, contribute to improve the distribution of edges and to rerouting them on less loaded edges. As a result, the skewness to the left is accentuated and the maximum utilization rate is reduced. Figure 6 shows plots illustrating this for all ten test instances. Each plot has two parts. In the left, we represent the difference between the distributions found by heuristics `H2` and `H3`. For each interval, we give the increase in the corresponding number of edges in the solution found by `H3` with respect to those in the solution found by `H2`. Similar results are depicted in the right side of each plot, regarding the solutions obtained by `H2` and `GPRb`. In each case, the areas above and below the horizontal axis are equal.

TABLE 2. Numerical results for short runs.

	H1			H2		
Instance	Φ^*	Ω	distribution	Φ^*	Ω	distribution
att	615.34	6.079	228/17/13/1/2/13	1.4995	0.700	216/52/6
hier50a	312.03	2.355	109/21/6/4/0/8	1.2586	0.669	86/60/2
hier100a	460.48	3.151	231/54/25/12/6/39	1.4211	0.900	141/177/42
wax50a	49.048	1.290	109/79/24/3/6/9	1.7204	0.810	5/212/13
wax100a	101.01	2.026	198/218/42/4/5/9	3.8827	1.125	13/447/11/2/0/3
fr250	670.57	3.245	227/62/9/6/9/31	4.0348	1.026	62/223/44/13/2
fr500	959.05	4.119	264/68/37/13/9/62	3.6743	1.006	0/307/125/20/1
fr750	1118.5	4.749	235/98/44/12/13/96	4.3025	1.012	0/43/429/25/1
fr1000	1254.1	4.087	224/96/40/17/10/131	4.9545	0.935	0/1/454/63
fr1250	1909.5	6.420	185/77/49/20/21/183	2472.6	3.278	0/0/0/0/1/534
	H3			GPRb		
Instance	Φ^*	Ω	distribution	Φ^*	Ω	distribution
att	1.3682	0.689	225/46/3	1.3578	0.689	230/40/4
hier50a	1.2295	0.668	92/54/2	1.2141	0.667	103/44/1
hier100a	1.3363	0.898	202/136/22	1.3195	0.875	220/124/16
wax50a	1.4938	0.773	29/197/4	1.4467	0.772	29/197/4
wax100a	2.0175	1.087	18/451/4/0/3	1.9791	1.097	20/449/4/0/3
fr250	4.0042	1.026	69/217/43/13/2	3.3590	1.008	104/194/34/11/1
fr500	3.6270	1.006	2/315/115/20/1	3.1477	1.006	48/304/82/18/1
fr750	4.0242	1.012	1/152/316/28/1	3.5415	1.012	6/206/268/17/1
fr1000	4.6429	0.935	0/19/467/32	3.8461	0.990	0/88/416/14
fr1250	414.56	3.794	0/0/0/1/168/366	345.89	4.867	0/0/0/3/229/303

We note from these plots that both heuristics improve the greedy solution, by reducing the number of overloaded edges and increasing the number of underutilized edges. The plots also illustrate the general performance of GPRb with respect to H3. GPRb tends to improve trunk utilization with respect to H3 by more strongly shifting flow from overloaded to underutilized edges. As a consequence, it obtains solutions characterized by smaller costs and smaller utilization rates.

TABLE 3. Numerical results for long runs.

	computation time							
Instance	25s	125s	625s	3125s	15625s	78125s	Ω	distribution
att	1.3607	1.3578	1.3577	1.3577	1.3562	1.3562	0.689	231/39/4
hier50a	1.2219	1.2189	1.2146	1.2136	1.2129	1.2113	0.667	98/50
wax50a	1.4849	1.4716	1.4533	1.4462	1.4412	1.4384	0.758	21/206/3
fr250	3.7091	3.3590	3.2796	3.2727	3.2497	3.2496	1.008	135/163/34/11/1
fr500	3.5496	3.3340	3.1466	3.1317	3.1306	3.0912	1.006	39/326/69/18/1

Heuristics H1, H2, and H3, which are not multi-start heuristics, are much faster than the multi-start GPRb. However, there is a clear tradeoff in terms of solution quality when extra time is taken by GPRb. We ran GPRb on five of the ten instances in the experiment for about one CPU-day. Table 3 lists objective function values as a function of the running time for these instances, as well as the maximum utilization rate and the distribution of the number of edges that have flow in each of the intervals defining each cost function ϕ^b_{ij}, for the best solution found. We notice that in most of the cases GPRb continues to improve the solution as the running time increases. Even if the maximum utilization rate does not change, the

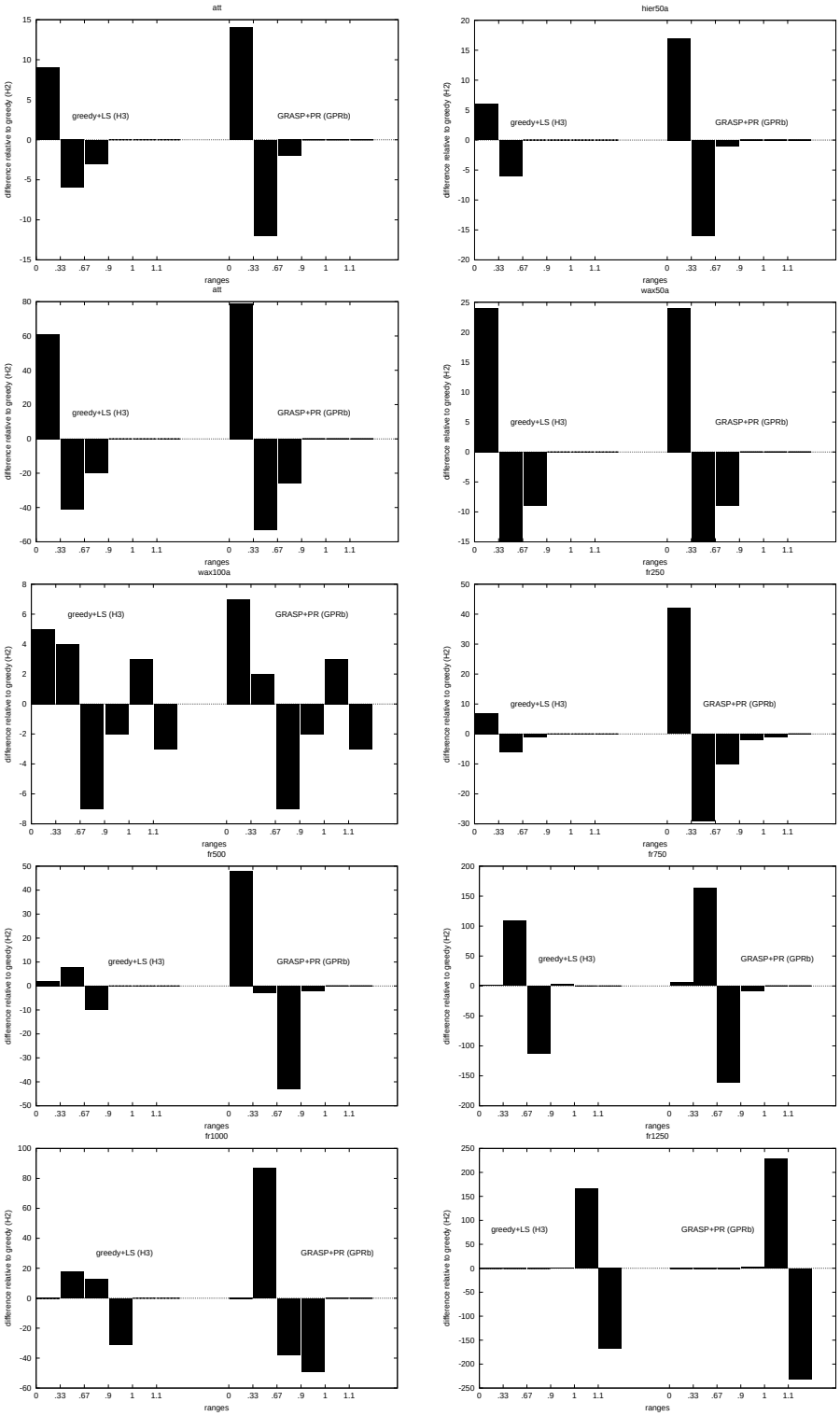

FIGURE 6. Distribution of differences in solutions with respect to the pure greedy heuristic H2

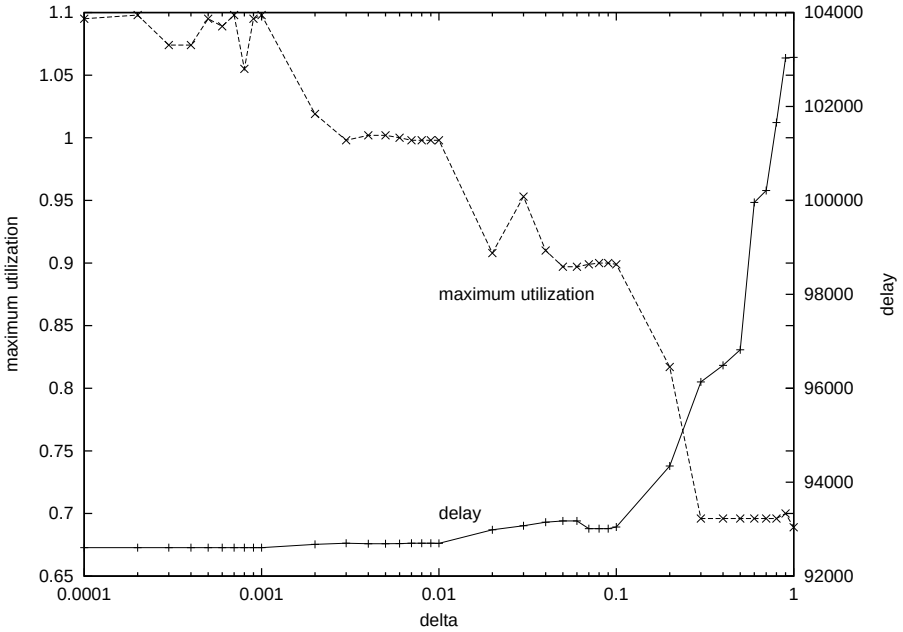

FIGURE 7. Delay and maximum utilization as a function of objective function parameter δ on instance `att` with unit edge delays.

distribution of the number of edges does. For example, on instance `hier50a`, the improved solution shifted several edges into the lowest range with respect to the solution in Table 2.

4.3. **Variation of the hybrid objective function parameter.** In this last experiment, we investigate the behavior of the hybrid objective function ϕ with the variation of the parameter δ, used to weigh the load balance and delay components. We ran `GPRb` for 2000 iterations on instance `att` using 41 different values of δ, ranging from 0 to 1. For each value of δ (with the exception of $\delta = 0$ which cannot be plotted on a log scale), the plot in Figure 7 shows the delay and the maximum utilization rate of the best solution found. Delays were computed using $\rho_k = r_k$ (see Section 2) and using unit delays, i.e. $d_{ij} = 1$, for every $(i, j) \in E$.

We first notice from this figure that there is a range of small values of δ, for which the delay is kept at a low value without serious overload. When the value of δ approaches 1, the maximum utilization rate is strongly reduced at the cost of larger delays. Likewise, when the value of δ approaches 0, the delay is strongly reduced at the expense of higher utilization rates. The extreme case, where $\delta = 0$, corresponds to using the purely greedy heuristic `H1`. In this case, the utilization rate is 6.08, and the delay is 92607, which is a lower bound on the value of optimal solution of the capacity constrained delay minimization problem. Since the resulting utilization rate is high, this is an indication that one should use a strictly positive value of δ.

We also observe that, as the value of δ increases from 0 to 1, the maximum utilization rate decreases, following approximately a step function taking values equal to those appearing in the definition of the functions ϕ^b_{ij}, i.e. 1.1, 1.0, 0.9,

and 0.67. As the value of δ increases, the minimization of the maximum utilization rate dominates the objective function. As a consequence, the algorithm attempts to reduce the flow on edges with higher loads. To balance this reduction, flows on less loaded edges are increased up to the next breakpoint in its cost function. Therefore, the flows have a tendency to concentrate around breakpoint levels. This characteristic provides a useful strategy for setting the appropriate value of the parameter δ of the objective function, to achieve some quality of service (QoS) level defined by a desired maximum utilization rate.

5. Concluding remarks

In this paper, we presented a new formulation for the bandwidth packing problem arising in the context of offline PVC routing. This formulation uses an objective function that simultaneously takes into account network delays and load balance. Emphasis on either component is controlled by a single parameter. We proposed a family of heuristics for finding approximate solutions to this problem, ranging from a simple greedy algorithm (`H2`) and its improved version using local search (`H3`), to an elaborate combination of GRASP and path-relinking.

Experimental results on realistic-size test problems show that even the simplest greedy heuristic (`H2`) is able to improve on a heuristic used in traffic engineering by network planners (`H1`). The two new simple heuristics (`H2` and `H3`) are fast and find good approximate solutions. The GRASP with path-relinking variants are able to significantly improve upon these simple heuristics, at the expense of additional computation time. GRASP with path-relinking has been shown to be efficiently implemented in parallel with approximate linear speedups in the number of processors [1] and such a strategy could be applied to accelerate `GPRb` and its variants.

The structure of the objective function proposed in this paper is such that as the weight of its load balance component increases, the maximum utilization rate decreases, following approximately a step function. As a consequence, this structure provides a useful strategy for setting the appropriate value of the weight parameter of the objective function, to achieve some quality of service (QoS) level defined by a desired maximum utilization rate.

Acknowledgements

This research was done while C. C. Ribeiro was visiting AT&T Labs Research during his sabbatical leave. His work was sponsored by FAPERJ research grant 150966/99 and by CNPq research grants 302281/85-1, 202005/89-5, and 910062/99-4.

References

[1] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path-relinking for the three-index assignment problem. Technical report, AT&T Labs-Research, 2000.

[2] A. Amiri, E. Rolland, and R. Barkhi. Bandwidth packing with queueing delay costs: Bounding and heuristic solution procedures. *European Journal of Operational Research*, 112:635–645, 1999.

[3] D.O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. Technical Report RFC 2702, Network Working Group, 1999. (`http://search.ietf.org/rfc/rfc2702.txt`).

[4] C. Barnhart, C.A. Hane, and P.H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48:318–326, 2000.

[5] S. Binato, W.J. Hery, D. Loewenstern, and M.G.C. Resende. A GRASP for job shop scheduling. In P. Hansen and C.C. Ribeiro, editors, *Essays and surveys on metaheuristics*. Kluwer Academic Publishers, 2001.

[6] J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the AAAI-96*, pages 271–278, 1996.

[7] K. Calvert and M. Doarand E.W. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, 35:160–163, 1997.

[8] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38, 2001.

[9] I. Chlamtac, A. Faragó, and T. Zhang. Optimizing the system of virtual paths. *IEEE/ACM Trans. on Networking*, 2:581–587, 1994.

[10] G. Dahl, A. Martin, and M. Stoer. Routing through virtual paths in layered telecommunication networks. *Operations Research*, 47:693–702, 1999.

[11] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. on Computing*, 5:691–703, 1976.

[12] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[13] B. Fortz and M. Thorup. Increasing internet capacity using local search. Technical report, AT&T Labs Research, Florham Park, NJ 07932, 2000. Preliminary short version of this paper published as "Internet Traffic Engineering by Optimizing OSPF weights" in *Proc. IEEE INFOCOM 2000 – The Conference on Computer Communications*, pp. 519–528.

[14] F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

[15] F. Glover. Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In M. Laguna and J.L. Gonzáles-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 1–24. Kluwer, 2000.

[16] F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

[17] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[18] M. Laguna and F. Glover. Bandwidth packing: A tabu search approach. *Management Science*, 39:492–500, 1993.

[19] M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[20] L.J. LeBlanc, J. Chifflet, and P. Mahey. Packet routing in telecommunication networks with path and flow restrictions. *INFORMS J. on Computing*, 11:188–197, 1999.

[21] A. Ouorou, P. Mahey, and J.P. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46:126–147, 2000.

[22] M. Parker and J. Ryan. A column generation algorithm for bandwidth packing. *Telecommunication Systems*, 2:185–195, 1994.

[23] L.I.P. Resende and M.G.C. Resende. A GRASP for frame relay permanent virtual circuit routing. In P. Hansen and C.C. Ribeiro, editors, *Extended Abstracts of the III Metaheuristics International Conference (MIC'99)*, pages 397–401, 1999.

[24] C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. Technical report, Computer Science Department, Catholic University of Rio de Janeiro, 2001.

[25] C.-C. Shyur and U.-E. Wen. Optimizing the system of virtual paths by tabu search. *European Journal of Operational Research*, 129:650–662, 2001.

[26] C.S. Sung and S.K. Park. An algorithm for configuring embedded networks in reconfigurable telecommunication networks. *Telecommunication Systems*, 4:241–271, 1995.

[27] B.M. Waxman. Routing of multipoint connections. *IEEE J. Selected Areas in Communications*, 6:1617–1622, 1998.

[28] J.R. Yee and F.Y.S. Lin. A routing algorithm for virtual circuit data networks with multiple sessions per O-D pair. *Networks*, 22:185–208, 1992.

[29] E.W. Zegura. GT-ITM: Georgia Tech internetwork topology models (software). Technical report, Georgia Institute of Technology, 1996. (Online document at `http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz`).

[30] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of 15th IEEE Conf. on Computer Communications (INFOCOM)*, pages 594–602, 1996.

(M.G.C. Resende) Information Sciences Research, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com

(C.C. Ribeiro) Department of Computer Science, Catholic University of Rio de Janeiro, r. Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900 Brazil
E-mail address, C.C. Ribeiro: celso@inf.puc-rio.br