Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 151 (2003) 280–295
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

A tabu search algorithm for self-healing ring network design

Bernard Fortz [a], Patrick Soriano [b], Christelle Wynants [c,*]

[a] *Institut d'Administration et de Gestion, Université Catholique de Louvain, Louvain-la-Neuve, Belgium*
[b] *Service des méthodes quantitatives de gestion, École des Hautes Études Commerciales, Centre de recherches sur les transports, Université de Montréal, Montréal, Canada*
[c] *Electrabel—Risk Asset Liability Management, Quantitative Analysis, 8, Boulevard du Regent, Brussels 1000, Belgium*

Abstract

We consider the problem of designing self-healing rings in order to protect the transmission of telecommunication demands in a zonal network. This problem stems from a real application with operational constraints such as dual homing and hop limit per ring. A modeling approach taking into account ring interactions is proposed as well as a tabu search heuristic for solving it. Computational results for a comprehensive set of real and randomly generated instances are presented.

Keywords: Telecommunications; Network design; Survivability; Self-healing rings; Tabu search

1. Introduction

A network is said to be *survivable* if traffic interrupted by the failure of some of its elements can be rerouted via spare or excess capacity specifically placed in the network for that purpose. In addition, if that rerouting process can be automated and the network can reconfigure itself to cope with failures without human intervention, then the network is said to be *self-healing*. Of course, designing a network to survive any type of failures including those involving several elements at the same time would result in designs of prohibitive cost. However, since it is generally considered that failures affecting more than one element at a time are extremely improbable, operators instead define a restricted set of realistic failure scenarios for which the network needs to be survivable. This set usually includes all single link and node failures and, possibly some multilink or node failure scenarios considered critical. The design process then proceeds under the fundamental hypothesis that only one failure needs to be handled at any given time and produces a network that has the topology and spare capacity required to route all the affected traffic "around" these failures (see [6]).

With the advent of SDH/SONET networks and of add–drop multiplexers (ADMs), a new protection technique, known as self-healing ring (SHR), was introduced which maintains very fast reconfiguration times while achieving lower redundant capacity requirements (see [28,34]). Demand nodes are grouped together forming a closed loop or cycle in the network. They are then connected by

* Corresponding author.
E-mail addresses: fortz@poms.ucl.ac.be (B. Fortz), patrick@crt.umontreal.ca (P. Soriano), christelle.wynants@electrabel.com (C. Wynants).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(02)00826-3

using two (eventually more) separate fibers, each carrying signals in opposite directions. At each node connected to the ring, an ADM having an appropriate transmission speed or capacity is installed allowing signals to access or leave the ring. Hence, within such a ring architecture, there always exist two link and node disjoint paths connecting any pair of nodes belonging to the ring. All traffic flowing through a ring is therefore protected against any single failure (as well as some multiple failures) of the links or nodes forming it by providing enough spare capacity on the alternate path of each demand. When a failure is detected within a ring, the ADMs on the ring react and switch the affected traffic to the protection path.

There are in fact several types of SHR depending on the way the reconfiguration of the traffic is performed and which type of protocol is used. Note that all ADMs on a given ring have identical capacities and this is referred to as the ring *capacity* or size. This capacity must be sufficient to accommodate the maximum edge flow among the edges belonging to the ring. The amount of spare capacity required is then simply equal to the ring *working* capacity. However for determining the appropriate ring capacity for a given set of demands, SHRs are classified as belonging to one of two categories depending on the way the working traffic may be routed on them under normal conditions (see [4,35]).

In the first type, called *unidirectional* SHR, all traffic is routed in the same direction along one fiber called the working fiber (e.g. clockwise). The second fiber is set aside as spare to protect the working fiber. Whenever a failure is detected, the system automatically switches the affected traffic from the working fiber onto the protection one as illustrated in Fig. 1. For any communication between a pair of nodes (A, C), the signal from A to C travels on one side of the working fiber ring and the return signal form C to A travels in the same direction but on the opposite side of that ring. Hence, every traffic demand travels the whole circumference and the maximum link flow on the links of the ring is the sum of all demands passing through the ring. Therefore the capacity required for both the working and protection fibers is the sum of all demands carried by the ring. This type of ring is sometimes referred to as a *dedicated protection* SHR since for each demand there is a corresponding amount of spare capacity set aside (i.e. dedicated) specifically to protect it.

In the second type of SHR, known as *bi-directional* SHR, not all working demands need to be routed in the same direction. In a two fiber implementation of such a ring, the bandwidth of each fiber is divided into two equal parts: one serving as working capacity and the other as spare. Both fibers may therefore be used to carry working traffic. For instance, in Fig. 2, both channels of the demand A–C travel on the two fibers along the upper side of the ring, while those of A–D travel along the lower side. Hence, each demand only uses up capacity on the links on which it is routed,

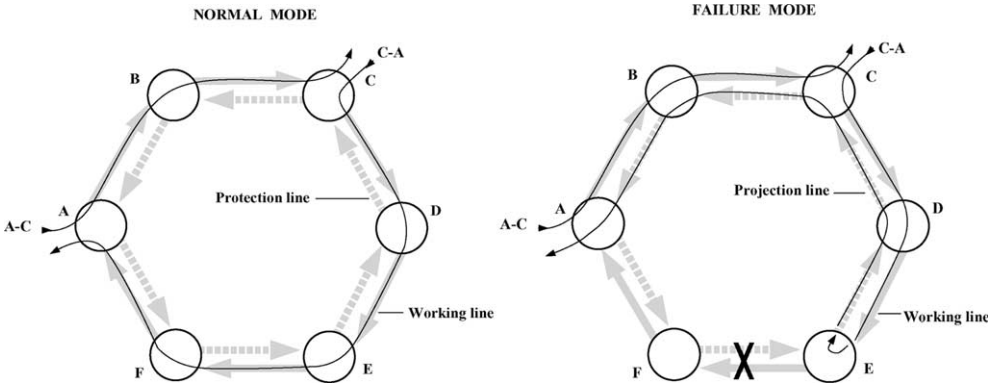

Fig. 1. Unidirectional or dedicated protection SHR.

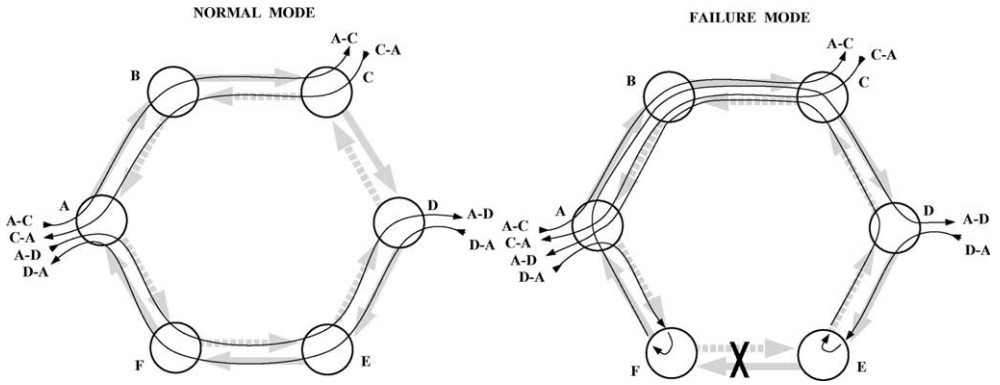

Fig. 2. Bi-directional or shared protection SHR.

leaving the capacity on the rest of the links available for other demands. The capacity is therefore shared among the different demands carried by the ring, which is why it is also referred to as *shared protection* SHR. In the case of a failure, the affected traffic is switched to the spare capacity as illustrated in Fig. 2.

Because of the division into working and protection bandwidth, the overall capacity requirement for each fiber is two times the maximum flow passing through any of its links. By routing the demands along the ring so as to minimize this maximum flow, one may therefore achieve reductions in capacity requirements when compared with the unidirectional SHR. Indeed, if the traffic pattern is relatively uniform between the nodes of the ring then a bi-directional SHR will be more efficient capacity-wise. However, if the traffic is essentially *hubbed*, then there will be no advantage over the unidirectional SHR. This subproblem of *balancing* the flow among the edges of a bidirectional SHR is in itself a very difficult problem. For more details the reader is referred to the following: [5,7,20,23].

Planning the introduction of SHRs in transmission networks is a complex matter that is being studied by a growing number of researchers. This planning problem has been approached from various angles giving rise to quite different design problems depending on the simplifying assumptions considered and what characteristics the resulting survivable architecture should have.

One of the simplest versions of the problem addresses a situation where the network is composed of a single cycle (or topological ring) passing through all the nodes and on which several stacked SHRs must be defined. The network contains a single designated hub node where all SHRs interconnect, allowing demands to flow from one ring to another. Each node can be connected to several SHRs, requiring the placement of an ADM each time. In this stacked ring network design problem, one needs to determine which nodes should be connected to each SHR and how to route the traffic through the rings in order to minimize costs (see [3,30]).

A second class of design problems arises when one requires that the architecture of the resulting network contains multiple rings corresponding to several different topological cycles. This class of problems is further divided depending on whether the different cycles or clusters are required to be disjoint with respect to the demand nodes they connect or not, the former being considered as the ''simpler'' version (see [1,2,8,12,16,22,29]).

In this paper, we consider a real application belonging to the latter type of multiple SHR network design problems. It deals with networks in which the node clusters defining rings are not required to be disjoint (i.e. nodes can be connected to more than one ring) and the resulting architecture is not hierarchical by definition (i.e. inter-ring traffic is routed on the SHRs defining the network). Some of the earlier and better known work on this topic was carried out at Bellcore and resulted in the planning software package known as SONET Toolkit (see [5,6,32,33]). It is essentially

a "greedy" approach that sequentially constructs a survivable network design. It is based on a three level decomposition of the overall planning problem that can be described as follows. In the first stage, clusters of nodes that would make "good" SHRs are identified. These clusters are made up of nodes that share a significant amount of traffic between them and are geographically close together (i.e. thus defining a *community of interests*). Then, a heuristic procedure determines if the network contains an admissible cycle connecting the cluster nodes to form a SHR [31]. If such a cycle exists then the "ring" is included in the final design, otherwise the cluster is rejected. While there are still demands to protect, the procedure tries to identify other interesting clusters and so on. If some demands are left unprotected after completing the ring selection phase (i.e. their origins and destinations do not belong to the same ring), then the software tries to protect them by routing them on (up to three) interconnected rings or, as a last recourse, by assigning them to a diverse protection mechanism. Once all demands have been protected with such structures, the last step of the approach determines the required size for the different SHRs and eventual protection mechanisms that were selected.

Within this same class, Laguna [18] considers the clustering part of the problem that is grouping demand nodes to form individual SHRs. Each node may be connected to several rings. The resulting SHRs are only "logical" structures since no steps are taken in the procedure to guarantee that there exists a feasible cycle connecting the nodes of the ring. All demands must be protected either by being routed within a single ring or, when no ring contains both their end-nodes, by routing them on two different rings, each containing one of the end-nodes. No assumption or restriction is made regarding inter-ring traffic, but a linear cost is associated to it which makes this equivalent to considering that all traffic between any two SHRs pass through a designated hub node. Finally, only unidirectional SHRs (of different capacities) are considered. The objective is to find a set of logical SHRs that optimizes the tradeoffs between ADM placement and inter-ring traffic costs. The author proposes a tabu search heuristic to find feasible solutions.

In this paper, we consider the same design problem but in which inter-ring traffic has to be accounted for explicitly and routed on the SHR network being designed. As in Laguna [18], we also consider unidirectional SHRs of different sizes but in addition we include in our model *hop* constraints and *dual homing* constraints—i.e. constraints forcing each SHR to have at least two different interconnection nodes with other SHRs so that all inter-ring traffic is protected from intermediate node failures. We propose a tabu search algorithm to find good feasible solutions.

To complete this overview, note that contrary to the preceding contributions which were generally intended for application at the local or regional level and used unidirectional rings, some papers deal with variants of the multiple SHR network design problem that are more related to backbone transmission network applications and the use of bi-directional SHRs (see [15,17,19, 21,24]). We refer the reader to Soriano et al. [27] for a more detailed overview.

The remainder of this paper is organized as follows. In the next section, we present the specifics of the problem at hand and a mathematical model for the logical design. The solution approach is described in Section 3 and computational results are presented in Section 4. Finally, to simplify the presentation and given our European industrial partner, we will use only SDH terminology. However, everything translates directly into SONET context.

2. Model and solution approach

The SHR network design problem (or RNDP in short) can be stated as follows: given a set of demand nodes N, a two-connected (meshed) network $G = (N, E)$ connecting these nodes via a set of edges (fiber optic links) E, an *O–D* demand matrix $F = (f^k)_{k \in K}$, where $k = (i, j)$ and f^k is the traffic demand between the origin $i = o(k)$ and destination $j = d(k)$ nodes of commodity k, with $i, j \in N$, one wants to determine a set of feasible SHRs that protects all demands at minimal cost. Let us now characterize more precisely the *RNDP* variant studied here.

- We consider only unidirectional SHRs but of several capacities. To simplify the model formulation, we will denote by $R = \{1, \ldots, R_{\max}\}$ the index set of all possible *ring types* where $R_{\max}$ is an upper bound on the number of rings to install. By ring type, we mean the pair of interconnection nodes that are connected to the ring and its nominal capacity. Note that several indices may correspond to the same features since the solution might require different rings that share common interconnection nodes and have identical capacity specifications. This index set can be trivially bounded above by considering that each demand is protected individually by as many rings as required (for every combination of interconnection node pair and capacity).
- Nodes can be connected to more than one ring. A node is connected to a ring $r \in R$ by the installation of an ADM. This induces a fixed cost c_r depending on the ring size.
- Hub constraints: all traffic between any two SHRs has to pass through a hub node (or interconnection node or zonal center) belonging to both SHRs. We denote by Z the subset of these nodes.
- A demand between nodes i and j can be either routed on a ring containing both nodes i and j, or through 2 or more rings inducing then inter-ring traffic costs. In practice, this cost is a step function depending on the capacity of the interfaces required at the interconnection node but it can be reasonably approximated by a linear function with unit cost b.
- A demand can be split among several paths (i.e. multirouting is allowed) and there is no arbitrary upper limit on the number of rings used by any given demand path.
- Each SHR is required to contain at least two different interconnection nodes (dual homing constraints).
- There are no more than a pre-determined number Q of different nodes connected to any given ring (hop count limit).

In addition, the application considered here deals with transmission networks at the regional level. In such contexts, there are generally no really stringent restrictions with respect to fiber availability or ring layout within the cable network. Therefore, the physical realization of a given "logical" SHR, as defined by the subset of demand nodes to be connected, is very often determined a posteriori, once the decision of which nodes to interconnect with that particular ring has been taken (see [6,18,19]). The following model also uses this simplifying assumption. We will therefore concentrate on the logical design (LD) aspect of the problem, that is determining which nodes should be connected by which SHR and how each individual demand flowing on the network is to be protected. The problem can then be formulated as an integer program. Let $x_{ir} = 1$ if an ADM is placed at node i connecting it to ring r and 0 otherwise, $y_r = 1$ if ring r is used and 0 otherwise, v^k_{ir} (respectively v^k_{rj}) be the integer flow variables representing the amount of traffic of commodity k that accesses (respectively leaves) ring r at node $i = o(k)$ (respectively $j = d(k)$), w^{kz}_{rs} the amount of inter-ring traffic of commodity k that passes from ring r to ring s at interconnection node z, and u_r be the capacity of ring r. The logical design of SHRs can be formulated as:

Formulation (*LD*)

$$z^*_{\mathrm{LD}} = \min \sum_{r\in R} c_r \sum_{i\in N} x_{ir} + b \sum_{r\in R} \sum_{s\in R: s\neq r} \sum_{k\in K} \sum_{z\in Z} w^{kz}_{rs}. \tag{1}$$

Flow constraints

$$\sum_{r\in R} v^k_{ir} = f^k, \quad k \in K,\ i = o(k), \tag{2}$$

$$\sum_{r\in R} v^k_{rj} = f^k, \quad k \in K,\ j = d(k), \tag{3}$$

$$v^k_{ir} + \sum_{s\in R: s\neq r} \sum_{z\in Z\cap r\cap s} w^{kz}_{sr} = v^k_{rj} + \sum_{s\in R: s\neq r} \sum_{z\in Z\cap r\cap s} w^{kz}_{rs}, \quad r \in R,\ k \in K : i = o(k),\ j = d(k). \tag{4}$$

Capacity constraints

$$\sum_{k\in K} \left(v^k_{o(k)r} + \sum_{s\in R: s\neq r} \sum_{z\in Z\cap r\cap s} w^{kz}_{sr} \right) \leqslant u_r, \quad r \in R. \tag{5}$$

Design constraints

$$\sum_{k\in K:o(k)=i} v_{ir}^k + \sum_{k\in K:d(k)=i} v_{ri}^k \leqslant \sum_{k\in K:o(k)=i \text{ or } d(k)=i} f^k x_{ir}, \quad i \in N,\ r \in R, \tag{6}$$

$$\sum_{i\in Z} x_{ir} \geqslant 2y_r, \quad r \in R, \tag{7}$$

$$\sum_{i\in N} x_{ir} \leqslant Qy_r, \quad r \in R, \tag{8}$$

$$x_{ir}, y_r \in \{0,1\}, \quad r \in R,\ i \in N, \tag{9}$$

$$u_r \in C, \quad r \in R, \tag{10}$$

$$v_{ir}^k, v_{rj}^k \geqslant 0 \text{ integer}, \quad r \in R,\ k \in K : i = o(k), j = d(k), \tag{11}$$

$$w_{rs}^{kz} \geqslant 0 \text{ integer}, \quad r,s \in R : r \neq s,\ k \in K \text{ and } z \in Z \cap r \cap s. \tag{12}$$

The objective (1) consists in minimizing the sum of ADM and inter-ring traffic costs. Constraints (2)–(4) are demand satisfaction and flow conservation constraints, constraints (5) are capacity constraints for the SHRs, and finally, constraints (6)–(8) are design constraints. Constraints (6) ensure that no demand accesses or leaves a given SHR at an origin or destination node if that node is not connected to that particular SHR, while (7) impose that each ring contains at least two interconnection nodes. Furthermore, constraints (8) ensure that there are no more than Q different nodes connected to any given ring (also termed *hop limit*). Finally, it should be stressed that the different flow variables are constrained to take integer values because they represent high order multiplexed signals that cannot be split into non-integer fractions for transmission. For instance, STM-4 signals can be split into several STM-1 but the STM-1 individual signals must stay whole.

For realistic sized instances, this model results in a huge integer program that is clearly not amenable to exact solution approaches. Indeed, there are $O(|N|R_{\max} + |K|R_{\max}^2)$ integer variables and $O((|N| + |K|)R_{\max})$ constraints.

This logical design problem induces two interdependent decision problems: (**P1**) which deals with assigning demand nodes to the rings and (**P2**) which performs the routing of the resulting inter-ring traffic. Indeed, once the ADM connections are chosen, the problem reduces to a minimum cost multicommodity flow problem.

Our solution approach exploits this natural decomposition of the problem in an iterative procedure. The solution of P1 will define the data of P2, then the solution of P2 will allow us to modify the data of P1, and so on. We will therefore successively evaluate multiple node assignments to the rings by solving multicommodity flow problems. In this context, a tabu search algorithm that explores the space of possible assignments (x_{ir} variables) seems to hold good potential.

3. A tabu search heuristic

In this section, we present a tabu search heuristic developed to solve LD. We first detail the initialization step that builds a feasible solution. Then, we describe how inter-ring traffic is determined once the topology is fixed. We approximate the optimal routing cost by solving a multicommodity flow problem on some auxiliary graph. Finally, we present the neighbourhood structure, the solution attributes and the diversification mechanism used in our tabu search algorithm.

In the following, we will restrict ourselves to two ring sizes, namely STM-4 and STM-16. In the context of our industry partner, this was sufficient. However, considering more than two ring sizes (STM-64, etc.) does not present a challenge within our methodology, it only increases the overall size of the problem and therefore computing times. In these applications, larger equipments present economies of scale: STM-16 rings have four times the capacity of an STM-4 ring while STM-16 ADMs cost less than 2 times the cost of an STM-4 ADM. In addition, inter-ring traffic requires the installation of interfaces on both rings at the interconnection nodes, but the cost of such interfaces is much smaller than that of ADMs (i.e. roughly 25 times smaller than an STM-4 ADM).

3.1. Initialization step

The initialization procedure consists in a greedy heuristic that constructs a feasible solution of LD

one ring at a time. For simplicity, we decided to build a solution without inter-ring traffic. However, to provide some free capacity in anticipation of the inter-ring traffic that will appear during the tabu search, we decided to limit the flow on a ring to a proportion α of its nominal capacity. In addition, since each ring contains at least two interconnection nodes, one is assured that demands between these nodes can be split over several rings, thus adding flexibility in view of the tabu search phase. The initialization step is outlined below.

Algorithm 1 (*Initialization*)

While there remain unassigned demands do

1. Create a new ring and assign some unassigned demands to it.
2. Fill residual capacity of that ring by adding new nodes and demands while the used capacity stays below a proportion α of the capacity of the ring and the number of nodes does not exceed Q.
3. If the used capacity is below α times the capacity of the ring, add some unassigned demands between the interconnection nodes of the ring (if any) up to α times the capacity.

End while

We now describe in detail the different steps of this algorithm.

1. *Creation of a new ring*

We first determine the node having the maximum total demand for any pair of interconnection nodes. Then, we build a ring containing these three nodes and we assign it the corresponding demands. If there is no more unassigned demand between a node and an interconnection node, we create a ring with the origin and the destination of the highest unassigned demand, and two randomly chosen interconnection nodes. If the sum of demands not yet assigned is "large enough" (i.e. greater than a proportion β of an STM-16), the ring constructed is an STM-16, otherwise it is an STM-4.

We denote by $\mathrm{dem}(i,j)$ the portion of the demand between i and j that is not yet assigned to a ring, and by $\mathrm{dem}R(i;r)$ the total amount of traffic between i and the other nodes in ring r that is not yet assigned to a ring. Given a ring r, $n(r)$ is the number of nodes connected to r and $\mathrm{usedcap}(r)$ is the capacity of ring r already used for routing. The first step of the initialization phase can be summarized as follows.

- Let $z1$, $z2 \in Z$, and $i \in N \setminus Z$ be the nodes such that

$$D^* := \mathrm{dem}(i,z1) + \mathrm{dem}(i,z2) := \max\{\mathrm{dem}(a,z) + \mathrm{dem}(a,z') : z,z' \in Z, a \in N \setminus Z\}.$$

- If $D^* \neq 0$,
  - create a ring r containing $z1$, $z2$ and i ($n(r) := 3$); if Total unassigned demand $\geqslant \beta$ *cap* (STM-16), r is an STM-16, otherwise it is an STM-4,
  - route demands $(i,z1)$ and $(i,z2)$ on r without exceeding the maximal proportion α.
- If $D^* = 0$,
  - let k be the highest demand not yet assigned,
  - choose randomly two interconnection nodes $z1$ and $z2$,
  - create a ring r containing $z1$, $z2$, $o(k)$ and $d(k)$ ($n(r) := 4$), and choose the capacity of r as above,
  - route demand k on r without exceeding maximal proportion α.
- Update $\mathrm{dem}(.,.)$, $\mathrm{dem}R(.,.)$ and $\mathrm{usedcap}(r)$.

2. *Filling capacity by adding new nodes*

Given the ring just created, we determine the node maximizing the total demand with nodes already in this ring, and we add it if we do not exceed the proportion α of capacity.

While ($\mathrm{usedcap}(r) < \alpha u_r$ *and* $n(r) < Q$), *do*

- If $S := \{i : \mathrm{usedcap}(r) + \mathrm{dem}R(i;r) \leqslant \alpha u_r\} \neq \emptyset$, then let $j := \mathrm{argmax}\{\mathrm{dem}R(i;r) : i \in S\}$. Add j to r and assign the corresponding demands to r.
- Otherwise, fill the ring with interconnection node traffic as in Step 3. If no interconnection node traffic can be routed in r, if r is

an STM-4, change it to an STM-16 and restart the whole procedure.
- If all the above fails, select the maximal unassigned demand k between two nodes not in r such that $\text{usedcap}(r) + f^k \leqslant \alpha u_r$, and assign it to r. If no such demand exists, start with a new ring.
- Update $\text{dem}(.,.)$, $\text{dem}R(.,.)$, $n(r)$ and $\text{usedcap}(r)$.

End while

3. *Filling capacity with interconnection node traffic*

If the previous operation does not fill the capacity up to a proportion α, and if there is some unassigned demand between the interconnection nodes that belong to the ring, we route this demand up to α times the capacity of the ring.

3.2. Routing and cost evaluation

Once subproblem P1 is solved, rings are determined and demand nodes are assigned to these rings. Subproblem P2 then consists in routing demands through these rings, verifying maximum ring capacity constraints and minimizing the inter-ring traffic cost. This problem reduces to a minimum cost multicommodity flow problem in an auxiliary graph presented in this section.

Recall that a demand between nodes i and j can be either routed on a ring containing both nodes i and j, or through 2 or more rings inducing then linear inter-ring traffic costs with unit cost b.

The auxiliary graph $G^* = (N^*, A^*)$ is defined as follows. The set N^* contains two types of nodes:

- for each node $i \in N$, we define two nodes in N^*, the first one i will play the role of an "origin" point and the second one i' will correspond to a "destination";
- for each ring r, we define two nodes in N^* and we denote them by r and r'.

As illustrated in Fig. 3, we consider three types of arcs.

- *Node-ring arcs*: (i, r) and (r', i') if node $i \in N$ is connected to ring r, which allow demands to access rings and exit from them at their end nodes.

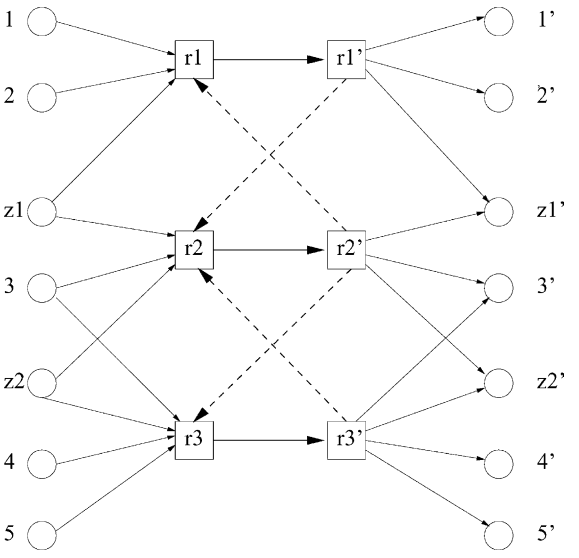

Fig. 3. Auxiliary graph G^*.

- *Ring arcs*: (r, r'), for each ring, with a maximum capacity u_r (bold arcs in Fig. 3). All the traffic routed on ring r will pass through this arc.
- *Inter-ring arcs*: (s', r) and (r', s) if rings r and s have a common interconnection node (doted arcs in Fig. 3), which allow inter-ring traffic to flow through the network.

Note that the maximum capacity of node-ring and inter-ring arcs is unlimited. For every traffic requirement between nodes i and j in N, we define an equivalent demand between nodes i and j' in N^*. Therefore, given an assignment of nodes to rings, subproblem P2 reduces to a minimum cost multicommodity flow problem in G^*, where the unit routing cost is zero for node-ring and ring arcs, while it is equal to b for inter-ring arcs. However, the existence of a feasible solution depends on the given node-ring assignment. Therefore, we add an infinite cost arc between each o–d pair with an unlimited maximum capacity. If such an arc appears in the solution to the multicommodity flow problem, it means that the given node-ring assignment is not feasible. When performing the neighbourhood evaluation of the TS procedure, such a solution will be rejected.

3.3. Basic tabu search algorithm

First introduced by Glover [9] (and independently by Hansen [14]), tabu search is a general iterative search method that has been very successful in solving numerous types of hard combinatorial optimization problems (cf. [10,11,26]). It is often referred to as a *meta-heuristic* since it uses a set of rules to control an internal local search mechanism or neighbourhood search. The principle of the method is to continue the search for better solutions even if this produces a degradation of the objective function, thus preventing the method from getting trapped in local optima. Since this can lead to cycling, tabu search uses the concept of "memory" to record the recent history of the search process and guide it away from previously visited solutions.

3.3.1. Solution space

A solution S of the RNDP problem is characterized by:

- a ring list $R(S) := \{r_1^S, \ldots, r_{n(S)}^S\}$, where r_j^S is the set of nodes connected to ring j by an ADM,
- a list of demand assignments to the rings $DA(S)$, to each demand corresponds a list of rings on which it is routed.

The total cost $c(S)$ of a solution is the sum of fixed cost $\mathrm{fc}(S)$ (ADM installation) and inter-ring traffic cost $\mathrm{tc}(S)$.

3.3.2. Neighbourhood definition

To define a neighbourhood $N(S)$ of such a solution S, we consider standard ADD/DROP moves of the ring list $R(S)$, namely:

(1) remove node i from ring r_j^S (i.e. remove the ADM installed at i to connect it to r_j^S),
(2) add node i to ring r_j^S (i.e. place an ADM to connect i to r_j^S).

Moreover, when adding a node to a ring, we need to verify that we do not exceed the maximum number of nodes per ring, and when suppressing the last node which is not an interconnection node, the ring is destroyed.

3.3.3. Evaluation of a move

We have to evaluate the effect of a move on the traffic. An optimal routing can be obtained by solving a minimum cost integer multicommodity flow problem on the auxiliary graph induced by the new node-ring assignment. However, this is an NP-hard problem; solving it at optimality would take too much time. Therefore, we do not perform the routing of all the demands, but concentrate instead on the subset of demands directly affected by the move.

If we add node i to a ring, we only modify the routing of demands whose origin or destination is equal to i. The routing evaluation is then performed by solving a minimum cost transshipment problem with one origin and multiple destinations—one for each affected demand—in the auxiliary graph described in the previous section.

If we drop node i from a ring, in order to use the residual capacity more efficiently, it is often useful to reroute the interconnection node traffic as well as the demands affected by the removal. To solve approximately this multicommodity flow problem, we use a greedy heuristic that sequentially solves a series of minimum cost transshipment problems corresponding to each node involved (origin or destination).

3.3.4. Neighbourhood evaluation

Given a current solution S, the basic step of our tabu search algorithm consists in finding the best *feasible* neighbour, that is:

$$S^* := \operatorname{argmin}\{c(S') : S' \in N(S)\}.$$

However, a complete evaluation of the neighbourhood is too time consuming. We therefore developed some criteria to reduce the size of the neighbourhood, to speed up its evaluation, as well as to improve the quality of solutions by introducing more flexibility.

As fixed costs are large compared to inter-ring traffic costs, removing a node often leads to a decrease in total cost, while adding one generally results in increased total cost. Hence, given a feasible solution S, we distinguish the neighbourhood $N^-(S)$ obtained by removing a node from a ring and the neighbourhood $N^+(S)$ obtained by adding a node to a ring. Instead of evaluating the global

neighbourhood $N(S) = N^-(S) \cup N^+(S)$, we first search neighbourhood N^- until no feasible solution is found. We then evaluate neighbourhood N^+ until feasibility is restored.

To speed up the evaluation of N^-, we decided to focus first on the more profitable moves a priori and to adopt a first improvement strategy. Hence, we first examine if it is possible to obtain a feasible solution by removing an STM-16 ADM without creating inter-ring traffic. If such a neighbour exists, we implement it right away since it is certainly one of the best solutions in the neighbourhood. Then we look for STM-16 removals that create inter-ring traffic, and implement the best one if such a feasible move exists. Otherwise, we evaluate if a feasible solution can be found by removing an STM-4 ADM without creating inter-ring traffic. Again, we stop as soon as we find one. If no such solutions are found, we then select the best removal among those that generated inter-ring traffic.

Infeasible solutions obtained while evaluating N^- are not considered. Moreover, when searching neighbourhood N^-, if removing a node from a ring leads to an infeasible solution, it is highly probable that this modification will stay infeasible while moves are restricted to removals. Therefore, these moves are stored in an unlimited tabu list until the end of this phase. When a new node is added, the list is emptied. We denote by $TL2$ this tabu list.

During the second phase of the neighbourhood evaluation process (i.e. for N^+), the best choice for adding a node is always an STM-4 ADM since it is more economical than an STM-16. Nevertheless, it turns out that this kind of move generally keeps the search within the same region of the solution space. To avoid this drawback and speed up the search, we perform a partial evaluation of N^+: for STM-16 all possible locations are tested while for STM-4 each node is added into only one randomly selected ring (instead of all possible rings).

3.3.5. Tabu mechanism

To prevent the procedure from cycling, the move, characterized by a pair (i, r_j^S), leading to the best neighbour is declared tabu for a number T of iterations. We denote by $TL1$ this tabu list.

3.3.6. Stopping criterion

We use two stopping criteria: a maximum number of iterations and a maximum computing time.

3.3.7. Tabu search algorithm

The tabu search algorithm is outlined below.

Algorithm 2

1. Initialization
   - Let S^0 be the feasible solution obtained with Algorithm 1.
   - Let $t := 0$, $S^* := S^0$, $TL1 := \emptyset$ and $TL2 := \emptyset$.
2. Neighbourhood evaluation

   While the stopping criterion is not satisfied, do
   - *Phase* 1
     - Evaluate neighbourhood $N^1(S^t)$ (except moves in $TL2$), stop if a move does not create inter-ring traffic.
     - If any solution is infeasible, update $TL2$.
     - If no feasible solution was found, empty $TL2$ and go to Phase 2. Otherwise, apply Procedure 1 UPDATE($N^1(S^t)$, t, $TL1$, S^*) and go to Phase 1.
   - *Phase* 2
     - Evaluate neighbourhood $N^2(S^t)$.
     - Apply Procedure 1 UPDATE($N^2(S^t)$,t, $TL1$,S^*) and go to Phase 1.

   End while

Procedure 1 (UPDATE$(N, t, TL1, \mathbf{S}^*)$)
- Let S^{t+1} be the best feasible solution in N which is not tabu.
- If $c(S^{t+1}) < c(S^*)$ then $S^* \leftarrow S^{t+1}$.
- Update $TL1$.
- $t \leftarrow t + 1$.

3.4. Diversification phase

An important ingredient for local search efficiency is diversification. The aim of diversification is to guide the search process away from regions that have already been explored and into unex-

plored and hopefully promising areas of the solution space (see [11,25] for more details).

In our particular case, the basic search mechanism used by the tabu search algorithm just described cannot easily modify the number of rings present in a solution (i.e. it has to either empty a ring one node at a time or create a new ring having a single demand node and two interconnection nodes which is generally not a very attractive move costwise) and does not consider possible changes in the capacity of a given ring. The diversification procedure we propose enables the overall procedure to circumvent those limitations by evaluating the possible merger of two existing rings (possibly increasing the capacity of the resulting ring) or the splitting of an existing ring into two new rings (again possibly changing the capacity). These types of more radical moves should in turn help the search process reach new regions of the solution space.

The main ideas that drive these modifications are the following:

1. If for a given ring r, a subset r_1 of its nodes has only a small amount of traffic demand in common with the nodes in $r_2 := r \setminus r_1$, then splitting the ring into two smaller rings r_1 and r_2 may lead to a solution where the overall capacity required is smaller, therefore reducing the fixed cost without significantly increasing inter-ring traffic.
2. On the other hand, if two rings generate a large amount of inter-ring traffic, it could then be more economical to merge these rings into a single larger ring to prevent unnecessary inter-ring traffic to use up too much of the available capacity on the network.

The objective in splitting a ring in two is to minimize the inter-ring traffic generated. It is easy to see that this corresponds to solving a minimum-cut problem in a complete graph where the nodes correspond to demand nodes in the ring and each edge has a capacity equal to the demand between the nodes routed in the ring. However, it often turns out that the global minimum cut in this graph is given by a set containing only one node, while we would prefer to have a more balanced split. To avoid this drawback, we use the following heuristic: a minimum cut is computed for each pair of nodes using Gomory and Hu's algorithm [13], and we select the minimum cut corresponding to an edge not incident to a leaf in the Gomory–Hu tree. If no such cut exists however (meaning the Gomory–Hu tree is a star), the global minimum cut is chosen. We replace the ring by two new rings formed with the nodes on each side of the cut together with the interconnection nodes of the original ring. If it is feasible, we choose the two rings to be STM-4. Otherwise, we select the best combination of an STM-4 and an STM-16, and if none is feasible, both new rings are STM-16.

For merging, we select a pair of rings such that the sum of used capacities in both rings does not exceed an STM-16, the number of different nodes belonging to the pair does not exceed the threshold Q, and the interconnection nodes are the same in both rings. We then replace the original rings by a new one connecting all their nodes and we reroute on it all the corresponding demands and inter-ring flow. If the resulting flow is less than the capacity of an STM-4, the new ring is an STM-4, otherwise it is an STM-16.

The complete diversification procedure is summarized hereafter and can be seen as an additional last step within the *While* loop in Algorithm 2.

Algorithm 3 (*Diversification procedure*)

If best solution unimproved for iter_div iterations then do

1. Try splitting each STM-16 ring in best solution and select best move, even if it results in a deterioration of solution quality (in order to have a minimal diversifying effect).
2. Repeat splitting step as long as it produces improvements (descent towards a local minimum with respect to the splitting neighbourhood).
3. Evaluate all ring pair mergers (as described above) and select best merge move. Implement the move if it improves the solution.
4. Repeat merge step as long as it produces improvements.
5. This final solution is set as the current solution and the basic tabu search resumes from it.

4. Computational results

In this section we present and discuss the numerical experiments that were performed to calibrate and evaluate the TS algorithm presented in Section 3. We first describe the problem instances and then the specific experiments. All programs were implemented using the C++ computing language and were run on a PC with a Pentium III 466 MHz processor running under Linux.

The set of test problems we considered contains *real* as well as *randomly* generated instances. Our industry partner provided a set of "slightly perturbed" real instances. These 35 instances are grouped into 3 families according to the number of demand nodes ($n = 18$, 32, 48). All have exactly two interconnection nodes ($zc = 2$). They are identified in Table 1 by the indicator real(n, zc).

In addition, we developed a generator that produces realistic random instances. Recall that in our context the physical realization of the rings is not considered to be restrictive, hence the data required to describe an instance simply consists in a symmetric demand requirement matrix. In real instances, there are usually three rather distinct levels for demand values: the largest values correspond to demands between interconnection nodes (i.e. zonal centers); the middle level corresponds to requirements between interconnection and *normal* demand nodes; the lowest demands are those between pairs of normal demand nodes with eventually several of them equal to zero.

To generate realistic random instances, we first specify the three parameters characterizing the network to be generated, that is the number of demand nodes n, the number of interconnection nodes zc, and the proportion p of all possible pairs of normal demand nodes for which there will be a nonzero demand. Note that it is considered here that the only demands that can be zero are those between normal demand nodes. All other possible demands (i.e. between pairs of interconnection nodes and interconnection and normal demand nodes) are nonzero. Then we associate to each node i an integer value $D_i \in [10, 15]$ if i is an interconnection node, and $D_i \in [5, 10]$ otherwise. The demand between nodes i and j is then set equal to $D_i * D_j / r_{ij}$ rounded to the nearest integer and where r_{ij} is randomly generated between 2 and 10. We considered the different parameter combinations obtained with: $n = 20$, 30, 40; $zc = 2$ or 3; and $p = 0.1$, 0.3 and 0.5. For each combination, five different instances were generated thus providing us with 18 series of 5 instances each for a total of 90 random test problems.

The whole set of instances is summarized in Table 1. For each series of instances, we first give its identificator and then list the number of nodes, the number of interconnection nodes, the proportion of nonzero demands of the third type, the number of instances and the problem type (*real* or *random*).

A preliminary set of experiments was carried out to determine appropriate values for the different parameters of the procedure:

- the proportion threshold β which determines if a ring is an STM-16 or STM-4 in the initial solution,
- the proportion α which limits the flow on a ring in the initial solution,
- the tabu list size T,
- the maximum number *iter_div* of iterations without improvement before calling the diversification procedure.

The greedy heuristic (i.e. Algorithm 1) constructs a feasible solution without any inter-ring

Table 1
Test problem characteristics

Series	n	zc	p	# Instances	Type
real$(18, 2)$	18	2		11	Real
real$(32, 2)$	32	2		18	Real
real$(48, 2)$	48	2		6	Real
data(n, zc, p)	20, 30, 40	2, 3	0.1, 0.3, 0.5	$18 * 5$	Random

traffic. When considering the greedy heuristic in isolation, the best results are of course obtained by setting α to 1 so that no portion of the ring capacity is set aside for possible inter-ring traffic (i.e. rings are filled as much as possible). As for the proportion threshold β controlling the size of the ring being created, several values were tested. Low proportions (i.e. 0.3 or 0.5) tend to create more STM-16 rings, however these cannot be adequately filled afterwards with the simple constructive steps of the heuristic. Higher values of β (i.e. 0.7, 0.9 or 1) on the other hand, reduce the number of STM-16 rings and provide better capacity utilization in general. The best combination was clearly $\alpha = 1$ and $\beta = 0.9$.

In the context of the tabu search procedure, the initial solution needs to have some unused capacity in order to accommodate the inter-ring traffic that will be generated when exploring different alternate solutions. Setting $\beta = 0.7$ seemed to provide the required flexibility and this was then fixed for the rest of our experiments. We then tested several strategies for setting parameters α and T. Two stopping criteria were used simultaneously: a maximum number of iterations (1000) and a maximum computing time (1 hour) which was never reached. In fact, even the largest instances ran within 30 minutes of CPU time.

Results are summarized in Table 2. This table reports mean improvements, in percentage points, of solutions found by the basic tabu search procedure relative to the best solutions produced by the greedy heuristic. All values in a line are averages over all instances in the corresponding series. For example, the line data(20) contains averages over all random instances with 20 nodes.

As can be seen from this table, the best results were obtained with parameters (T, α) set to $(7, 0.9)$. It appears that, given a fixed value of α, the procedure is rather robust with respect to the tabu list size producing solutions of very similar quality for the different tabu list lengths. When analyzing the results in details, we observe that $T = 7$ is already sufficient to avoid cycling. Furthermore, for the same number of iterations, $T = 7$ allows to evaluate more solutions than $T = 15$, which explains why setting $T = 7$ or 10 gives slightly better results.

Table 2
Adjustment of parameters T and α (improvement %)

(T, α)	$(5, 0.5)$	$(5, 0.7)$	$(5, 0.9)$	$(5, 0.95)$
real(18, 2)	−13,64	−15,91	−6,82	−4,55
real(32, 2)	−0,47	2,16	2,80	2,61
real(48, 2)	11,42	15,93	16,33	17,60
Real	−2,57	−1,16	2,10	2,93
data(20)	23,41	23,60	24,27	24,16
data(30)	30,68	32,51	32,21	31,36
data(40)	36,79	39,20	38,51	38,46
Random	30,29	31,77	31,66	31,33
Average	21,09	22,55	23,39	23,38
(T, α)	$(7, 0.5)$	$(7, 0.7)$	$(7, 0.9)$	$(7, 0.95)$
real(18, 2)	−13,64	−13,64	−4,55	−4,55
real(32, 2)	−1,00	1,26	2,67	2,53
real(48, 2)	13,68	14,79	16,43	17,87
Real	−2,45	−1,10	2,76	2,94
data(20)	22,34	22,73	24,17	23,82
data(30)	29,52	32,60	32,19	30,90
data(40)	36,74	38,22	38,55	38,62
Random	29,53	31,18	31,64	31,11
Average	20,58	22,14	23,55	23,22
(T, α)	$(10, 0.5)$	$(10, 0.7)$	$(10, 0.9)$	$(10, 0.95)$
real(18, 2)	−16,48	−11,36	−2,27	0,00
real(32, 2)	−0,95	1,53	2,73	2,35
real(48, 2)	10,09	14,54	15,88	17,69
Real	−3,94	−0,29	3,41	4,24
data(20)	21,41	22,21	22,86	23,76
data(30)	30,53	32,33	31,40	30,55
data(40)	37,49	38,68	38,19	38,20
Random	29,81	31,07	30,82	30,84
Average	20,36	22,29	23,14	23,39
(T, α)	$(15, 0.5)$	$(15, 0.7)$	$(15, 0.9)$	$(15, 0.95)$
real(18, 2)	−15,07	−13,64	−6,82	−4,55
real(32, 2)	−2,41	1,47	2,52	2,55
real(48, 2)	11,71	14,60	16,26	17,18
Real	−3,97	−1,03	1,94	2,83
data(20)	20,26	21,36	22,69	21,60
data(30)	30,30	31,53	30,84	29,87
data(40)	36,38	38,21	38,27	37,49
Random	28,98	30,37	30,60	29,65
Average	19,75	21,58	22,58	22,14

From Table 2, it is also clear that the value of α has an important impact on solution quality. For any given tabu list length, results are superior with high values of α, that is when rings in the initial solution are fuller. Clearly, the performance of the basic tabu search heuristic depends rather significantly on the initial solution. Moreover, for some small sized real instances, the solution it produces is inferior to the best solution found by the greedy heuristic.

As described in the previous section, the observation that the basic tabu search neighbourhood did not allow for easy changes in the number of rings composing a solution or to the capacity of a given ring motivated the introduction of a diversification procedure. To evaluate the impact of this procedure we carried out another set of experiments in which we fixed $T = 7$ (i.e. the best value according to the previous results) and considered two settings $\alpha = 0.7$ and 0.9. For each combination we then tested several values of *iter_div* (i.e. 30, 50 and 70 iterations without improvement). The results are summarized in Table 3 which again reports mean improvements, in percentage points, relative to the best greedy heuristic solutions. All values in a line are averages over all instances in the corresponding series. Note that to keep the table to a reasonable size, we do not report the individual results for each of the 18 data(n, zc, p) series but present instead averages for each network parameter value over all series having that value in common. For example, the line data$(n, zc, 0.1)$ contains the averages over all random instances having a proportion $p = 0.1$ of nonzero demands of the third type. The two columns under the heading *Tabu* reproduce the results obtained by the basic tabu search algorithm (i.e. without the diversification procedure) for $\alpha = 0.7$ and 0.9 as they appear in Table 2.

From Table 3, one can see that introducing the diversification mechanism significantly improves performance qualitywise. Even though for the series of larger real instances (i.e. real(48,2)) and those of randomly generated instances with the sparsest demand requirement matrices (i.e. with $p = 0.1$) the diversification did not improve the average solution quality and even resulted in inferior solutions for some instances, its overall im-

Table 3
Solution quality for basic tabu search and diversification procedure for various parameter settings (improvement over greedy heuristic in %)

	Tabu		Diversification					
α	0.7	0.9	0.7			0.9		
iter_div			30	50	70	30	50	70
real(18, 2)	−13.64	−4.55	0.00	0.00	0.00	0.00	0.00	0.00
real(32, 2)	1.26	2.67	7.95	6.87	7.65	7.56	7.29	7.60
real(48, 2)	14.79	16.43	11.83	11.68	12.61	12.23	12.26	12.30
Real	−1.10	2.76	6.11	5.54	6.09	5.98	5.85	6.02
data(20)	22.73	24.17	26.77	26.55	27.08	26.53	27.33	26.30
data(30)	32.60	32.19	35.90	35.91	35.94	36.09	35.65	35.92
data(40)	38.22	38.55	41.93	41.56	41.43	41.88	41.20	40.91
data$(n, 2)$	28.88	29.01	30.28	29.66	30.03	30.08	30.16	29.56
data$(n, 3)$	33.49	34.27	39.46	39.70	39.60	39.59	39.30	39.19
data$(n, zc, 0.1)$	24.85	26.89	17.47	17.22	17.59	17.66	17.47	17.07
data$(n, zc, 0.3)$	35.27	35.01	38.96	39.00	38.99	38.92	38.97	38.72
data$(n, zc, 0.5)$	33.43	33.02	48.16	47.81	47.87	47.92	47.75	47.33
Random	31.18	31.64	34.87	34.68	34.82	34.83	34.73	34.38
Average	22.14	23.55	26.82	26.52	26.77	26.76	26.64	26.43

pact is still clearly positive generating significant improvements in 90 out of 125 instances. Moreover, for the small sized real instances, it succeeds in matching the results obtained by the best greedy heuristic, something the basic tabu search procedure could not accomplish on its own. The improvement in solution quality is more significant for random instances with 3 interconnection nodes and for those with the largest proportion of non-zero demands ($p = 0.5$) which are the largest and more difficult instances in our test set.

As for the different settings for parameters α and *iter_div*, results are quite similar generating improvements in solution quality of about 26% on average over all instances. Again these results tend to confirm the robustness of the overall approach with respect to parameter settings. Nevertheless, it seems that calling diversification at a higher frequency leads to slightly better results.

Finally, as no lower bounding procedure is currently available for this problem, it is not possible to assess exactly the quality of the solutions found by the tabu search procedure. However, attaining over 26% improvement on average over a greedy constructive heuristic similar to the ones found in the industry is very encouraging and was deemed quite satisfactory by our industry partner.

5. Conclusion

In this paper, we propose a tabu search heuristic for the design of SHRs in a zonal network. The model studied comes from a real application with operational constraints such as dual homing and limited number of nodes per ring.

The algorithm is based on simple moves such as adding or removing a single node from a ring. Moreover, computational results presented on both real and randomly generated instances show the great benefit obtained by applying a diversification phase that modifies the best solution found using a more elaborate set of moves based on splitting rings in two and on merging existing rings.

This tabu search heuristic has demonstrated its efficiency by improving significantly upon the best solutions that could be found with a greedy constructive heuristic over a large set of real and randomly generated instances.

Acknowledgements

The authors wish to thank two anonymous referees for their constructive comments. Financial support and real instances for this research were provided by the Belgian telecommunication society, Belgacom. Additional financial support was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) under grants OGP0184121, 01-ER-3254, and 97-NC-1654. Their support is hereby gratefully acknowledged.

References

[1] K. Altinkemer, Topological design of ring networks, Computers and Operations Research 21 (1994) 421–431.
[2] K. Altinkemer, B. Kim, Heuristics for ring network design, Working paper, Krannert Graduate School of Management, Purdue University, 1994.
[3] M. Armony, J.G. Klincewicz, H. Luss, M.B. Rosenwein, Design of stacked self-healing rings using a genetic algorithm, Journal of Heuristics 6 (1) (2000) 85–106.
[4] J. Baudron, A. Khadr, F. Kocsis, Availability and survivability of SDH networks. Alcatel Electrical Communications, 1993, 4th Quarter, pp. 339–348.
[5] S. Cosares, I. Saniee, An optimization problem related to balancing loads on SONET rings, Telecommunication Systems 3 (1994) 165–181.
[6] S. Cosares, D.N. Deutsch, I. Saniee, O.J. Wasem, SONET toolkit: A decision support system for designing robust and cost-effective fiber-optic networks, Interfaces 25 (1995) 20–40.
[7] M. Dell'Amico, M. Labbé, F. Maffioli, Exact solution for the sonet ring loading problem, Operations Research Letters 25 (1999) 119–129.
[8] M. Di Lascio, A. Gambaro, U. Mocci. Protection Strategies for SDH Networks. Proceedings of the 6th International Network Planning Symposium—Networks'94, Budapest, Hungary, 1994, 4–9 September, pp. 387–392.
[9] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 5 (1986) 533–549.
[10] F. Glover, E. Taillard, D. de Werra, A user's guide to tabu search, Annals of Operations Research 41 (1993) 3–28.
[11] F. Glover, M. Laguna, Tabu Search, Kluwer, Norwell, MA, 1997.

[12] O. Goldschmidt, A. Laugier, E.V. Olinick, SONET/SDH ring assignment with capacity constraints. INFORMS Meeting, Montreal, Canada, 1998, 26–29 April (also working paper, Department of IE and OR, University of California, Berkeley).
[13] R.E. Gomory, T.C Hu, Multi-terminal network flows, SIAM Journal on Applied Mathematics 9 (1961) 551–570.
[14] P. Hansen, The steepest ascent mildest descent heuristic for combinatorial programming. Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986.
[15] J.L. Kennington, V.S.S. Nair, M.H. Rahman, Optimization based algorithms for finding minimal cost ring covers in survivable networks. Report 97-CSE-12, Department of Computer Science and Engineering, Southern Methodist University, 1997.
[16] J.G. Klincewicz, H. Luss, D.C.K. Yan, Designing tributary networks with multiple ring families. Working paper AT&T Labs, Holmdel, NJ 07733, Computers and Operations Research 25 (12) (1997) 1145–1157.
[17] M. Labbé, G. Laporte, P. Soriano, Covering a graph with cycles, Computers and Operations Research 25 (1998) 499–504.
[18] M. Laguna, Clustering for the design of SONET rings in interoffice telecommunications, Management Science 40 (1994) 1533–1541.
[19] H. Luss, M.B. Rosenwein, R.T. Wong, Topology network design for SONET ring architecture, IEEE Transactions on Systems, Man and Cybernetics 28 (06) (1998) 780–790.
[20] Y.-S. Myung, H.-G. Kim, D.-W. Tcha, Optimal load balancing on SONET bidirectional rings, Operations Research 45 (1997) 148–152.
[21] G. Pesant, P. Soriano, An optimal strategy for the constrained cycle cover problem. Research report CRT 98-51, Centre de recherche sur les transports, Université de Montréal, Annals of Mathematics and Artificial Intelligence, in press.
[22] P. Semal, K. Wirl, Optimal clustering and ring creation in the network planning system PHANET. Proceedings of the 6th International Network Planning Symposium-Networks'94. Budapest, Hungary, 4–9 September 1994, pp. 303–308.
[23] A. Schrijver, P.D. Seymour, P. Winkler, The ring loading problem, SIAM Journal of Discrete Mathematics 11 (1) (1998) 1–14.
[24] J.B. Slewinsky, W.D. Grover, M.H. MacGregor, An algorithm for survivable network design employing multiple self-healing rings, Proceedings of IEEE GLOBECOM'93, 1993, pp. 1568–1573.
[25] P. Soriano, M. Gendreau, Diversification strategies in tabu search algorithms for the maximum clique problem, Annals of Operations Research 63 (1996) 189–207.
[26] P. Soriano, M. Gendreau, Fondements et applications des méthodes de recherche avec tabous, RAIRO 31 (2) (1997) 133–159.
[27] P. Soriano, C. Wynants, S. Séguin, M. Labbé, M. Gendreau, B. Fortz, Design and dimensioning of survivable SDK networks, in: B. Sansò, P. Soriano (Eds.), Telecommunications Network Planning, Kluwer, Norwell, MA, 1998, pp. 149–169.
[28] J. Sosnosky, T.H. Wu, SONET ring applications for survivable fiber loop networks, IEEE Communications Magazine (June) (1991) 51–58.
[29] A. Sutter, J.L. Fullsack. SDH network planning in a changing environment, Proceedings of the 6th International Network Planning Symposium-Networks'94. Budapest, Hungary, 4–9 September 1994, pp. 149–154.
[30] A. Sutter, F. Vanderbeck, L.A. Wolsey, Optimal placement of add/drop multiplexers: Heuristic and exact algorithms, Operations Research 46 (5) (1998) 719–728.
[31] O.J. Wasem, An algorithm for designing rings for survivable fiber networks, IEEE Transactions on Reliability 40 (1991) 428–432.
[32] O.J. Wasem, R.H. Cardwell, T.H. Wu. Software for designing survivable sonet networks using self-healing rings. Proceedings of IEEE International Conference on Communication (ICC'92), 1992, pp. 425–431.
[33] O.J. Wasem, T.H. Wu, R.H. Cardwell, Survivable SONET networks—design methodologies, IEEE Journal on Selected Areas in Communications 12 (1994) 205–212.
[34] T.H. Wu, D.J. Kolar, R.H. Cardwell, Survivable network architectures for broad-band fiber optic networks: Models and performance comparison, IEEE Journal of Lightwave Technology 6 (1988) 1698–1709.
[35] T.H. Wu, Fiber Network Service Survivability, Artech House, Boston, MA, 1992.