Comparing Metaheuristic Algorithms for the SONET Network Design Problems

Comparing Metaheuristic Algorithms for Sonet Network Design Problems

Roberto Aringhieri
DTI, University of Milan
Via Bramante 65, 26013 Crema - Italy

Mauro Dell'Amico*
DISMI, University of Modena and Reggio Emilia
Viale A. Allegri 13, 42100 Reggio Emilia - Italy

Second revision - November 12, 2004

Abstract

This paper considers two problems that arise in the design of optical telecommunication networks when a ring-based topology is adopted, namely the SONET Ring Assignment Problem and the Intraring Synchronous Optical Network Design Problem. We show that these two network topology problems correspond to graph partitioning problems with capacity constraints: the first is a vertex partitioning problem, while the latter is an edge partitioning problem. We consider solution methods for both problems, based on metaheuristic algorithms. We first describe variable objective functions that depend on the transition from one solution to a neighboring one, then we apply several diversification and intensification techniques including Path Relinking, eXploring Tabu Search and Scatter Search. Finally we propose a diversification method based on the use of multiple neighborhoods. A set of extensive computational results is used to compare the behaviour of the proposed methods and objective functions.
Keywords: Metaheuristics, SONET ring, Optical Networks, Graph Partitioning

1 Introduction

In the last ten years the widespread adoption of internet technology and its integration into the international communications infrastructure has drastically

*corresponding author, email: dellamico@unimore.it

changed the communications landscape. In these years the number of users of internet-based applications has exponentially increased and new sophisticated applications have been introduced. As a consequence the request for transmission capacity, or bandwidth, has greatly increased.

Fiber optics are the current technological solution allowing for the fast transmission of large quantities of data through telecommunication networks. Several communications can be transmitted at the same time on the same fiber through multiplexing techniques (namely the Wavelength Division Multiplexing). The current standard for optical networks is denoted as SONET (Synchronous Optical NETwork) and it is concerned with a *ring*-based topology. More specifically, each customer is connected to one or more rings and the entire network is made up of a collection of such rings. The choice of assigning a customer to a single ring or to multiple rings, and the way the rings are connected, determine different designing issues.

Each customer uses an add-drop-multiplexer (ADM) to send/receive transmissions to/from a ring. The ADMs are associated with the *nodes* of the rings. Each node has exactly two links connecting it to its two neighboring nodes on the ring. The links have duplicated fibers to allow a bidirectional transmission. When a failure occurs on a link (i, j) the ring topology is used to recover this failure by transmitting the traffic originally sent on the link on the surviving part of the ring. Hence, the capacity of any link of a bidirectional ring, say B, has to accommodate the bandwidth request for the transmission toward all other nodes.

In this paper we consider two basic designing techniques determining different network topologies, and we propose optimization algorithms for each of them.

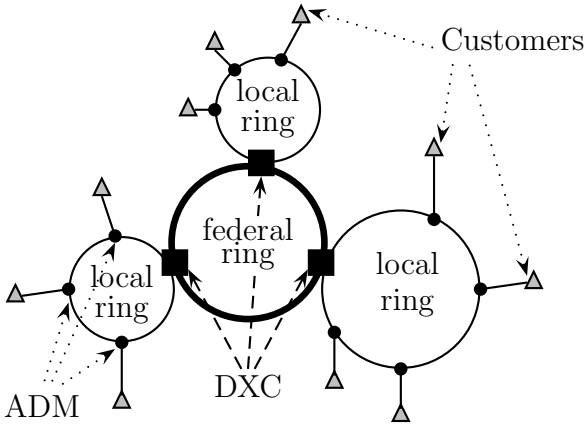


Figure 1: A SONET network with DXC

In the first topology the customers set is partitioned into subsets, each of which is associated with a *local ring*. The local rings are connected to a wider ring called *federal ring* used to transit the *inter-ring* traffic. Each node of the federal

ring is assigned a digital cross connector (DXC), i.e., a special device allowing two different rings to exchange transmissions. Since a DXC is the most costly network component, a topology with the smallest number of rings is preferred. An example of such a topology is depicted in Figure 1. The optimization problem associated with this topology consists of minimizing the number of rings (i.e., the total number of DXCs) in such a way that: i) each customer is connected to exactly one ring; and ii) the maximum capacity of each ring is bound by the common value B. This problem is usually called *SONET Ring Assignment Problem* (SRAP) *with capacity constraint*.

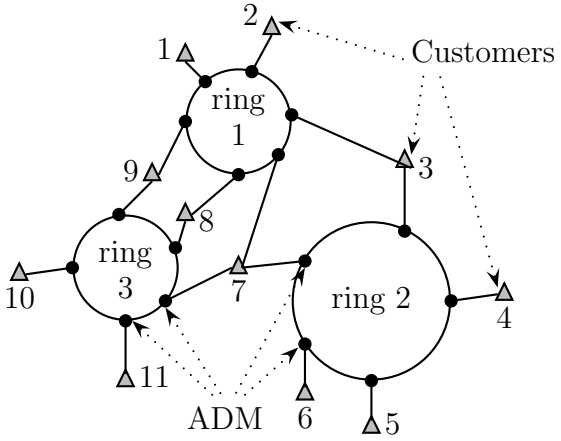


Figure 2: A SONET network without DXC

When the network has to connect customers located in a restricted area, a second topology is possible. Each ring is designed in such a way that it transports only the traffic among its own customers. To satisfy this requirement it may be necessary that a single customer is connected to more than one ring. In Figure 2 we suppose that customers 8 and 9 have to communicate with customers 1, 2, and 10; hence, they have to be connected both to ring 1 and ring 3. Customer 3 has to be connected to ring 1 and ring 2, whereas customer 7 has to be connected to all three rings. This topology generally uses more ADMs than the first one, but no DXC at all is required. Hence, the total cost of the second network could be lower than that of the first one.

The optimization problem associated with the second topology consists of minimizing the total number of ADMs in such a way that: i) each pair of customers needing to communicate with each other has to be connected to the same ring; ii) the maximum capacity of each ring is bounded by the common value B. This problem is called *Intraring Synchronous Optical Network Design Problem* (IDP).

The above problems are known to be $\mathcal{NP}$-hard (see [8,9] for details and proofs). The aim of this paper is to provide and compare several algorithms for solving

SRAP and IDP. These algorithms are based on Tabu Search and Scatter Search methodologies (see e.g. [4, 6, 12] for a general introduction to these topics) and exploit different intensification and diversification techniques. Moreover, we will introduce a *variable objective function* driving the search from unfeasible to feasible solutions, and a diversification technique based on the strategic use of *multiple neighborhoods*. This work extends the seminal ideas introduced by Aringhieri, Dell'Amico and Grasselli [1] for the solution of SRAP, and applies them to IDP.

The paper is organized as follows: In Section 2 we introduce graph theory models for the two problems, while previous works on SRAP and IDP are briefly resumed in Section 3. The basic ingredients for developing Local Search algorithms are introduced in Section 4, different intensification and diversification strategies are then described in Section 5, and Section 6 reports extensive computational results comparing the various approaches on benchmark instances, both from the literature and new ones proposed in this study. Section 7 concludes the work.

2 Models

In this section a model based on graph theory is proposed for each problem.

Consider a set of n customers and a symmetric *traffic matrix* $[d_{uv}]$ $(u, v = 1, \ldots, n, u \neq v)$ where each entry gives the amount of traffic between customer u and v. We consider an undirected graph $G = (V, E)$ where the node set V contains one node for each customer and the edge set E has an edge $[u, v]$ for each pair of customers u, v such that $d_{uv} > 0$ (remind that $d_{uv} = d_{vu}$, due to the symmetry of the traffic matrix). Given a subset of edges $\widetilde{E} \subset E$, let $V(\widetilde{E}) \subseteq V$ be the set of terminal nodes of the edges in $\widetilde{E}$.

Problems SRAP and IDP correspond to two different partitioning of the above graph, subject to capacity constraints. In particular, SRAP involves a node partitioning, whereas IDP, an edge partitioning (the state of the art algorithms for such partitioning problems are those presented in [8, 9]).

SRAP *partitioning problem*

Given a partition of V into k subsets $V_1, V_2, \ldots V_k$, the corresponding SRAP network is obtained by defining k local rings and a federal ring, as follows. All the customers of subset V_i are associated to the i-th local ring by means of $|v_i|$ ADMs, while the federal ring uses a DXC to connect each local ring. So the resulting network uses n ADMs and k DXCs.

Solving SRAP corresponds to finding the partition $V_1, \ldots V_k$ minimizing k, and

such that

$$\sum_{u \in V_i} \sum_{\substack{v \in V \\ v \neq u}} d_{uv} \leq B, \quad i = 1, \ldots, k \tag{1}$$

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \tag{2}$$

Constraints (1) impose that the total traffic on each ring i, that is, the sum of the traffic internal to i plus the traffic from i to the other rings, does not exceed the bound B. Constraint (2) impose that the total traffic on the federal ring is not larger than the bound B.

IDP *partitioning problem*

Given a partition of E into k subsets $E_1, E_2, \ldots E_k$, the corresponding IDP network can be obtained by defining k rings and connecting each customer of $V(E_i)$ to the i-th ring by means of an ADM. The resulting network uses $\varphi = \sum_{i=1}^{k} |V(E_i)|$ ADM and no DXC.

Solving IDP corresponds to finding the partition $E_1, \ldots E_k$ minimizing φ and such that

$$\sum_{[u,v] \in E_i} d_{uv} \leq B, \quad i = 1, \ldots, k \tag{3}$$

Constraints (3) assure that the traffic inside each ring does not exceed the bound B.

We now introduce some notation necessary to simplify the presentation. Given a subset of edges $\widetilde{E} \subseteq E$, we denote with $d(\widetilde{E})$ the sum of the weights of the edges of $\widetilde{E}$ (i.e. $d(\widetilde{E}) = \sum_{[u,v] \in \widetilde{E}} d_{uv}$). Given two disjoint subsets of nodes $V_1 \subset V$, $V_2 \subset V$ ($V_1 \cap V_2 = \emptyset$), let $\delta(V_1, V_2) = \{[u, v] \in E : u \in V_1, v \in V_2\}$ denote the set of edges in the cut separating V_1 from V_2. If $V_1 = \{u\}$ and $V_2 = V \setminus \{u\}$, we use $\delta(u)$ instead of $\delta(\{u\}, V \setminus \{u\})$. We will use the operator $argmax\{f(S)\}$ (resp. $argmin\{f(S)\}$) to return the argument of the element of set S determining the maximum (minimum) value of function f. We assume that $argmax\{\emptyset\}$ returns a NULL value. Finally, we will call *feasible* a ring that satisfies constraints (1) or (3), for SRAP or IDP, respectively.

3 Results from the Literature

Problem SRAP has been recently investigated by Goldschmidt, Laugier and Olinick [9]. It has been shown that the problem is $\mathcal{NP}$-hard and three greedy approaches, namely the *edge-based*, the *cut-based* and the *node-based* heuristics

have been proposed. The first two algorithms start their computation by assigning each node to a different ring and iteratively reduce the value of k by merging two rings, provided that the resulting ring is feasible. In the edge-based approach, the two rings connected by the maximum weight edge are first merged, while the rings corresponding to the pair V_i, V_j maximising the weight of the cut (V_i, V_j) are first chosen in the cut-based approach. Note that both methods may have different behavior if different tie break rules are used.

The third algorithm receives as input a tentative value k and it randomly assigns a node to each of the k rings. The approach disregards the capacity constraint and iteratively assigns the remaining $n-k$ nodes as follows. First the ring V_i with current largest unused capacity is selected, then the unassigned node u maximising the weight of the cut $\delta(\{u\}, V_i)$ is assigned to V_i. The algorithm is run ten times decreasing the value of k by one when a feasible solution is obtained.

The authors have tested the three greedy algorithms on a set of 160 benchmark instances with n ranging from 15 to 50 and density of the graph ranging from 5% to 72%. They first run the edge-based and cut-based heuristics, then the smallest k value obtained is used as input for the node-based heuristic. This procedure is repeated ten times by randomly breaking ties for the first two algorithms. When the best solution is feasible and k is larger than the simple lower bound

$$k_{lb} = \left\lceil \sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv}/B \right\rceil \tag{4}$$

an attempt to prove its optimality by means of CPLEX© is made.

Aringhieri, Dell'Amico and Grasselli [1] attach SRAP with metaheuristic algorithms mainly based on Tabu Search. The authors introduce an objective function that depends on the current search status, and use a strategic oscillation obtained through the swap of two neighborhoods. Comparisons among different diversification strategies applied to the benchmark instances proposed in [9] are presented: all the resulting algorithms perform better than those proposed in [9] when the same computing time is given to all algorithms.

Goldschmidt, Hochbaum, Levin and Olinick [8] consider the special case of IDP in which all edges are given the same weight. They show that this problem, hence the more general IDP, are $\mathcal{NP}$-hard. Two linear-time approximation algorithms with fixed performance guarantee are also presented.

Lee, Sherali, Han and Kim [14] study IDP with the additional constraint that the number of ADMs in each ring must not exceed a given bound, say R. They formulate the problem as a mixed-integer programming model, develop a branch-and-cut algorithm, and introduce an effective heuristic procedure, called LSHK in the sequel. The authors present computational experiments on 20 test instances with n ranging from 15 to 25 and density from 12% to 29%. The running times of the exact approach are high: 1504 seconds of a Pentium processor at 200 MHz, on

average. The heuristic LSHK defines an initial solution by constructing one ring at a time, as follows: The subset of edges corresponding to the current ring r is initialized by choosing a node u with maximum degree, with respect to the edges not yet considered, and then adding to r the edge $[u, v]$ such that v has maximum degree. The subset E_r is iteratively increased by appropriately selecting a node w such that all edges in $\delta(w, V(E_r))$ can be feasibly assigned to it. This solution is then improved through local search by moving one edge at a time from one ring to another.

Laguna [11] considers a problem that mixes the two designing techniques. Similar to problem IDP, each customer may be connected to one or more rings, but it is not required that two customers u, v with $d_{uv} > 0$ are connected to the same ring. The network may therefore need to transmit some inter-ring traffic. Unlike SRAP, the technology used to connect the rings is not specified, so an approximation of the corresponding cost, proportional to the traffic amount, is used. The resulting objective function includes both the ADMs cost and the estimated cost of the inter-ring traffic. Laguna introduces a mixed-integer model for the problem and describes a simple short-term memory Tabu Search to select integer variables configurations. The objective function corresponding to a selection is evaluated through the simplex algorithm.

4 Basic Local Search Elements

In this section we introduce the main ingredients necessary to implement Local Search algorithms for SRAP and IDP. In particular we will describe: (a) simple procedures to compute a starting solution; (b) neighborhoods; (c) data structures needed to efficiently implement the search of the neighborhood; and, (d) a simple tabu list.

4.1 Starting solution

A solution of SRAP can be computed with the three greedy methods introduced in Goldschmidt, Laugier and Olinick [9] (see Section 3). Another possibility is to start the Local Search from the very simple solution obtained by assigning each node to a different ring. Note that this solution is certainly unfeasible, since all the traffic is routed through the federal ring. However, the Local Search can easily reconduct the solution to a feasible one, as shown in our computational experiments (see Section 6).

Consider now problem IDP. An immediate method from the literature for finding a feasible solution is Algorithm $LSHK$ given in [14] applied with $R = n$. The approximation procedures presented in Goldschmidt, Hochbaum, Levin and Olinick [8], instead, have been designed for the unweighted case and cannot be adapted to our problem.

We now introduce four new heuristic procedures for solving IDP, that, in some cases, have better performances than the methods we described in the previous section. The new algorithms are particularly useful when used together with the other methods from the literature to find a good starting solution for a local search method. The first two methods are derived from the *Best-Fit Decreasing* (BFD) and *Next-Fit* (NF) procedures for the Bin Packing Problem (see e.g. Martello and Toth [15]). Our implementation of BFD considers one edge at a time, ordered by non-increasing weights, and assigns it to the ring having the smallest residual capacity so that the feasibility of the ring is preserved. If no assignment is possible a new ring is initialized containing the current edge only. Our NF procedure considers the edges sorted by non-decreasing weight and assigns the current edge to the current ring if possible, otherwise the ring is no longer considered and a new ring is initialized with the current edge.

The third method is based on the idea that good solutions should have very dense rings to save ADMs. A ring might even be a clique of graph G. Our

Algorithm 1 Clique-BF

$U := E; r := 0;$
while $(U \neq \emptyset)$ **do**
  heuristically find a clique $C \subset U$ such that $d(C) \leq B$;
  let $j :=$ $argmin\{B - d(E_i) - d(C) : i = 1, \ldots, r, B - d(E_i) - d(C) \geq 0\}$;
  if $(j =$ NULL$)$ **then** $r := r + 1; j := r$ **end if**;
  $E_j := E_j \cup C$; $U := U \setminus C$
end while

procedure *Clique-BF* (see Algorithm 1) uses a constructive greedy heuristic to iteratively select a clique of unassigned edges with total traffic not larger than B. All the demands of the clique are assigned to a single ring that minimizes the residual capacity while preserving the feasiblity, if possible, otherwise to a new ring. Note that in any non trivial instance $\max\{d_{uv}\} < B$, hence at any iteration it is always possible to identify a clique that can be feasibly assigned to the same ring. This clique might even contain a single edge. Our last heuristic procedure, called *Cycle-BF*, is similar to *Clique-BF*, but it relaxes the requirement of finding a complete clique. More specifically, at each iteration instead of looking for a complete subgraph we look for a (small) cycle with as many cords as possible. To do this we use a modified Dijkstra's shortest path algorithm. When a new node u enters the shortest path tree, the algorithm looks for a possible edges $[u, v]$ with v being an already labeled node. If such an edge exists we consider the cycle made by $[u, v]$ and by the two paths from the root to u and to v, respectively. If the traffic on the cycle is smaller than or equal to B we add the cycle to the appropriate ring and we also add to the ring all the possible cords, in a greedy fashion. If the traffic on the cycle is larger than B we continue the search with the remaining edges of $\delta(u)$ and finally we continue to grow the tree. Note that unlike *Clique-BF*, there is not guarantee that *Cycle-BF* will find a cycle that can

be entirely assigned to a ring. Hence this method could terminate with a partial solution that we will complete in a greedy way by adding one edge at a time as in BFD.

4.2 Neighborhoods

Given a solution of a generic partitioning problem, two basic neighborhoods can be obtained by implementing either of the following rules: (a) move an object from a subset to another, or (b) swap two objects assigned to two different subsets.

Following the above rule (a), Aringhieri, Dell'Amico and Grasselli [1] proposed, for problem SRAP, a neighborhood consisting of moving a node from one ring to another (including a new one), under the requirement that the receiving ring is assigned a total traffic not greater than B. In this paper we propose an extension of this neighborhood obtained by allowing to construct unfeasible solutions. The same kind of neighborhood can be used for IDP: a neighboring solution is obtained by moving an edge from a ring to another, disregarding the feasibility or unfeasibility of the resulting solution. We will refer to this neighborhood, both for SRAP and IDP, as neighborhood N_1. If we denote with $\overline{r}\,(< |V|)$ an upper bound on the maximum number of rings in an optimal solution, then the complete exploration of N_1 requires $O(\overline{r}|V|)$ time.

The general neighborhood, say N_b, determined by a complete application of rule (b) requires considering all the possible pairs of objects to be swapped, hence $O(|V|^2)$ time is necessary to explore it. Here we propose a second neighborhood, called N_2, that is the union of N_1 with a restricted version of the above general neighborhood. As for N_1 we start by moving an object (node or edge, depending on the problem) from a subset, S_1, to another subset, S_2. If the resulting solution is feasible we are done; otherwise, we try to move an object previously assigned to S_2 back to subset S_1. The resulting solution is considered a possible candidate even if it turns out to be unfeasible. The worst case time complexity of N_2 is the same of N_b, but, on average, exploring N_2 is computationally convenient. Therefore we adopted N_2 for our experiments.

4.3 Data structures

To speed up the search of the best solution in the two neighborhoods we need to use some appropriate data structures.

The edge set E is stored as a forward star and an array $WeightStar$ is used to store the total weight of the edges emanating from each node u (i.e., $WeightStar(u) = d(\delta(u))$ for all $u \in V$).

Let again $\overline{r}$ be an upper bound on the maximum number of rings in an optimal solution. For both problems we use an array $RingLoad(r)$, for $r = 1, \ldots, \overline{r}$ to store the total traffic on the ring. For SRAP, $RingLoad(r) = \sum_{u \in V_r} \sum_{v \neq u} d_{uv}$, whereas for IDP, $RingLoad(r) = d(E_r)$. For each pair u, r, with $u \in V$ and

$r = \{1, \ldots, \overline{r}\}$ we store the total weight of the edges in $E_r \cap \delta(u)$, and the cardinality of set $E_r \cap \delta(u)$ in matrices $WeightRing$ and $CardRing$. Table 4.3 summarizes the above definitions.

All these data structures expect $CardRing$ are used to check efficiently the feasibility (or unfeasibility) of a solution during the exploration of the neighborhood. Matrix $CardRing$ is used to evaluate the objective function. More specifically, if $CardRing(u, r) = 0$, then no edge emanating from u has been assigned to ring r, hence no ADM for customer u has to be installed on ring r. On the contrary, if $CardRing(u, r) > 0$, then customer u requires a single ADM on ring r, independently of the total number of edges in $\widehat{E}_{ur}$. Hence the total number of ADMs on a ring r is $|\{CardRing(u, r) > 0 : u \in V_r\}|$.

Table 1: Data structures for SRAP and IDP

Name	SRAP	IDP
$WeightStar(u)$	$d(\delta(u))$	–
$RingLoad(r)$	$\sum_{u \in V_r} \sum_{v \neq u} d_{uv}$	$d(E_r)$.
$WeightRing(u, r)$	$d(E_r \cap \delta(u))$	–
$CardRing(u, r)$	–	$\lvert E_r \cap \delta(u) \rvert$

The updating of the above structure is done as follows:

SRAP

When a node u is moved from ring r to ring s, we have to subtract from $RingLoad(r)$ the total traffic going from u to all rings different from r, so we have to compute $RingLoad(r) = RingLoad(r) - (WeightStar(u) - WeightRing(u, r))$. Similarly, we have to update the traffic on each ring $s \neq r$ computing $RingLoad(s) = RingLoad(s) + (WeightStar(u) - WeightRing(u, s))$ (for more details see [1]). Finally, $WeightRing(i, r)$ and $WeightRing(i, s)$ for $i \in V \backslash \{u\}$ have to be updated by scanning the forward star of node u.

IDP

Moving an edge $[u, v]$ from ring r to ring s imposes the following calculations: $RingLoad(r) = RingLoad(r) - d[u, v]$, $RingLoad(s) = RingLoad(s) + d[u, v]$, $CardRing(\ell, r) = CardRing(\ell, r) - 1$ for $\ell = u, v$ and $CardRing(\ell, s) = CardRing(\ell, s) + 1$ for $\ell = u, v$. Array $WeightRing$ is updated as in SRAP.

4.4 Tabu lists

Short-term memory has been implemented by using two kinds of taboos. The first one prevents a recently moved "object" (node or edge) to be moved again. The second one is less restrictive: it prevents the return of an object into the ring from which it was removed, but it allows the object to be inserted into another

ring. The moves blocked by the second taboo are a subset of the moves blocked by the first one, hence the contemporary existence of the two taboos makes sense only if the length of the list implementing the first one is strictly shorter than the length of the list used for the second one.

The length of the two lists is dynamically adapted to the evolution of the search by using the method proposed in [3]. When the trajectory in the solution space enters a promising region the list lengths ℓ_i, $i = 1, 2$, are decreased to intensify the search. On the contrary, when we encounter an unpromising region, we increase the list lengths to speed up the leaving of this region. More precisely, we define a starting tabu-tenure value $start_i$ ($i = 1, 2$), then when we detect an *improving phase* (see below), we set:

$$\ell_i = max(\ell_i - 1, \frac{1}{2} start_i), \quad i = 1, 2,$$

whilst when we detect a *worsening phase*, we set

$$\ell_i = min(\ell_i + 1, \frac{3}{2} start_i), \quad i = 1, 2.$$

We define as *improving phase* a sequence $(s_1, s_2, \ldots, s_{\Delta_i p})$ of Δip consecutive iterations lowering the objective function value (i.e., $z(s_1) > z(s_2) > \cdots > z(s_{\Delta_i p})$), whereas we call *worsening phase* a sequence of Δwp consecutive iterations in which the objective function value is not improved (i.e., $z(s_1) \leq z(s_2) \leq \cdots \leq z(s_{\Delta_i p})$). Table 2 summarizes the values of the above parameters which we used in our experiments.

Table 2: Parameters used for the adapting tabu list strategy

	SRAP		IDP	
param.	list 1	list 2	list 1	list 2
$start_i$	5	10	20	30
Δip	5	5	5	5
Δwp	3	3	3	3

4.5 Objective functions

It is known that a good evaluation function of a metaheuristic algorithm should capture, besides the value of the solution at hand, its "propensity" to lead to high quality solutions. In particular for SRAP and IDP, the simple value of the objective function gives very poor information. Consider e.g. SRAP: there are hundreds of solutions, feasible or not, that have the same number of rings but very different loads of these rings.

Let z^0 be a basic objective function counting the number of rings of a solution for SRAP, and the total number of ADMs for IDP. Moreover, let BN denote the highest (bottleneck) load of a ring. We first defined and then tested the following objective functions:

$$z^1 = z^0 + \max\{0, BN - B\},$$

$$z^2 = z^1 + \begin{cases} \alpha \cdot Ringload(r) & \text{if the last move has created a new ring } r, \\ 0 & \text{otherwise} \end{cases}$$

$$z^3 = z^0 \cdot B + BN$$

with $\alpha \geq 1$.

Before discussing the rational of the above functions it is worth recalling that the number of rings in a solution of SRAP, or the number of ADMs in a solution of IDP, is much smaller than the load of a ring, hence $z^0 \ll \min(B, BN)$.

Function z^1 minimizes the basic function z^0 while penalizing the unfeasible solutions (having $BN > B$). The idea embedded into z^2 is to add a specific penalty for moves that increase the number of rings. This penalty has been chosen as α times the weight of the new ring created by moving a single node or edge for SRAP or IDP, respectively. Function z^3 has been designed so that solutions with small z^0 are encouraged, while among solutions with the same value of z^0 the ones minimizing the bottleneck are preferred and the search is driven from unfeasible solutions toward feasible ones. For SRAP, we set α to the average number of nodes per ring , i.e., $\alpha = |V|B/d(E)$, whilst for IDP, we set $\alpha = 1$.

The last objective function z^4 we are going to introduce is an adapting technique that modifies the evaluation according to the status of the search. More specifically, z^4 is a *variable objective function* having different expressions for different transitions from the current status to the next status.

$$z^4 = \begin{cases} z^{4a} = z^0 B + BN (= z^3) & \text{(a): from feasible} \quad \text{to feasible} \\ z^{4b} = (z^0 + 1)BN & \text{(b): from feasible} \quad \text{to unfeasible} \\ z^{4c} = z^0 B & \text{(c): from unfeasible} \quad \text{to feasible} \\ z^{4d} = \beta z^0 BN & \text{(d): from unfeasible} \quad \text{to unfeasible} \end{cases}$$

with $\beta \geq 2$. This function has been designed to encourage transitions from unfeasible to feasible solutions and, within the set of feasible solutions, to choose those with smallest load.

In particular, note that an unfeasible solution has $BN > B$, hence $z^{4b} = (z^0 + 1)BN > (z^0 + 1)B \geq z^{4a}$. Moreover, $z^{4a} > z^{4c}$ and $\beta z^0 \geq z^0 + 1$, so the following ordering holds:

$$z^{4d} \geq z^{4b} > z^{4a} > z^{4c}. \tag{5}$$

Parameter β has been set to $|V|B/d(E)$ for SRAP and to 2 for IDP.

5 Intensification and Diversification Strategies

The elements introduced in the previous section have been used to implement a Basic Tabu Search, which we call BTS. In the next sections we consider general frameworks aimed at improving the performance of a local search algorithm. In particular, we address the following intensification/diversification methods: Path Relinking, eXploring Tabu Search and Scatter Search. Then we propose a new multi-neighborhood diversification technique.

5.1 Path Relinking

The first enhancing technique we tested is an implementation of the Path Relinking (PR) method [4, 7].

The basic idea in PR can be summarized as follows: select a set of moves, determine the paths in the solution space joining pairs of them, and finally, consider the solutions on these paths to continue the search.

In our tests we have considered two implementations of PR, embedding the basic tabu search BTS. For both implementations the relinking is performed when BTS has evaluated γ non improving solutions.

In the first approach, say $PR1$, we generate only the minimum length path linking the current solution to the best one (called starting and guiding solution, respectively, using the Path Relinking terminology). More specifically we determine the minimum set of moves, say MV, necessary to transform the current solution into the best solution. We construct the new starting solution for TS by applying $|MV|/2$ moves: each one is selected as the move that locally minimize the objective function.

The second implementation, say $PR2$, we maintain a set ES of *elite solution* and generate the minimum length paths that transform the current solution into the elite ones (see [13]). Using the same technique as in $PR1$ (i.e., applying one half of the moves of each path) we generate $|ES|$ new possible starting points and we select the best one. Set ES consists of the best Δ_{ES} solutions encountered in the search, where Δ_{ES} is the second parameter of this method, besides γ.

5.2 eXploring Tabu Search

The basic idea of the eXploring Tabu Search method (XTS) introduced in [3] is to use systematic jumps in the solution space based on long term memory information.

More specifically, XTS maintains a list (called *Second* list), that stores some of the *second best solution* of the explored neighborhoods. We say that a solution is second best for a neighborhood if it has the second smallest value and it is not selected to continue the search. Within each of these solutions it also stores all the parameters and other elements (e.g., tabu lists) that determine the status of

the search at the moment in which the solution was evaluated. The *Second* list is ordered by non-decreasing solution values and when the search seems to be not profitable the current solution is abandoned and the first solution in the list, say $s2$, is adopted within its associated parameters and elements. In this way the search jumps backward to the point in which $s2$ was evaluated (but not chosen) and continues with $s2$ instead of the best solution of that neighborhood. The use of the second best solution of a neighborhood is well suited for problems with an objective function with many flat regions, as SRAP and IDP have. In these cases the value of the objective function provides no information on the direction in which the search should continue to reach the global optimum. A first attempt to solve this problem was to introduce more sophisticated objective functions, as done in Section 4.5. The use of jumps to equivalent or near-equivalent solutions (the second-best solutions), proposed in the XTS framework, is another method to overcome this difficulty.

A second idea from XTS is to adopt a strategic use of a complete restart of the search. In this case one should provide a procedure that generates starting solutions uniformly distributed in the solution space. Unfortunately, this task is often as difficult as the original problem, hence a simple random restarting is adopted. (For more details on the implementation of XTS the interested reader is addressed to [2, 3]).

In [2] three methods are suggested to detect if XTS must jump to a solution from the second list:

1. the tabu status prevents all the solutions in the current neighborhood from being used;
2. the current objective function value has not been improved in the last $\overline{imp}$ iterations;
3. the global best solution has not been improved from a given number of iterations.

Three other methods are proposed to decide if recourse to a restart is necessary:

4. one of conditions 1-3 above indicates that it is necessary to jump to a solution from the *Second* list, but the list is empty;
5. after a prefixed number of iterations, counted from the last global restart, the value of the best solution found in these iterations is not close "enough" to the value of the global best solution;
6. a jump to a *Second* best solution occurred for $\#Second$ times after the last restart, without improving the best solution.

In order to simplify the parameters' tuning in this work we reduced the criteria used to control the strategy by adopting only conditions 1, 2, 4, and 6 above, all of which need only the two parameters $\overline{imp}$ and $\#Second$.

5.3 Scatter Search

In the Scatter Search (SS) methodology (see, e.g., [4] [5] [7] for a detailed treatment) a small population of solutions, called *Reference Set*, evolves through combination of its solutions. The combination must pursue two opposite objectives: intensify the search in proximity of good solutions, and diversify the search to explore a wide area. To do this the reference set is partitioned into two sets: the subset of the *high quality* solutions (HQ) and the subset of the *diverse* solutions (DV). We compute the diversity of two solutions as the number of nodes (resp. edges) for SRAP (resp. IDP) that are assigned to different rings in the two solutions. Hence, the diversity function has integer values in $[0, |V|]$ for SRAP, and in $[0, |E|]$ for IDP.

Our implementation traces the general scheme proposed in [12]; here, we report only the specific adaptation necessary for SRAP and IDP.

Diversification Generation Method

The solutions included in the first Reference Set should be drawn from all the regions of the solution space, so that they represent a significative sampling of this space. We have decided to build a random set in which the relevant differentiating element is the number of rings. We adopted the following strategies:

SRAP

Recall that we denote with k_{lb} the continuous lower bound value given in (4). We generate a first set of q random solutions, where q was fixed to $4k_{lb} - 1$ through preliminary computational experiments. Each solution s_h $(h = 1, \ldots, q)$ has exactly $h + 1$ rings and is obtained by randomly assigning each node to one of the rings. A second set of additional q solutions is then generated by perturbating each of the first q solutions through a movement of each node to a ring chosen with a uniformly random distribution. In practice each node has probability $1/h$ to be moved to the h-th ring, including its current ring. Each of the starting solutions is then optimized through a run of BTS with a limit of LS_{iter} iterations. The best RS solutions generated are selected to initialize the Reference Set.

IDP

A set of $8k_{lb} - 1$ starting solutions are generated as in SRAP, but assigning the edges instead of the nodes (Again the number of solutions to be generated was experimentally determined).

Solution Combination Method.

The solutions in the Reference Set are combined through an adaptation of the classical scoring function for SS (see e.g. [7, 12] for details). Roughly speaking, the idea is to consider a set S of solutions and the value of a variable, and to give this value a score proportional to the times it appears in these solutions, weighted with the objective function value. In detail, let $z(s)$ denote the objective function value of solution s and let x^s_{ir} be a boolean variable associated with solution s

assuming value 1 iff node i (resp. edge i) for SRAP (resp. IDP) is assigned to ring r. The score for pair (i, r) is then

$$score(i, r) = \frac{\sum_{s \in S} z(s) x_{ir}^s}{\sum_{s \in S} z(s)} \tag{6}$$

We construct a new solution assigning each node i (resp. edge i) to the ring r^* such that $score(i, r^*) = max_r\{score(i, r)\}$. To generate more solutions we adopt a scheme similar to that described in [12]. At each phase of the algorithm we select subsets S of the Reference Set with $|S| = 2, \ldots, 5$. First we consider all the 2-element subsets, then we iteratively add to each of these subsets the best not included solution until we have 5-element subsets. Each subset is used to generate a new solution.

Improvement Method.

The quality of any solution was improved by applying procedure BTS with a limit of LS_{iter} iterations. This limit was defined through some preliminary computational experiment.

Not all the objective functions described in Section 4.5 can be used with the Scatter Search approach. In particular we cannot use the functions based on the concept of 'move'. So we will use function z^1 and the following version of z^4 which includes z^3 as a special case:

$$z^4(s) = \begin{cases} z^{4a}(s) = z^0 B + BN & \text{if the solution is feasible} \\ z^{4d}(s) = \beta z^0 BN & \text{otherwhise} \end{cases}$$

Each new solution $\tilde{s}$ is inserted in the high quality set HQ if its objective function value is better than that of the worst solution in HQ. On the other hand, the new solution is inserted in Diverse (DV) set if its distance from the 'closest' solution in HQ is larger than the minimum distance between a solution in DV and one in HQ. Formally, $\tilde{s}$ enter DV when

$$\min_{s \in HQ} D(\tilde{s}, s) > \min_{s' \in DV, s \in HQ} D(s', s) \tag{7}$$

5.4 Diversification by Multiple Neighborhoods

We propose a Diversification by Multiple Neighborhhods (DMN), using more than one neighborhood to obtain a diversification in the search. More specifically, we mainly use neighborhood N_2, but sometimes we adopt a second neighborhood N_3 (to be described later) for a few moves. The idea has some similarity with the *variable neighborhood search* (VNS) method (see [10]), but in fact it is very different from it in two fundamental aspects. First of all, VNS adopts a new neighborhood when the current solution is a local optimum for the current neighborhood, whereas DMN switches to the second neighborhood when

some indicator says that the search needs to be diversified. Moreover, each of the neighborhoods used in VNS could be used alone in a local search method. Instead, the fundamental characteristic of neighborhood N_3 of DMN must be to construct solutions very different from the current one, even infeasible ones. Thus, N_3 used alone in a local search method does not provide good, or even feasible solutions.

N_3 empties a ring by moving its elements (nodes or edges, for SRAP or IDP, respectively) to the other rings while disregarding the capacity constraint and locally minimizing the objective function.

The indicator we used to devise the necessity of a switch from N_2 to N_3 is a series of at least Δ_{DMN} consecutive not improving iterations. Moreover, the neighborhoods switch is performed only if the solution at hand is feasible.

Neighborhood N_2 is immediately re-adopted after one transition with N_3. During the switch from N_2 to N_3 and again back to N_2 we continue to keep and update the tabu lists without performing any reset or re-initialization of these lists.

We conclude by noting that DMN could be seen as a very specific implementation of a *strategic oscillation* (see, e.g., [6]) in which the *critical level* depends on the evolution of the search and not on the quality of the solution, and the paths in the region below and over the critical level are obtained with neighborhoods N_2 and N_3, respectively.

6 Computational Results

The solution methods described in Section 5 have been coded into ANSI C and tested on a Pentium III/1Ghz with 256 Megabytes of core memory running under the Linux operating system. We have considered instances from the literature and newly generated ones. For each instance we have solved both the corresponding SRAP and IDP problem.

6.1 Benchmark Instances

To test the algorithms we have used three sets of instances that we call GLO, AD and $LSHK$. Set GLO has been introduced in [9] and consists of 160 instances divided into two subsets of 80 instances:

- *geometric* instances representing *natural cluster*, that is, the fact that customers mainly communicate more with their neighbors;
- *random* instances in which no preferred communication exists.

The traffic demand between two customers is drawn from a uniform random variable and gives the number of T1 lines required to serve the estimated traffic

(a T1 line has a capacity of 1.544 Mbs). Both the geometric and the random subsets have 40 *low* demand instances with traffic uniformly random in $[3, 7]$ and 40 *high* demand instances with traffic in $[11, 17]$. The instances with low demand are assigned a ring capacity $B = 155$ Mbs, whereas the high demand instances have $B = 622$ Mbs. The graphs considered have $|V| \in \{15, 25, 30, 50\}$. For each triplet (type, demand level, size) ten instances have been proposed therefore giving a grand total of 160 instances.

We first tried to solve the SRAP problem on these instances by means of CPLEX 8.0. Imposing a time limit of $3 \cdot 10^5$ seconds we were able to prove that 42 instances are unfeasible as well as to find a proven optimal solution for the remaining 118 (note that Goldschmidt, Laugier and Olinick [9], using CPLEX 6.5 on a 300 Mhtz workstation left open the feasibility status of some instances). For problem IDP any instance is feasible (we could always assign each demand to a different ring).

We obtained the second set AD by randomly modifying the 42 instances of GLO that are unfeasible for problem SRAP. For each instance we first ran the SRAP version of BTS for 1000 seconds, thus obtaining an unfeasible solution s, then we randomly eliminated one traffic demand at a time until the total traffic was reduced by a quantity greater or equal to the traffic exceeding value B in the ring of s with maximum load. The resulting instance was inserted in set AD only if it passed the following "hardness" test. We applied a simple Multistart Local Search algorithm (MLS) consisting of randomly generating 10^5 starting solutions and optimizing each of them through a Local Search method based on neighborhood N_2. We consider an instance "hard" if the best solution found by MLS was worse than that found by CPLEX within a time limit of 1000 seconds.

The above procedure was repeated until 230 new hard and feasible instances had been generated.

Finally, the last set of benchmark instances $LSHK$ we considered was the one proposed in [14], made up of 40 instances with $|V| \in \{15, 20, 25\}$, $|E| \in \{30, 35\}$, ring capacity $B = 48$ T1 lines and demands in $[1, 30]$. (For more details on the structure of these instances see [14]).

Note that both set AD and $LSHK$ are very difficult to solve to the optimum with CPLEX: a 24 hour run for each instance was sufficient to solve only 12 out of 270 of the SRAP instances. The IDP instances were all solved for set $LSHK$. So in our experiments we compare the results of the heuristic algorithms either with the optimal solution, if available, or with the best solution provided by all methods (our heuristics and CPLEX).

6.2 Comparing the Strategies

We now describe the results obtained for SRAP and IDP on the above three benchmark sets, by algorithms Basic Tabu Search (BTS, see Section 5), Path Relinking (two implementations: $PR1$ and $PR2$, Section 5.1), eXploring Tabu

Table 3: Parameters used in the experiments

prob.	$PR1$	$PR2$	XTS	SS	DMN
SRAP	$\gamma = 75$	$\gamma = 100$	$\#Second = 10$	$RS = 20$	$\Delta_{DMN} = 100$
		$\Delta_{ES} = 15$	$\overline{imp} = 20$	$LS_{iter} = 20$	
IDP	$\gamma = 75$	$\gamma = 50$	$\#Second = 5$	$RS = 20$	$\Delta_{DMN} = 100$
		$\Delta_{ES} = 10$	$\overline{imp} = 20$	$LS_{iter} = 20$	

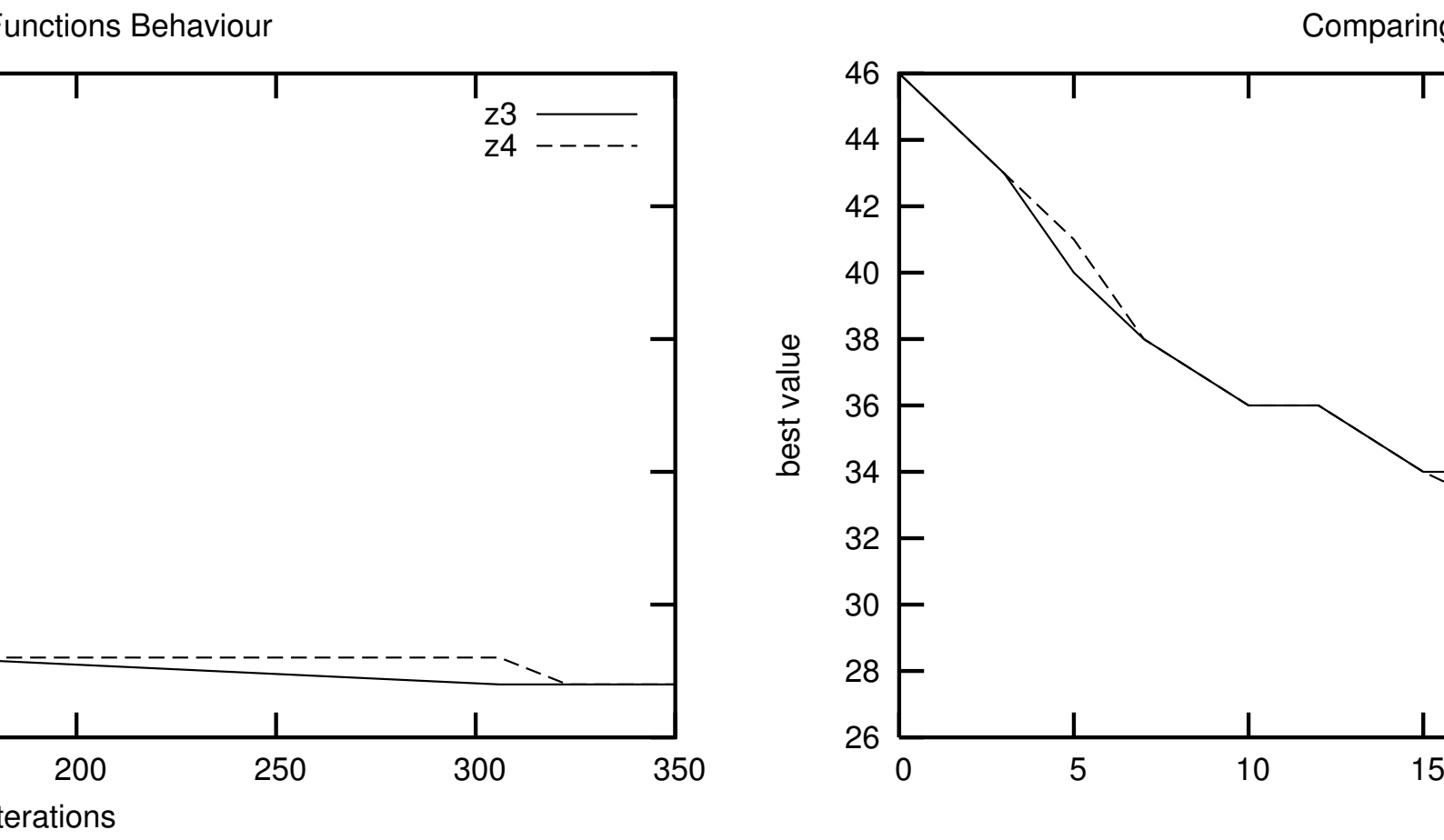
Search (XTS, Section 5.2), Scatter Search (SS, Section 5.3), and Diversification by Multiple Neighborhoods (DMN, Section 5.4). For each algorithm we consider the four objective functions of Section 4.5, but for SS we use the two functions of Section 5.3.

A set of preliminary experiments were done to tune the parameters of the algorithms, giving the values of Table 3. These values were then used for the complete set of computational tests.

We gave a time limit of 5 seconds to each run of an algorithm, but we obviously terminate if the current best solution found by an algorithm is equal to the simple lower bound (4). Furthermore BTS is halted when all the moves in the current neighborhood are tabu. Just observe that the 10^5 iterations of the Multistart Local Search method used to prove the hardness of an instance of set AD require one minute to solve an instance with 15 nodes and one hour to solve an instance with 50 nodes.

Before discussing in detail the results of our experiments, we want to remark the main average differences observed when solving SRAP and IDP through local search methods. In Figure 3 we plot the value of the current best solution obtained by algorithm DMN with function z^4 during a typical run. In particular we used a graph with 25 nodes, taken from set GLO, that have been solved at the optimum and we report the number of iterations performed, on the x axis, and the number of rings in the best solution, on the y axis. The behaviour of these two figures similarly applies to the other instances. We note that with SRAP the algorithm iterates about 10 times more than with IDP. But most important is the fact that with SRAP the objective function value rapidly decreases down to few units, then many iterations are necessary to reduce the value of one or two units, so reaching the minimum. With IDP instead we have a continuous (almost "linear") decreasing of the number of rings until the minimum is reached. Furthermore looking at the size of the solutions we see that IDP has one order of magnitude more rings than SRAP. These differences in the evolution of the algorithm will lead to different behaviour of the objective functions we used to drive the search. /4 /3

In Figures 4 and 5 we report the grand total of optimal/best solutions found by the algorithms using the four objective functions (remind that the SRAP sets



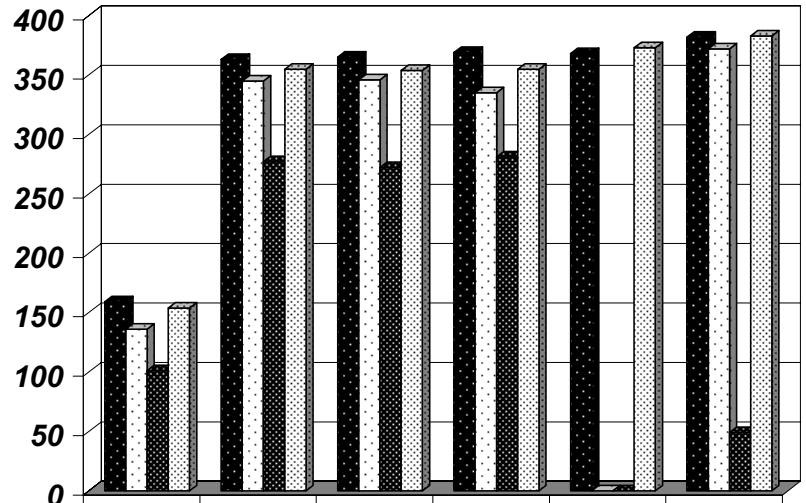
(a) (b)

Figure 3: Typical behaviour when solving SRAP and IDP instances.

contain a grand total of 388 feasible instances, while the IDP sets have 430 instances. Further recall that objective functions z_2 and z_3 cannot be applied with SS, see "Improvement Method" in Section 5.3).

We first try to derive some conclusions on the effect of the objective function when solving SRAP (see Figure 4). We have already observed (see Section 4.5) that z^0, the natural objective function of the mathematical models (counting the number of rings), is very flat for these problems. Indeed, many different solutions, even feasible or unfeasible, may have the same number of rings. Our experiments performed at the very beginning of this study immediately showed that adopting z^0 we have no information that can be used to guide the search. Thus functions $z^1, \ldots, z^4$ have been proposed to add information basing on the solution at hand (z^1 and z^3) or on the last move performed (z^2 and z^4).

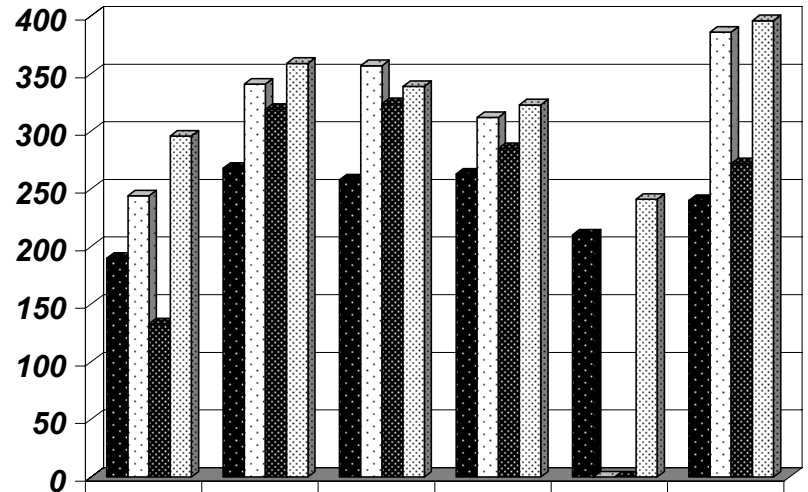
It immediately appears (see Figure 4) that, for problem SRAP, z^3 provides bad results with all methods, and, in particular, very bad results when used



	BTS	PR1	PR2	XTS	SS	DMN
z1	159	363	365	369	368	382
z2	136	345	346	335	0	372
z3	101	277	272	281	0	49
z4	154	355	354	355	373	383

Figure 4: Grand total for SRAP

with DMN. For problem IDP, instead, z^3 gives good performances (see Figure 5). A possible justification of this behaviour is that, due to numerical aspects, z^3 may prefer an unfeasible solution to a feasible one having one more ring. (For a numerical example, consider an unfeasible solution s_u with r rings and $BN = B + \delta_u$, and a feasible solution s_f with $r + 1$ rings and $BN = B - \delta_f$. Using z^3 we have $z^3(s_u) = rB + B + \delta_u < (r + 1)B + B - \delta_f = z^3(s_f)$ provided $\delta_u < B - \delta_f$. Using z^1 we have instead $z^1(s_f) = r + 1 < r + \delta_u = z^1(s_u)$ and z^1 prefers the feasible solution to the unfeasible one. Also function z^4 prefers to move to s_f if the leaving solution is unfeasible. If instead the current solution is feasible $z^4(s_f) < z^4(s_u)$ only in certain cases.) We have seen (see Figure 3(a)) that for SRAP it is relatively easy to find a good solution with few rings, but it is then hard to find an optimal one. Hence starting from a good solution it is not advantageous to move immediately to an unfeasible solution with one less ring since it will be very difficult to re-conduct this solution to a feasible one with the same number of rings. For SRAP, instead, it is more convenient to carefully look for feasible solutions and move to a solution with less rings only if it is feasible. For problem IDP the picture changes. Indeed, we have an almost continuous decrease of the number of rings during the evolution of the algorithm (see Figure 3(b)). So, when the aggressive function z^3 selects an unfeasible solution with one less ring, we have a number of chances to be able to convert this solution into a



	BTS	PR1	PR2	XTS	SS	DMN
z1	190	268	258	263	210	240
z2	244	341	357	312	0	386
z3	133	319	324	285	0	272
z4	296	359	339	323	241	396

Figure 5: Grand total for IDP

feasible one with the same number of rings or even less rings.

We can conclude that function z^3 is appropriate when we expect that finding a good solution is almost difficult as finding an optimal one (i.e. the decrease of the objective function is "linear" with the number of iterations). If instead it is easy to find a good solution, but it is then hard to determine the optimal one z^3 is not appropriate at all.

The three functions z^1, z^2 and z^4 provide good results, for SRAP, but functions z^1 and z^4 compete for the best performances. On IDP, instead, the two best functions are z^2 and z^4.

Looking at the algorithms we see that for SRAP the Basic Tabu Search is certainly dominated by all the other methods, so proving that diversification techniques enhance the performances of a metaheuristic method. The performances of the two Path Relinking implementations are quite similar and close to that of the eXploring Tabu Search. The Scatter Search slightly improves upon the results of the previous algorithms, but the Diversification by Multiple Neighborhhods is the most appropriate algorithm for the problem. We can conclude that diversification is a fundamental tool for designing algorithms able to find very good solutions for SRAP. There is no strong difference among the various methods, but DMN, that can be seen as the most drastic diversification tech-

nique we presented, gives the better experimental results.

A slightly different behaviour of the algorithms can be observed for IDP (see Figure 5). First note that BTS has better performances here than when applied to SRAP. However it is confirmed that diversification is an important tool for metaheuristic algorithms, but for IDP it does not provide performance improvements so large as for SRAP. The worst algorithm is SS. This is mainly due to the slow convergence of the method that needs to generate much more solution than for SRAP before the number of rings significantly decreases (see again Figure 3(b)). With the given 5 second time limit the algorithm often terminates before it has been able to construct a set of good quality solutions. We performed some experiments by giving a large time limit to SS and observed a great improve in the performances with results better than those of the Path Relinking an XTS methods. The best overall results (396 out of 430) are still obtained by procedure DMN with function z^4.

Tables 4 and 5 report some detail of our experiments. For each benchmark set, each algorithm and each objective function we give:

(i) in columns labeled 'sec.' the average running time in seconds over all the instances of a set;

(ii) in columns labeled 'opt.' or 'bst.' the number of solutions with value equal to the optimal one (provided by CPLEX), or to the best available solution value (when CPLEX fails to prove the optimality).

Regarding the computing times one can see that almost all the algorithms are very fast, with the only exception of SS that reaches the 5 seconds limit in some SRAP instances and in all IDP instances.

7 Conclusions

We have first described two techniques used in the design of telecommunication networks when a ring-based topology is adopted. Both techniques can be modeled through a graph and correspond, respectively, to a vertex partitioning problem and to an edge partitioning problem, both with capacity constraints.

We have summarized the relevant literature and introduced basic elements for building Local Search algorithms: neighborhoods, data structures to efficiently explore the neighborhoods, tabu lists and objective functions. In particular, we have described a new variable objective function that depends on the transition from one solution to a neighboring one.

We have then discussed how to apply several diversification and intensification techniques, including Path Relinking, eXploring Tabu Search and Scatter Search. We have also proposed a novel diversification method that we call Diversification by Multiple Neighborhoods.

Extensive computational results on existing and newly generated benchmark instances show that the variable objective function in conjunction with the new diversification method produce the best results for both problems.

Acknowledgments

This research was supported by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), Italy and by Consiglio Nazionale delle Ricerche (CNR), Italy.

References

[1] R. Aringhieri, M. Dell'Amico, and L. Grasselli. Solution of the sonet ring assignment problem with capacity constraints. Technical Report 12, DISMI, University of Modena and Reggio Emilia, 2001.

[2] M. Dell'Amico, A. Lodi, and F. Maffioli. Solution of the cumulative assignment problem with a well-structured tabu search method. *Journal of Heuristics*, 5(2):123–143, 1999.

[3] M. Dell'Amico and M. Trubian. Solution of large weighted equicut problems. *European J. Oper. Res.*, 106(2-3):500–521, 1998.

[4] F. Glover. A template for scatter search and path relinking. In J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Lecture Notes in Computer Science*, volume 1363, pages 13–54. 1997.

[5] F. Glover. Scatter search and path relinking. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 297–316. McGraw Hill, 1999.

[6] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.

[7] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.

[8] O. Goldschmidt, D. S. Hochbaum, A. Levin, and E. V. Olinick. The sonet edge-partition problem. *Networks*, (41):13–23, 2003.

[9] O. Goldschmidt, A. Laugier, and E. V. Olinick. SONET/SDH ring assignment with capacity contraints. *Discrete Applied Mathematics*, (129):99–128, 2003.

[10] P. Hansen and N. Mladenović. Variable neighborhood search. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*. Oxford Academic Press, 2001.

[11] M. Laguna. Clustering for the design of sonet rings in interoffice telecommunications. *Management Science*, 40(11):1533–1541, 1994.

[12] M. Laguna. Scatter search. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*. Oxford Academic Press, 2001.

[13] M. Laguna, R. Martí, and V. Campos. Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Computers Oper. Res.*, 26:1217–1230, 1999.

[14] Y. Lee, H. D. Sherali, J. Han, and S. Kim. A branch-and-cut algorithm for solving an intraring synchronous optical network design problem. *Networks*, 35(3):223–232, 2000.

[15] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, 1990.

Table 4: Comparing the algorithms for SRAP

GLO (118 instances)

obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.
z^1	0.10	98	0.17	102	0.24	105	0.29	106	1.79	111	0.09	117
z^2	0.11	98	0.16	102	0.23	105	0.06	83	-	-	0.09	118
z^3	0.40	88	0.50	87	0.70	86	0.63	89	-	-	0.01	6
z^4	0.29	90	0.21	90	0.44	90	0.36	91	1.73	107	0.08	117

AD (230 instances)

obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.
z^1	0.06	35	0.31	225	0.36	224	0.33	227	4.63	220	0.29	226
z^2	0.04	27	0.37	213	0.38	211	0.31	223	-	-	0.28	222
z^3	0.05	5	0.40	165	0.54	166	0.60	174	-	-	0.62	35
z^4	0.07	37	0.34	225	0.42	224	0.34	224	4.12	226	0.33	226

LSHK (40 instances)

obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.
z^1	0.11	26	0.25	36	0.23	36	0.26	36	3.88	37	0.30	39
z^2	0.11	11	0.26	30	0.25	30	0.29	29	-	-	0.29	32
z^3	0.09	8	0.48	25	0.51	20	0.48	18	-	-	0.60	8
z^4	0.12	27	0.40	40	0.37	40	0.39	40	3.54	40	0.30	40

Table 5: Comparing the algorithms for IDP

GLO (160 instances)												
obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.
z^1	0.51	65	0.42	81	0.36	93	0.75	101	4.96	84	0.35	66
z^2	0.78	86	0.63	99	0.71	122	1.01	110	-	-	0.58	131
z^3	0.55	45	0.68	100	0.80	115	1.10	102	-	-	0.53	78
z^4	0.63	98	0.78	120	1.23	114	1.13	110	4.98	100	0.89	137
AD (230 instances)												
obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.	sec.	bst.
z^1	0.65	123	0.31	183	0.44	161	0.78	161	4.99	96	0.37	172
z^2	0.88	149	0.66	202	0.73	195	1.01	190	-	-	0.63	215
z^3	0.59	76	0.65	181	0.70	171	1.12	168	-	-	0.54	175
z^4	0.71	181	0.77	199	0.81	185	1.05	184	5.00	110	0.93	220
LSHK (40 instances)												
obj.	*BTS*		*PR*1		*PR*2		*XTS*		*SS*		*DMN*	
func.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.	sec.	opt.
z^1	0.05	2	0.05	4	0.05	4	0.06	1	5.00	28	0.06	2
z^2	0.42	9	0.13	40	0.13	40	0.14	12	-	-	0.08	40
z^3	0.22	12	0.11	38	0.11	38	0.14	15	-	-	0.07	19
z^4	0.29	17	0.22	40	0.22	40	0.62	29	4.89	25	0.18	39