Distributed Algorithm to design telecommunication Capacitated Survivable Network

Meriem Khelifi
Badji Mokhtar university-algeria,Annaba,Algeria
khelifi.meriem@hotmail.fr

Malika Babes
Badji Mokhtar university-algeria,Annaba,Algeria
malikababes@yahoo.fr

Abstract: In this paper we present the capacitated survivable network design in telecommunication networks. We propose a greedy approach that permits to achieve survivability, based on hierarchical scheme. Our heuristic constructs the routing paths, working capacities by the routing of demands, and then adds the links and the reserve capacities to ensure survivability subject to the minimization of the global construction network cost.

To design the reserve network, we have tried two methods, the local rerouting and the end-to-end rerouting which permit to select the optimized one.

The order of routing and rerouting demands influences considerably the result design. So we have proposed to deal with all possible combinations, throughout parallel and distributed systems.

Keywords: Survivable telecommunication networks, capacitated network design, heuristics, routing and rerouting, link failures.

1. Introduction

Telecommunication networks are subject to link and equipment failures. Since failures cannot be entirely avoided, networks have to be designed so as to survive failure situations.

Network design problems become significantly more difficult when the networks have to be designed to survive failures. As a simultaneous failure of multiple network elements occurs very infrequently, here we focus on single-edge failures. A network is said to be survivable if sufficient capacity exists on the edges of the network, so that disrupted flow can be rerouted if an edge failure. To do so, requires installing spare capacity to the network. This implies huge capital investment for network operators. Therefore designing capacity-efficient survivable networks (i.e., networks with low capacity installation cost) is a critical problem.

When a failure occurs, the perturbed traffic is restored by rerouting techniques. Two different rerouting methods may be used: local or end-to-end rerouting. The choice of the rerouting methods considerably affects the nature of the resulting network. With the local rerouting, each failure induces a new local demand between the end points of the failed link, with a value equal to the total flow initially routed on this failed link. However, in the end-to-end rerouting method, each demand affected by failure is rerouted from its proper origin to its destination. Obviously, end-to-end approach is much more efficient capacity wise and thus generally preferred [10]. The current practice to dimension survivable networks is to design two networks: a base network used in nominal operating state with no failures and a reserve network dedicated to reroute traffic in failure cases [1, 3, 5, 8, 14, 15].

Various heuristics and exact approaches have been developed for designing survivable networks. However, the heuristic approaches are more likely to be trapped in local optima, whilst the exact approaches are only to small or medium size problems. Due to the weaknesses and strengths of the two approaches, and the increasing popularity of meta-heuristic approaches, recently, we have witnessed many meta-heuristics being applied to the survivable network optimization problems [3, 4, 13, 16, 18, 20].

Notice that in telecommunication networks, the capacities are modulated. A capacity module is called capacity facility. We have the possibility to install a single type of facility with constant capacity [5] or different kinds of facility with varying
capacities. The installation of two kinds of facility has proved its influence on the coast of installation’s network [14]. The capacity installed on a link between two nodes allows running of flow up to the capacity in both directions. Thus capacity is undirected even though flow is directed.

In this paper, we are interesting in designing low coast survivable networks. Given point-to-point traffic demands and a coast-capacity function for each link, we aim at finding the topology and the minimum coast capacities satisfying the given demands and the survivability requirements. This problem is known to be NP-hard. In fact, it includes as special cases a number of well-known NP-hard combinatorial optimization problems such as the capacity allocation and traffic management [2,12].

With the above concern in minds, recently we have proposed a Constructive Heuristics in Meta conference [9], which we will tack it as a point of start. CH-CSND based on an hierarchical scheme, contains two sub heuristics. The first one is the Working Capacity Search WCS, determines a minimal set of links with their capacities to routing demands in nominal state, and a sub set that contains links to be added to the previous topology to ensure survivability. The second heuristic is the Reserve Capacity Search RCS, takes the working links found in WCS and applied a re-routing mechanism in order to find the adequate links to secure the topology. The re-routing processes applied in the WCS and the RCS differs entirely, which results two distinct sub sets of links, so we select the optimized one.

The rest of the paper is organized as follows, section 2 presents the formal description and formalization of the Capacitated Survivable Network Design (CSND) with modular capacities. In section 3 we present and discuss our CH-CSND proposition, section 4 explains our advanced operations that we aim to introduce in the CH-CSND. The last section presents our conclusion and some future perspectives.

2. Problem description

In this section we describe the Capacitated Survivable Network Design Problem (CSNDP).We recall that the objective is to determine where and how much necessary capacity to install in the network while minimizing the overall coast. The installed capacities have to be enough to route all traffic demands in case without failures (nominal state) and to reroute the perturbed traffic during a failure occurs (failure state).

The input of this problem is an undirected graph G(V,E), where V is the set of nodes and E is the set of potential links. Matrix D represents the set of traffic demands on links, where D_{ij} gives the size of traffic demands from a source node i to destination node j. Before presenting the objective function with the constraints to satisfy, lets us begin with some notation. We denote:

\[ f_i \quad \text{the fixed coast for establishing a link in the topology graph} \]
\[ l_i \quad (0/1) \text{ decision variable indicating whether the link} \]
\[ w_{ij} \quad \text{capacities allocated for working demands from node} \]
\[ r_{ij} \quad \text{sparse capacities allocated to the link} \]
\[ c_{ij} \quad \text{the cost per unit of capacity added to a link} \]

The objective function of the CSNDP is to minimize:

\[ \text{Min} \sum_{(i,j) \in E} \left[ f x l_{ij} + c_{ij} (w_{ij} + R_{ij}) \right] \quad (1) \]

In our case, capacities are modulated. So, the capacities are expressed by number of modules installed on each link.

In order to repair the number and kinds of facilities installed on each link, we introduce a M(m×x) named Module facility matrix. M [ij,t]=t indicates the number of modules t\times installed on the link (ij).

The coast of installation facilities differs from a link to another. So, we introduce a CM(m×x) called Coast Module matrix, CM[ij,t]=t\times indicates the installation coast of module capacity t\times on link (ij).

The objective function is written as follows:

\[ \text{Min} \sum_{(i,j) \in E} \left[ f x l_{ij} + \sum_{t=1}^{x} CM[ij,t] x M[ij,t] \right] \quad (2) \]

Subject to the following constraints:

- **Working flow balance constrains:** for each demand pair, the total source flow equals the demand, and the total sink flow also equals the demand.No net sourcing or sinking of flows for the given source-destination (S-D) pair occurs at any other node.

- **Spare capacity constrains:** for each demand pair, the routed spare flow equals to the traffic flow; the spare flow for a source-destination pair does not pass through the working path of the pair. The total sink flow also equals to the spare demand. No net sourcing or sinking of flows for a given spare source-destination pair occurs at any other node.
3. Constructive Heuristic for the CSNDP [9]

CH-CSND is a greedy heuristic containing two heuristics working together to construct a feasible solution. The first heuristic is the Working Capacity Search WCS, routes the traffic demands and determines the necessary links and capacities to install for the nominal state without failures. WCS is a simple constructive heuristic, it starts with an empty graph, and then adds links one by one according to the specific roles. Before describing it, we shall define some notations:

- $\pi$: the set of the node pairs with traffic demands (i.e. those non-zero elements in the traffic demand matrix), the demands are ordered in increasing order. Where $\pi_{xy}$, $1 \leq y \leq n$ represents the source-destination node pair, and $n$ is the number of node pairs with traffic demands.

- $T$: a set of cardinality $x$ contains all kinds of facilities that we can install on the network, the facilities are ordered in an increasing order according to their capacities.

- $W$: the set of the node pairs with traffic demands (i.e. those non-zero elements in the traffic demand matrix), the demands are ordered in increasing order. Where $W_{xy}$, $1 \leq y \leq n$ represents the source-destination node pair, and $n$ is the number of node pairs with traffic demands.

We used the same notation in the constructive heuristic as in [8]. Our algorithm routes the traffic demands listed on the matrix $D$ one by one in order $\pi_{xy}$ of the node pairs. Assume that for traffic demands $\pi_1 < \pi_2 < \ldots < \pi_n$ (1 $\leq y \leq n$), a set of working paths $\{WP_{\pi_1}, WP_{\pi_2}, \ldots, WP_{\pi_n}\}$ have been established. For each demand $\pi_{xy}$ in the routing process, we calculate the cost of all paths existing in $WP_{\pi_{xy}}$. Supposing that the $\pi_{xy}$-th node pair is $(s, d)$, and its working traffic demands is $g$, we search the adequate module to install. Let $t^g$, the nearest module on the capacity to the demand $g$, and we verify:

- The module $t^g$ has enough capacities to route the demand $g$ plus the last demand that has been previously routed on this link. In this case, we delete the previous installed module and we replace it by $t^g$.

- Otherwise, we search another module $t^{g+n}$ (n>0), that verifies:
  - The last demand on this link + the capacity of the module $t^{g+n}$ is less than the capacity of the module $t^g$.
  - The cost of the last module installed on this link + the installation cost of the module $t^{g+n}$ is less than the cost of the module $t^g$.

Consequently, we need to repair the last module installed and the last demand routed on each link.

We introduce a matrix named $Last(m \times 2)$. $Last[ij,module]$, $last[ij,demand]$ indicate the last module installed and the last demand on the link $(i,j)$ respectively.

After the choice of facility to install, we search from $WP_{\pi_y}$ the shortest path for the demand $\pi_{xy}$. We add the links of the chosen path to our topology and we select the second shortest path for the same demand. The second path is considered as an end to end rerouting. We add the links of the second path to the set $RP$. We repeat the process for all demands, and finally we delete links from $RP$ which exists in the topology of the network.

In WCS, we have fixed the nominal working paths and the nominal capacities. We have applied a rerouting process. The found topology is not closed, but we have an approximate prevision on the links that may exist to insure the survivability of the network.

For each demand in order $y$ do:

**Step 1:** Choose the appropriate module to install $t^g$.

**Step 2:** Calculate the cost of all paths in $WP_{\pi_y}$, if we install the module $t^g$, select the shortest one, set as $WP_{\pi_y}$.

**Step 3:** Select the second shortest path, set as $RP_{\pi_y}$.

**Step 4:** $\forall (i,j) \in WP_{\pi_y}$ do:

- **Begin**
  
  **If** $(U_{ij} = g)$ **then**
  
  Update the matrixes $U, W$ and $Last$. $(Last (ij, demand)=g$, we don’t install any module)

  **Else**

  **If** $(Last (ij, demand) + g) < t^g$ **then**

  **Begin**

  Install the module $t^g$, 
  Update the matrixes $Last$, $W$ and $M$.

  **End**

  **Else**

  **If** $(\exists t^{g+n}((Last (ij, demand) + g) t^{g+n})$ **then**

  **Begin**

  Install the module $t^{g+n}$, 
  Update the matrixes $M$, $W$, $U$ and $Last$.

  **End**

- **Else**

  **Begin**

  Install the module $t^g$, 
  Update matrixes $M$, $W$, $U$ and $Last$.

  **End**
Step 5: \( \forall l_{ij} \in RP_y \), if \( l_{ij} = 1 \) delete it from RP links.

The second heuristic is the RCS. Before describing it, we define some notations as follows:

\[ \mathcal{R}(m \times m) \]

matrix named Reserve capacity similar to Working capacity. Set \( \mathcal{G} \) the number of links obtained in the WCS heuristic, we put these links in decreasing order according to their working capacity. \( \rho_k \) (\( 1 < y < B \)) is given as a pair of (source, destination) nodes.

For each link \( \rho_k \) do:

Choose the appropriate module, enough to accommodate the total demand on \( \rho_k \), set as \( g' \).

In \( G(V, E \setminus \rho_k) \) select the shortest path for rerouting the link \( \rho_k \), the found path will be set as \( RL_{\rho_k} \).

For each link \( l_{ij} \in RL_{\rho_k} \) do:

If the link is not in the topology which is obtained in the first heuristic, add the link \( l_{ij} \) to the set \( RL \).

We repeat the process for all links. At the end, we obtain a set of links \( RL \) to be added to the topology to ensure survivability. This set is obtained with a local rerouting for each link failure. Remember that at the end of the WCS we obtain a set of links \( RP \) formed by an end-to-end rerouting.

At this stage, we have an initial topology, a working capacity installed on links which satisfy the demands of nominal state, and two sets of links. If we add one of them to the topology, the graph will become bi-connected.

The question is: which set we shall choose to ensure survivability?

In order to minimize the total cost, we propose to deal with each set separately. We suppose the addition of the \( RP \) links to the network. We calculate the reserve capacity to install on the network and the total cost engendered. Doing the same for \( RL \) links and calculates the total cost. Automatically, we choose to install the links that minimize the total cost of the network design.

**Step 1:** Choose the appropriate module \( t_y \).

**Step 2:** Select the shortest path to route \( g' \) in the graph \( G(V, E \setminus \rho_k) \), set as \( RL_{\rho_k} \).

**Step 3:** \( \forall (i, j) \in RL_{\rho_k} \)

If \( l_{ij} = 0 \) then \( RP = RP + \{ (i, j) \} \).

**Step 4:**

- Calculate the total network cost if we install \( RP \) links.
- Calculate the total network cost if we install \( RL \) links.

**Step 5:** Add to the network the set of links with the necessary sparse facilities that minimize the total cost.

Our choice of the order of demands to route and to reroute is: one time in an increasing order and on other time in a decreasing order. The justifications of these choices are expressed as the following remarks:

**Routing demands in increasing order:** each link installed on the network has a capacity non-used \( \geq 0 \), when we try to route a new demand \( g \) on these links, first we verify it’s a non-used capacity. If it is inferior to the demand, then we install a new module and we update the non-used capacities of this link, even if the new capacity non-used is less than the last one.

**Rerouting links in decreasing order:** in the reserve capacity search, we order the links found by the working capacity search in decreasing order. To route the demands, we shall use the non-used capacity of the link or install a new module. This order maximizes the best use of the facilities.

Our inspiration is tacked from [8], which is applicable for MTRS problems. The heuristic does not route the whole demand \( g \) at ones on a single route, but fixes a certain parameter value \( g' < g \) and iterate the routing process until all the \( g \) demands are allocated from the S-D nodes.

The heuristic diversified the demands in order to create a mesh topology and increase the possibility of survivability. A feasible solution depends on a set of control parameters. The authors used a genetic algorithm to find the best combination.

[8] Routes the nominal demand and re-route the working capacity on links in the same manner. The rerouting process is a link restoration which is well known as a simpler and faster, and it may lead to relatively high network cost [6].

**Illustrative example**

The input of this example is tacked from the SND library [21]. We will execute our heuristic to evaluate our proposition.

\[ T = \{63, 252\} \]

\[ f = 1000 \]

\[ \pi = [D[E, C], D[A, C], D[A, E], D[A, B]] \]

<table>
<thead>
<tr>
<th>( D )</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>128</td>
<td>50</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( CM )</th>
<th>63</th>
<th>252</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AB</td>
<td>1000</td>
</tr>
<tr>
<td>B</td>
<td>AD</td>
<td>1200</td>
</tr>
<tr>
<td>C</td>
<td>BC</td>
<td>2000</td>
</tr>
<tr>
<td>D</td>
<td>DE</td>
<td>1000</td>
</tr>
<tr>
<td>E</td>
<td>EC</td>
<td>1000</td>
</tr>
</tbody>
</table>

236
For end-to-end and local rerouting

or end-to-end and local rerouting

b) For increasing and decreasing order local rerouting

**Figure 1**: Example of the obtained network topology

If we compare the local rerouting with the end-to-end rerouting, we remark that our heuristic has proved the result obtained in [7]: the end-to-end rerouting is much more efficient. We have presented the local rerouting in the increasing order and in the decreasing one. The results proved that the order of demands has its influence in the construction cost; especially, when the capacities are modulated and the demands are routed on the same route.

The non-used capacity is very large, which is related to the type of the installed module. We used two modules \{63, 252\}. The difference between their capacities is important. If we introduce another module of capacity equal to 128 units, the non-used capacities will decrease.

### 4. Advanced operations

We remind that we have taken the proposition presented in [9] as a start point. Our perspectives are:
- Introducing the routing cost in the objective function;
- Verifying if an arbitrary order of demands can improve the result;
- And how can we decrease the non-used capacity in the installed modules on the network?

#### 4.1. Routing cost

The selection of links that construct the topology of the network is based on the routing of demands on the network. At each routing we try all admissible paths that joint the source to the target, and then we select the minimal one.

The cost of the path depends on the installation cost of their links, the allocation cost of the necessary capacities and the routing cost. So we must introduce the routing cost when we select the routings paths, as soon as in the objective function (2), to obtain a realistic cost of the network topology.

Let $RC_{ij}$ be the routing cost of routing a unit of demands on the link $(i,j)$. We remind that $W_{ij}$, $R_{ij}$ are respectively the number of unities accommodate from the source node $i$ to the target node $j$ in the nominal state and the reserve state.

The objective function becomes:

$$
\min \sum_{(i,j) \in E} \left\{ f \times l_{ij} + \sum_{t=1}^{x} \left[ CM[ij,t^t] \times M[ij,t^t] \right] + (W_{ij} + R_{ij}) \times RC_{ij} \right\}
$$

Our objective function aim to minimize the total construction cost of the network, it includes the...
fixed installation cost of links, the cost of all modules installed on the links and the routing cost of demands.

4.2. Runtime reduction

Our heuristic presented in [9] can be resumed in these steps:

**Step1:** Determine the working capacities and nominal links.

**Step2:** Take the result of step 1, apply an end-to-end rerouting to search the reserve topology with its capacities.

**Step3:** Take the working capacities and the topology found in step 1, apply a local rerouting for each link failure to calculate the reserve topology.

**Step4:** Compare between the total network cost if we implement the rerouting strategy of step2 or step3, select the minimal one.

We remark that the steps 2 and 3 are entirely independent and depend of step 1.

The WCS and the RCS algorithms applied an end-to-end rerouting and a local rerouting respectively, each strategy has its advantages and drawbacks. In our case we are sure that we have selected the best one that minimizes the total construction cost.

The application of two rerouting strategies to construct the reserve topology implies a double runtime, which makes the heuristic applicable for small instances. To avoid this drawback we propose to separate the execution of the WCS and the RCS on two machines. The input of the re-routing processes is the working topology found in the first stage of WCS.

We propose to parallel the execution of these two processes. The server machine determine the working topology, transfers it to the second machine to calculate the local rerouting. In this moment the server calculates the end to end rerouting. Finally, the server machine decides which rerouting processes minimize the total construction cost.

The following scheme demonstrates the parallel execution.

The server machine

1. Calculate the working topology
2. Calculate the end-to-end rerouting
3. Compare the results, select the minimal one.

The second machine

1. Calculate the global rerouting
2. The transfers between the machines
3. Consecutive executions

**Figure 2:** The execution of the WCS and the RCS on two machines

The parallel execution of some parts of the algorithm permits a certain reduction of the runtime, which can be increased, if the communication time between the two machines is important. In this case, we can consider two processes which implements step2 and step3 respectively. We execute the step1, and then we parallel the execution of these processes on the same machine. Only after the implementation of these propositions, we can decide which one decrease the running time.

4.3. The order of routing and rerouting

As mentioned in [8] “the order of the S-D node pairs will affect the obtained network topology, different orders of the S-D node pairs will result in different network topologies...”.

In [9] we have tried a combination between the increasing and the decreasing orders for routing and re-routing. So, we propose to evaluate arbitrary orders of S-D node pairs. At first we construct the working topologies according to the arbitrary orders; we select the minimal one to try again an arbitrary order of links to apply the local rerouting.

Therefore the execution of different orders of S-D node pairs perform the quality of the solution, the running time is its most important drawback. Let |d| be the routing demands, we can obtain until $2^{|d|}$ permutations of the S-D node pairs. It is very hard to evaluate all this permutations to select the best one.

To deal with this problem, we propose to implement the execution of the heuristic on a distributed network. Each machine takes one instance from the permutations generated by the server machine, executes the WCS heuristic, and returns the working topology to the server machine.

The server takes the best topology, and generates all possible permutation orders of their links to execute the local rerouting. Each machine calculates the reserve capacity throughout the order of rerouting received from the server. The best overall network cost will be selected from the set of returned solutions.

The distributed execution works as follows:

**Step1:** Generates the S-D node pairs by the server machine.
- Distributes the S-D node pairs on the network.

**Step2:** Each machine executes the WCS, and returns the result to the server machine.

**Step3:** Server machine selects the best S-D node pair according to the minimal construction cost of the working topology.
It generates all possible permutation orders of nominal links of the minimal construction and distributes the orders on the network.

**Step4:** Each machine executes the RCS throughout the order of links received from the server machine and returns the result to the server machine.

**Step5:** The server machine compares the received reserve topologies, and then selects the minimal one.

**Step6:** The server machine applies the end to end rerouting to the topology according to the best S-D node pairs obtained in step3.

Compares the result of the end to end rerouting network cost with the minimal total construction of step 5 and selects the minimal configuration.

The performance of the distributed heuristic depends of the running time of the transfers between the server machine and the others machines.

### 4.4. Reduction of unused capacity

If we look to step 4 of the WCS algorithm, our main objective is: how can we use the non-used capacities on links, to route the new demands without installing any additional modules?

Let’s g be the routing demand on the link (u, v), we verify the existence of unused capacities \( U_{uv} \) which is sufficient to route g. If it exists, we install any additional module. Otherwise, we must install a new module \( t^e \) enough to accommodate g. So, we update the unused capacity on the link (u, v). Let’s \( U'_{uv}=U_{uv}-g \) if \( (U'_{uv} > U_{uv}) \) then \( U_{uv}=U'_{uv} \).

We iterate this operation at each routing process on these links. After each iteration, we maintain only the biggest capacity unused on the link. This operation engendered a huge amount of non-used capacity, which is clearly appreciable in the column U of table 1.

To avoid this drawback, we have thought how we can maintain an historic status of all unused capacities on each link. We propose to substitute the matrix U by a set \( UC=(U_{uv},...)/\forall (u, v) \in E, U_{uv} \) contains the unused capacities on each module installed on the link (u, v).

With this operation we do not reduce any unused capacities in the network. But we just have precise information about it. Where resides the necessity of this information?

### 4.5. Diversified demands

The necessity of the last operation appears when we consider the diversified demands on the network. Telecommunication networks do not allow this diversification for all demands but for specific kinds of them.

When a demand is diversifiable, we can select more than one path for the routing. We have not imposed any restriction on the maximum percentage of demand that must be routed on a single path as in [17]. We just require that the demands don’t take more than three paths, because our objective is not to create a mesh topology as in [8], but we aim to obtain the maximum reduction of the non-used capacity on each installed module.

Let’s g, and \( WP_g \) be the routing demand and the admissible routing paths respectively. The diversification can be implemented as follows:

**Step1:** Select the link (u,v) from the \( WP_g \) pathsthat contain the biggest amount of non-used capacity, set as \( U_{uv} \).

**Step2:** Choose the path containing the link \( U_{uv} \), set as P. Update g and \( U_{uv}; g=g-U_{uv}, U_{uv} = C \).

We repeat this process until we accommodate the whole demand g.

The diversification of demands on multiple paths, with the priority of the routing on the path that contains the biggest unused capacity permits a best reduction of the non-used capacity. It allows the usability of the installed capacities before installing new additional modules.

### 6. Conclusion

We have dealt with the capacitated survivable network design problem, which is expressed in term of end-to-end rerouting and local rerouting without recuperation of the non-used capacities in nominal state. We have presented an algorithm based on an heuristic approach and solved the problem in two steps.

We have executed our heuristic presented in [9] on a small example, which permits to show the advantages and the drawbacks of the proposition. We have developed our perspectives formulated in [9], and we tried to improve the heuristic throughout the reduction of the runtime with parallel execution of some parts of the proposed algorithm. We treated the arbitrary orders of routing and rerouting with a distributed implementation.

With the modular capacities, we obtained an important amount of non-used capacities calculated in [9], to reduce it we proposed to diversify the demand, which is not allowed for all kinds of demands in telecommunication networks. The implementation of the advanced operations proposed is in progress to perform the results obtained in [9] and to test real instances.
References


[21] [http://sndlib.zib.de](http://sndlib.zib.de).