University of Würzburg
Institute of Computer Science
Research Report Series

# Decomposition of Large IP Networks for Routing Optimization

Jens Milbrandt, Stefan Köhler, Dirk Staehle[1] and Les Berry[2]

Report No. 293 February 2002

[1] Department of Distributed Systems
Institute of Computer Science
University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
*milbrandt@informatik.uni-wuerzburg.de*
*koehler@informatik.uni-wuerzburg.de*
*staehle@informatik.uni-wuerzburg.de*

[2] RMIT
City Campus
GPO Box 2476V
Melbourne 3001 Victoria Australia
*lberry@rmit.edu.au*

Presented at: European COST project 279 as TD(01)22, Leidschendam, The Netherlands, February 7-8, 2002

# Decomposition of Large IP Networks for Routing Optimization

**Jens Milbrandt, Stefan Köhler, Dirk Staehle**
Department of Distributed Systems
Institute of Computer Science
University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
*milbrandt@informatik.uni-wuerzburg.de*
*koehler@informatik.uni-wuerzburg.de*
*staehle@informatik.uni-wuerzburg.de*

**Les Berry**
RMIT
City Campus
GPO Box 2476V
Melbourne 3001 Victoria Australia
*lberry@rmit.edu.au*

**Abstract**

Optimizing the routing is one of the key issues to enable *Quality of Service* in today's IP networks. Therefore, the topic of routing optimization has been recently addressed in a number of scientific papers [1, 3, 4, 5]. But until now, all presented proposals have one major drawback in common, namely, the mechanisms are not scaling well with increasing network size. In order to overcome this fundamental problem we propose to use a *Divide & Conquer* approach as a heuristic to cope with networks that are too large to be handled with conventional optimization methods. In the following a general framework to decompose arbitrary networks is presented, where the network is partitioned in a first step, followed by the conventional optimization of the individual subnetworks and the merging of the partial optimization results to an overall solution.

**Keywords:** Routing Optimization, Decomposition, IP Network Planning and Management

## 1 Introduction

In the recent years, IP networks have become more and more important mainly caused by the enormous growth of the Internet [6, 7, 8]. Like in any other network, traffic flows have to be carried from a source to a destination. In IP networks, these flows are routed on certain paths that are determined by routing protocols like *OSPF* (Open Shortest Path First) or *EIGRP* (Extended Interior Gateway Routing Protocol). For the computation of the paths, these protocols use different metrics that describe the cost of a link. The length of a path is thereby defined as the sum of costs of all links contained by the path. The shortest path between the network nodes $i$ and $j$ is the one with the minimal length and every flow between $i$ and $j$ is routed on exactly this shortest path. As the link costs in an IP network can be set by the network administration – provided that the cost values are in the definition scope of the used routing protocol – one can influence the setup of the shortest paths. Routing optimization, as realized in this paper, tries to determine link costs, so that the flows routed on the resulting shortest paths do not lead to overload situations.
In general, routing optimization aims at balancing the utilization of the links in an IP network. Such a network with $N$ routers is defined by three matrices of size $N \times N$. The first matrix comprises the link capacities between each pair of routers. If no link exists the entry in the capacity matrix is set to zero. The matrix is not restricted to be symmetrical since, for example, *ADSL* (Asynchronous Digital Subscriber Line) links are asymmetric. The routing optimization is performed for peak rate traffic flows. Like the capacities these flows are defined in a second matrix which comprises an entry for each flow between any pair of nodes in the network. Furthermore, a third matrix is introduced which describes the physical delays of the links. These delays restrict the set of routing possibilities between two nodes. Of course, one would not send packets on a path which due to

its physical delay is many times longer than the shortest path. So the objectives of the routing optimization are:

1. Minimize the maximum link utilization.
2. Minimize the average link utilization.
3. Keep the physical delays within a specified range.

*Mixed Integer Programming* (MIP) as a method to optimize the routing within a specific network is already well known from [1, 3, 10]. So far, the usage of MIP on routing optimization was restricted to small or at most medium-sized networks with up to 30 routers, depending on the associated link structure. For network sizes beyond this order of magnitude, the resulting integer program proved to be too complex to be solved in appropriate time. Although the MIP approach has serious limitations concerning the network dimension, we try to overcome these scalability limits in this paper.
The idea behind MIP used for optimizing the routing of a particular network - with specified topology, link capacities and network flows - is to find certain integer values associated with the router interfaces. These interface costs are determined subject to a general objective function that, in this particular case, minimizes the average and the maximum link utilization, respectively, within the considered network. Based on these interface costs, the classical Internet routing protocols like *OSPF*, *EIGRP* and *IS-IS* [11, 12, 13, 14, 15] calculate their shortest paths resulting in a routing scheme that is now optimized with regard to utilization.
As already denoted, scalability problems arise as soon as the network to optimize gets too big. In our paper we face this issue by introducing a *Divide & Conquer* strategy, i.e. we try to split the network into two subnetworks that are optimized individually. The resulting partial solutions of both subnetworks are then to be joined into an overall solution for the original network.
The remainder of this paper is organized as follows. In Section 2 we describe how to retrieve an adequate decomposition scheme using the MIP technique. Section 3 contains the optimization of the subnetworks. Afterwards, two alternative methods for merging the partial solutions to an integral solution are presented. The optimization algorithms are then applied to a number of example networks and the results are shown in Section 4. And finally, we draw a conclusion from our optimization efforts and discuss future work in Section 5.

## 2 Decomposing a Network

Suppose that we are given an arbitrary IP network that is described as a directed graph $G(V, E)$. Let $V$ be the set of vertices, representing the network's routers, and let $E$ be the set of edges, representing the network links. In order to decompose the network we have to divide the set of vertices $V$ into three disjoint subsets $S$, $B$ and $W$ as illustrated in Fig. 1. We now wish to find an efficient algorithm to decompose the network graph into two disjoint connected sub-graphs in a way, that the following desirable objectives are fulfilled:

1. The number of vertices in subset $S$, separating $B$ and $W$, should be as small as possible.
2. The number of links located in or running into subset $S$ should be as small as possible.
3. The number of vertices in subsets $B$ and $W$ should be approximately the same.

Above objectives 1 and 2 ensure, that the "interface" between the set of nodes $B$ and $W$ is as small as possible, thus narrowing the network region where problems can emerge during the decomposition. Objective 3 is desirable, because, as will be described later, we want derive subnetworks from the sets $B$ and $W$. These subnetworks are then optimized individually and should therefore be of approximately the same size.

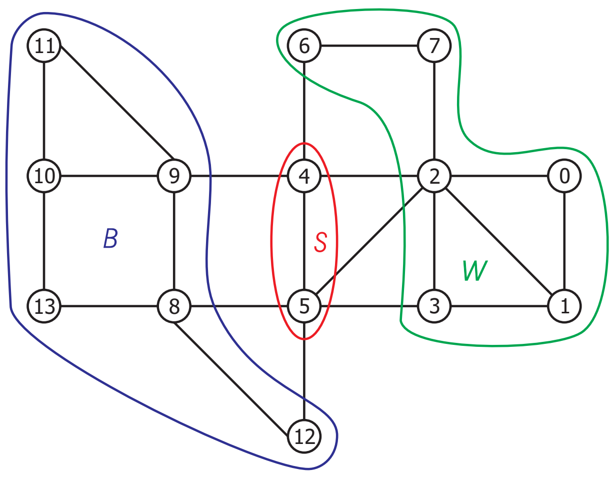


Figure 1: Dividing the graph's vertices into disjoint sets

### 2.1 An Integer Program for Decomposition

Instead of the decomposition algorithm used in [2], which originates from Dulmage and Mendelsohn [9], we use again the technique of MIP to formulate the problem of finding a suitable network decomposition. Therefore, we construct an integer program, that consists of two primary parts. First we have the objective function which acts as an indicator for the quality of a decomposition scheme. The linear constraints constitute the integer program's second part, thus restricting the solution space for feasible decomposition schemes.

#### 2.1.1 Variables

Before we can formulate the objective function and the linear constraints, some variables must be specified for subsequent construction purposes. For each vertex $i \in V$, three binary variables are introduced:

- $s_i = \{0|1\}$, where $s_i = 1$ if $i \in S$, else $s_i = 0$
- $b_i = \{0|1\}$, where $b_i = 1$ if $i \in B$, else $b_i = 0$
- $w_i = \{0|1\}$, where $w_i = 1$ if $i \in W$, else $w_i = 0$

For each vertex $i$ only a single of the above variables can be one at a time, whereas the other two variables have to be zero. Thus, each vertex $i$ is uniquely assigned to one of the sets $S$, $B$ or $W$.

#### 2.1.2 Parameters

In addition to the variables defined in the previous section, we now introduce three parameters that are used for the formulation of the integer program. With these parameters the user gains control over the structure of the desired decomposition scheme. This is important in the cases where there are multiple possible decomposition schemes with identical objective function values. With the help of the parameters, the user is able to propagate his preferences to the integer program formulation. The user-specified parameters are:

- A weight factor $wf$ that reflects the importance of balancing the number of nodes contained by $B$ and $W$ in relation to the size of $S$

- An upper bound $maxdif$ for the difference of the number of nodes in $B$ and $W$ (subject to the cardinality of $V$)
- An upper bound $maxsep$ for the size of $S$ (subject to the cardinality of $V$)

While the usage of parameter $wf$ in the objective function which is defined later, is mandatory, the parameters $maxdif$ and $maxsep$ can be omitted, since they are only contained by optional constraints that are defined later, too.

### 2.1.3 Objective Function

As denoted earlier, the objective function serves to quantify the performance of a feasible decomposition scheme. Thus, each potential solution that is not excluded by the linear constraints, is assessed with a single absolute real value. In this case, the objective function is to be minimized and a simple formulation subject to the sets of vertices $S$, $B$ and $W$ is given as follows:

$$f(S, B, W) = |S| + \sum_{s_i \in S} \ell_i \cdot s_i + wf \cdot (|B| - |W|) \tag{1}$$

Where:

| | |
|---|---|
| $\|X\|$ | is the the cardinality of vertex set $X$ |
| $\ell_i$ | is the number of links adjacent to vertex $s_i$ |
| $wf$ | is a weight factor |

The second addend in Eqn. (1) causes the number of links located in or running into $S$ (cf. Section 2) to be minimized. Note that the objective function above contains the parameter $wf$ as defined in Section 2.1.2. This way, the user imposes more importance to either the size of $S$ or the balanced sizes of $B$ and $W$. In order to get the absolute value of the imbalance between $B$ and $W$ as represented in the last addend of the objective function, we will have to specify some constraints ensuring that the cardinality of $B$ is, without loss of generality, greater than that of $W$.

### 2.1.4 Constraints

The integer program's linear constraints are used to define the set of feasible decomposition schemes. While some of them are mandatory for achieving a disjoint partitioning of the set of vertices $V$ (Eqn. (2), (3) and (4)), some others are included in order to give the user the chance to further influence the structure of the desired decomposition scheme (Eqn. (5) and (6)). The latter constraints are introduced with respect to the case, that there are decomposition schemes showing identical objective function results. Adding the optional constraints to the integer program, the user can specifically search for particular partitioning patterns. In the following, the mandatory and optional linear constraints are formulated in correspondence to the previously defined variables and parameters.

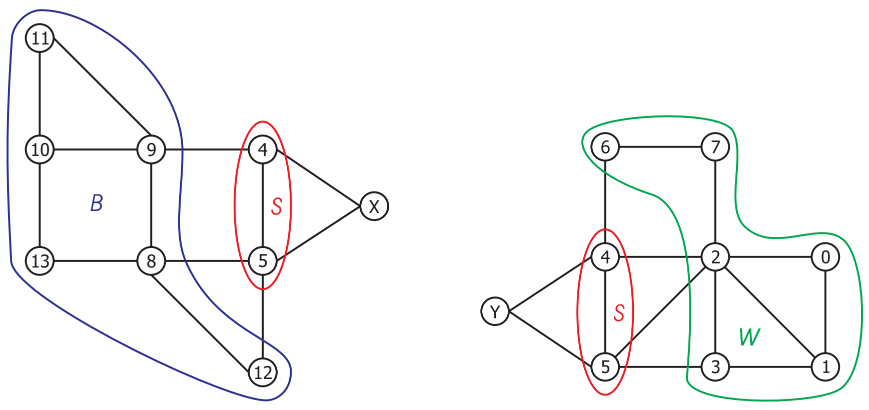
**Mandatory constraints:**

- Each node belongs to exactly one of the sets $S$, $B$ or $W$.

$$s_i + b_i + w_i = 1 \quad \forall i \in V \tag{2}$$

- The cardinality of $B$ is, without loss of generality, greater or equal than the cardinality of $W$. These constraints are necessary only for the correctness of the objective function.

$$\sum_{\forall i \in B} b_i \geq \sum_{\forall i \in W} w_i \tag{3}$$



(a) Sub-network $NX$ (b) Sub-network $NY$

Figure 2: Division of the original network

- There is no edge between a vertex $i \in B$ and a vertex $j \in W$, and vice versa. The vertices are always separated by at least one vertex in $S$.

$$b_i + w_j \leq 1 \quad \forall (i,j) \in E \ \wedge \ i \in B \ \wedge \ j \in W \tag{4}$$

**Optional constraints:**

- The imbalance of sizes of sets $B$ and $W$ is restricted subject to the cardinality of set $V$. The difference must not exceed $maxdif$ percent of the overall number of vertices.

$$\sum_{\forall i \in B} b_i \ - \sum_{\forall i \in W} w_i \ \leq \ maxdif \cdot |V| \tag{5}$$

- The size of set $S$ is limited subject to the cardinality of set $V$. It must not exceed $maxsep$ percent of the overall number of vertices.

$$\sum_{\forall i \in S} s_i \ \leq \ maxsep \cdot |V| \tag{6}$$

## 2.2 Splitting the Network

Based on the decomposition scheme described by the sets of vertices $S$, $B$, and $W$, we now have to divide the original network into two independent subnetworks further called $NX$ and $NY$. Both of them are specified by a share of the original network's resources as well as by set $S$ in association with either set $B$ or $W$. In the following, we consider the proceeding of splitting the original network in the case of subnetwork $NX$:

- Sub-network $NX$ comprises all nodes of $B$ and $S$ (cf. Fig. 2 (a)).
- $W$ is condensed to a single node $X$ that is added to subnetwork $NX$ (cf. Fig. 2 (a)).
- All capacities $c_{ij}$ and flows $f_{ij}$ between nodes $i, j \in S \cup B$ remain unaltered.
- All capacities $c_{ij}$ and flows $f_{ij}$ between nodes $i \in W$ and $j \in B \cup S$ are aggregated.
- All capacities $c_{ij}$ and flows $f_{ij}$ between nodes $i, j \in S$ are bisected.

As an example for the distribution of network resources denoted above, Fig. 3 presents three matrices containing the flows of the original network (Fig. 3 (a)) and the subnetworks $NX$ (Fig. 3 (b)) and $NY$ (Fig. 3 (c)). As described above, the flow between nodes $4$ and $5$, both located in set $S$, is bisected and results to a value of $0.5$ (cf. Fig. 3 (b, c)). Furthermore, all flows from a node in sets $S$ or $B$ to any node in set $W$ are aggregated as could be seen in the last column of the matrix depicted in Fig. 3 (b). The last row of this matrix contains the aggregated flows from all nodes in set $W$ to a node in sets $S$ or $B$.
The procedure of splitting the original network to obtain subnetwork $NY$ is analogous, except that $B$ is condensed to a single node $Y$. The resulting network structures derived from the decomposition scheme as shown in Fig. 1 are presented in Fig. 2.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 3 |
| **1** | 1 | 0 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 2 |
| **2** | 2 | 3 | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 2 | 1 | 2 | 3 | 1 |
| **3** | 1 | 2 | 3 | 0 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 3 | 2 | 1 |
| **4** | 1 | 2 | 1 | 3 | 0 | 2 | 3 | 1 | 2 | 2 | 1 | 1 | 2 | 3 |
| **5** | 3 | 2 | 2 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 |
| **6** | 1 | 3 | 2 | 1 | 2 | 1 | 0 | 1 | 2 | 2 | 3 | 3 | 1 | 1 |
| **7** | 1 | 2 | 1 | 3 | 1 | 1 | 1 | 0 | 2 | 3 | 3 | 3 | 2 | 3 |
| **8** | 2 | 1 | 2 | 3 | 2 | 1 | 2 | 3 | 0 | 1 | 2 | 2 | 3 | 3 |
| **9** | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 0 | 3 | 2 | 1 | 2 |
| **10** | 1 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 2 | 1 | 0 | 2 | 3 | 3 |
| **11** | 3 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 1 | 1 | 2 | 0 | 3 | 2 |
| **12** | 1 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 2 | 0 | 2 |
| **13** | 2 | 2 | 2 | 3 | 2 | 1 | 1 | 2 | 3 | 2 | 1 | 2 | 3 | 0 |

(a) Original flows

| | 4 | 5 | 8 | 9 | 10 | 11 | 12 | 13 | X |
|---|---|---|---|---|---|---|---|---|---|
| **4** | 0 | 1 | 2 | 2 | 1 | 1 | 2 | 3 | 11 |
| **5** | 0,5 | 0 | 1 | 2 | 1 | 1 | 2 | 1 | 12 |
| **8** | 2 | 1 | 0 | 1 | 2 | 2 | 3 | 3 | 13 |
| **9** | 2 | 3 | 3 | 0 | 3 | 2 | 1 | 2 | 10 |
| **10** | 2 | 1 | 2 | 1 | 0 | 2 | 3 | 3 | 13 |
| **11** | 2 | 1 | 1 | 1 | 2 | 0 | 3 | 2 | 15 |
| **12** | 2 | 2 | 1 | 2 | 3 | 2 | 0 | 2 | 13 |
| **13** | 2 | 1 | 3 | 2 | 1 | 2 | 3 | 0 | 12 |
| **X** | 11 | 11 | 11 | 12 | 12 | 15 | 12 | 11 | 0 |

(b) Flows in $NX$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 15 |
| **1** | 1 | 0 | 2 | 1 | 2 | 1 | 2 | 2 | 9 |
| **2** | 2 | 3 | 0 | 1 | 2 | 3 | 1 | 2 | 12 |
| **3** | 1 | 2 | 3 | 0 | 1 | 2 | 2 | 2 | 9 |
| **4** | 1 | 2 | 1 | 3 | 0 | 1 | 3 | 1 | 11 |
| **5** | 3 | 2 | 2 | 2 | 0,5 | 0 | 2 | 1 | 8 |
| **6** | 1 | 3 | 2 | 1 | 2 | 1 | 0 | 1 | 12 |
| **7** | 1 | 2 | 1 | 3 | 1 | 1 | 1 | 0 | 16 |
| **Y** | 10 | 11 | 14 | 15 | 12 | 9 | 11 | 15 | 0 |

(c) Flows in $NY$

Figure 3: Division of the flows

At this point, we are ready to individually optimize the constructed subnetworks. This can be done with an arbitrary optimization method but in the next section, we will rely on the MIP technique.

## 3 Partial Optimization and Integral Solution

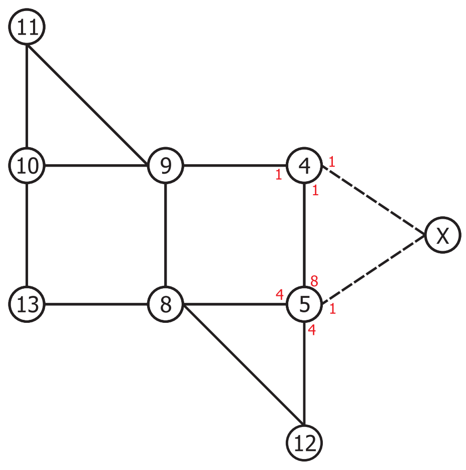
Until now, our concerns applied to the *Divide*-part, whereas in the remainder of this section, we will take care of the *Conquer*-part of our routing optimization problem. After independently conducting the optimization for both subnetworks, we will present two efficient methods for combining the individual results in order to obtain a near-optimal solution for the original network.

### 3.1 Optimizing the Sub-networks

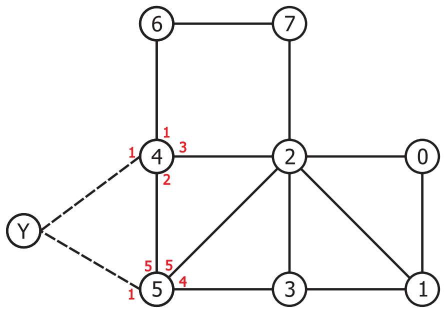
The subnetworks determined in the previous section are optimized based on the MIP technique presented in [1]. While the MIP approach shown there is flow-oriented, a similar path-oriented MIP approach, as described in [3], can be used alternatively. The latter method offers many possibilities for acceleration of the optimization process and will be used in the following. However, since the choice of the optimization algorithm is arbitrary, we can refrain from further details. The basic proceeding for this way of routing optimization contains the following steps:

1. Determine all possible paths within the given network.

2. Based on the set of paths determined in 1, formulate an integer program for the processing of an utilization-optimized routing scheme.



(a) Interface costs in $NX$



(b) Interface costs in $NY$

Figure 4: Differing interface costs

3. Based on the routing pattern obtained in 2, generate an integer program for the calculation of interface costs, such that the utilization-optimized routing scheme is compatible with the shortest path principles.

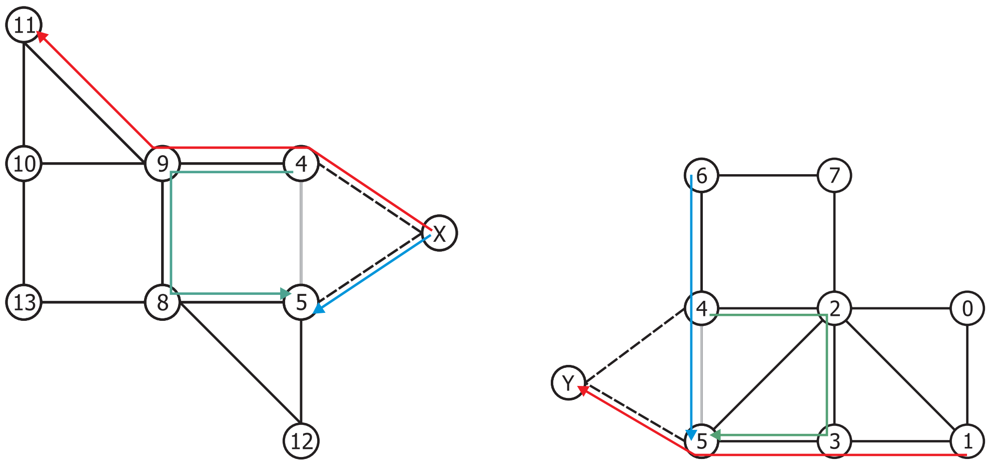
During the optimization of the subnetworks, we encounter a problem that complicates the later process of merging the partial solutions. While determining the set of all paths within a subnetwork, there are lots of paths containing the aggregated node $X$ or $Y$ as interim nodes. This means that traffic between a source and a destination, both located in the same subnetwork, is routed to the other subnetwork and back again. This effect is undesired, because the traffic routed this way burdens the other subnetwork while it is not accounted in the optimization process of this network. Therefore, all paths containing node $X$ respectively $Y$ as a transit node between source and destination are excluded.

### 3.2 Merging the Sub-networks' Optimized Solutions

After the two subnetworks have been separately optimized, we can continue merging both individual solutions to an integral one for the original network. As there are two integer programs to solve during the optimization process of each subnetwork, we get two results of a different kind. Solving the first integer program for both subnetworks delivers two sets of optimized paths, while the result of the second integer program is given with two sets of interface costs. Depending on the result type, we can imagine two different ways of achieving an overall routing solution:

- *Combination of Interface Costs* (**CoIC**): This method works on the two sets of interface costs. It handles the problem of differing interface costs, that were computed twice, once during the optimization of each subnetwork, respectively. The problem is illustrated in Fig. 4. There, the red colored cost values of interfaces around the separating nodes $4$ and $5$ in subnetwork $NX$ differ from those in $NY$. With regard to the interface costs from both subnetworks, we can try to combine these cost values in the original network.

- *Resolution of Incompatible Paths* (**RoIP**): This approach works on the two sets of optimized paths. Trying to combine these sets for the original network leads to the problem of incom-



(a) Paths in $NX$ (b) Paths in $NY$

Figure 5: Incompatible paths

patible paths. Two paths, each belonging to another subnetwork, are said to be incompatible whenever they do not match due to one of the following reasons:

– In most of the cases, path incompatibility exists for source/destination pairs of nodes located in different sets $B$ and $W$. Let therefore be node $i \in B$ and node $j \in W$. Then there are paths from $i$ to the aggregated node $X$ and from the aggregated node $Y$ to $j$, respectively. Both paths contain at least one node in $S$. Therefore, the incompatibility arises whenever the nodes in $S$ differ in both paths. This is the case, represented by the two red colored paths in Fig. 5.

– If nodes within set $S$ are linked, new incompatibilities can emerge, because some paths use this link while others do not. This is illustrated in Fig. 5 by the two blue colored paths.

– If source and destination nodes $i$ and $j$ are in $S$ and there is no direct link between them, the resulting path incompatibility is mandatory, because we do not allow the aggregation nodes $X$ and $Y$ to be interim nodes. As a consequence, routing from $i$ to $j$ and vice versa is only possible via the currently optimized subnetwork $NX$ or $NY$. Removing the grey colored links between nodes $4$ and $5$, this case gets clear when considering the two green colored paths.

After all incompatible paths have been resolved this method requires the computation of interface costs for the original network.

As we will learn from the results in Section 4, each of the alternatives above has its advantages and drawbacks. The main differences lie within the speed and the reliability of getting a solution having a good quality.

### 3.2.1 Combination of Interface Costs

The procedure of combining interface costs is an ad-hoc approach towards a configured routing scheme in the original network. Having the interface cost values of both subnetworks at our disposal, we propose to proceed this way:

- Adopt all interface costs from the subnetworks to the original network, except those belonging to a node in $S$.
- Calculate the average interface cost value from those of both subnetworks, not including those associated with a node in $S$.
- Apply the average cost value to all interfaces "around" all the nodes in $S$.

The above proceeding is illustrated in Fig. 6. There, the black colored interface costs are adopted directly from the subnetworks' cost sets, while the red colored values result from the computation of the average interface cost.

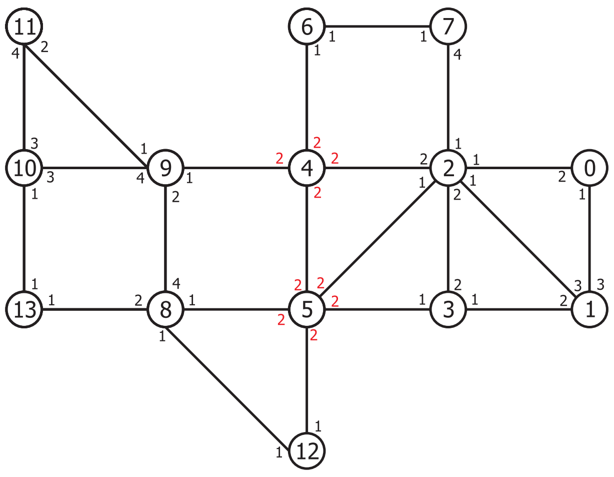


Figure 6: Combined interface costs

### 3.2.2 Resolution of Incompatible Paths

The second approach for merging the subnetworks' optimization results concerns the delivered routing patterns. While joining the two sets of routes, we have to be aware of paths from different sets being incompatible to each other. The term of path incompatibility was already explained in Section 3.2. In the following we present a method to resolve all incompatible paths. Before we start, the source/destination node pairs with incompatible paths should be sorted in descending order concerning their flows. Therefore, the largest flows are accommodated first during the resolution process. Finally, the following algorithm can be applied:

```
algorithm:
1: foreach routers $u$ and $v$ with incompatible paths
2:     generate all paths $p_k^{uv}$ between $u$ and $v$
3:     foreach path $p_k^{uv}$
4:         if $p_k^{uv}$ is conform to $RS$
5:             calculate $u_k^{avg}$ and $u_k^{max}$
6:     if $u_k^{avg}$ = $\min_j\{u_j^{avg}\}$ and $u_k^{max}$ = $\min_j\ \{u_i^{max}\}$
7:         $RS$ = $RS \cup p_k^{uv}$
```

*variables:*

| | |
|---|---|
| $p_k^{uv}$ | k-th path between routers $u$ and $v$ |
| $RS$ | current routing scheme |
| $u_k^{avg}$ | average link utilization caused by path $p_k^{uv}$ |
| $u_k^{max}$ | maximum link utilization caused by path $p_k^{uv}$ |

The above algorithm starts with a routing scheme for the original network containing all compatible paths from both subnetworks. It considers successively each flow between nodes with incompatible paths. For each such source/destination pair the set of all possible paths between them is determined. Then, for each generated path, the current flow is supposed to be routed over it. The average and maximum link utilization resulting thereby are calculated. From all the generated paths, the one producing the lowest average and maximum link utilization is adopted to the current routing scheme of the original network. After this best path has been found, the algorithm continues with the flow next in size.

## 4 Results

In this section the performance of the decomposition approach towards IP routing optimization is examined. First, we analyze the efficiency of the decomposition algorithm that was realized with help of the MIP technique. The efficiency is thereby observed with regard to different parameters such as network size and meshing degree. Thereafter, we consider the performance of IP routing optimization when the decomposition approach is used. In doing so, the optimization results using decomposition are compared to those from optimizing without decomposition.
All example networks generated for performance evaluation purposes were produced with help of the GT-ITM (Georgia Tech Internetwork Topology Models) software, a random graph generator from the Georgia Institute of Technology [16]. The input parameters for the GT-ITM are the number of network nodes and the probability for the existence of an edge between two nodes. The GT-ITM then generates a directed graph that can finally be transformed in the input format required for our decomposition algorithm.
In all simulations, the weight factor $wf$ defined in Section 2.1.2 was set to a constant value of $wf = 2$ leading to well-balanced sets $B$ and $W$ and adequate sizes for $S$. Moreover, the parameters $maxdif$ and $maxsep$ were set to a relatively high value of $maxdif = maxsep = 50\%$ thus merely restricting the decomposition's solution space.

### 4.1 Efficiency of the Decomposition Algorithm

In the following, we will show, that the MIP approach for conducting the decomposition, i.e. the detection of an appropriate network partitioning, performs good even for large networks.

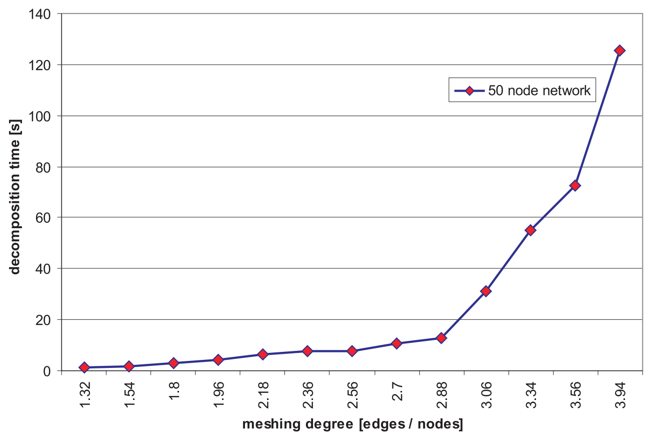


Figure 7: Impact of meshing degree on decomposition time

The speed by which a solution for the decomposition problem can be found depends on several network-specific parameters, as e.g. the meshing degree $md$ that is defined as

$$md = \frac{number\ of\ edges}{number\ of\ nodes} \tag{7}$$

The impact of the meshing degree on the decomposition time is considerable as can be seen in Fig. 7. Up to a meshing degree of about $md = 3$ we recognize a slight increase of the decomposition time. For meshing degrees beyond this value, the decomposition time rises much stronger than before but still constantly. However, a meshing degree in the interval $[1.2,\ 2.1]$ seems to be realistic with regard to current IP backbone networks. After the examination of several such networks, two of them gave the impulse for the above interval. The *Abilene* network [17] is only weakly meshed with a meshing degree of about $md = 1.2$, thus giving the interval's lower bound. In contrast, the *Sprint* network [18] is meshed much stronger with a meshing degree of about $md = 2.1$, that represents the upper bound of the interval.
Of course, the the number of nodes in a network represents another major factor influencing the length of time required for finding an appropriate decomposition scheme. For evaluating the impact of the network size on the decomposition time, the meshing degree of all examined networks has been fixed to an average value of $md = 1.8$. For each network size, random networks are generated on which the decomposition algorithm is applied. The decomposition times depicted in Fig. 8 are mean values based on the measurements of these different networks.
As expected, the times to solve the decomposition problem increase with the number of nodes contained by the current network. However, the slope of the curve is not constant, so the complexity of the decomposition problem rises exponentially with respect to the network size. This is obvious since the slope of the curve in Fig. 8 is growing steadily.

### 4.2 Performance of IP Routing Optimization using Decomposition

In Section 3.2 we introduced two different methods of merging the results from both subnetworks' optimizations. Each of those – the *Combination of Interface Costs* and the *Resolution of Incompatible Paths* – has been tested on several networks that are small enough to be optimized with and without the usage of decomposition. The statements concerning their IP routing optimization capabilities are presented by means of two networks of medium sizes. The first comprises

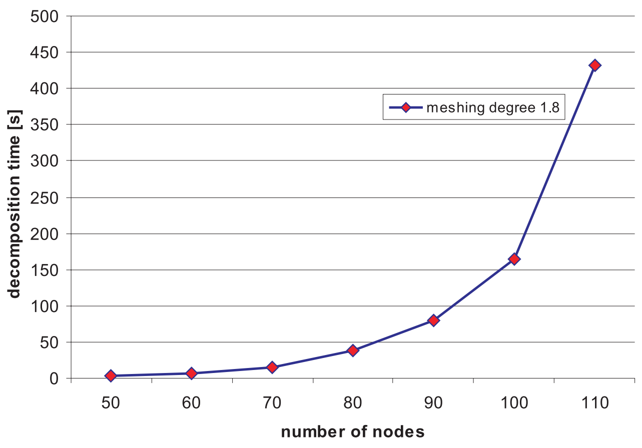


Figure 8: Impact of network size on decomposition time

14 nodes and is already known as the decomposition example network presented in Fig. 1. The second network reproduces the *AT&T IP Backbone* structure (as of mid 2000) [19], including 25 nodes as illustrated in Fig. 9. For both example networks, the end-to-end traffic flows have been randomly generated with regard to the current network capacities.

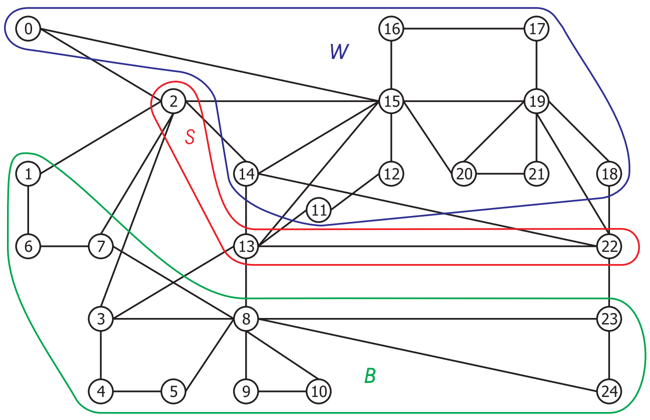


Figure 9: AT&T IP backbone structure

The optimization results are summarized in Tab. 1 for the AT&T network and in Tab. 2 for the 14 node example network. There, the rows of the tables contain measured values of a particular type. Those types are the average and maximum link utilization, the maximum-utilized link and the overall solution time required for conducting the optimization. The tables' columns represent the optimization methods. The first column shows values for the unoptimized network, whereas the next three columns are dedicated to the optimized case. From left to right the columns extend from optimizing without decomposition to optimizing with decomposition using either method RoIP or CoIC. With respect to the tables' contents we can draw several conclusion.
First, the utilization results of IP routing optimization using decomposition are always worse than those of direct optimization. This holds for the average and the maximum link utilization, respectively. Second, using decomposition for optimizing the IP routing delivers results always

| | unoptimized | optimized | **RoIP** | **CoIC** |
|---|---|---|---|---|
| average link utilization | 25.5% | 25.5% | 25.7% | 26.6% |
| maximum link utilization | 80.8% | 49.9% | 64.4% | 91.9% |
| maximum-utilized link | $8 \rightarrow 13$ | $8 \rightarrow 13$ | $15 \rightarrow 2$ | $7 \rightarrow 2$ |
| solution time | <3s | 158.83s | 75.32s | 4.32s |

Table 1: Decomposition results of the AT&T IP backbone network

| | unoptimized | optimized | **RoIP** | **CoIC** |
|---|---|---|---|---|
| average link utilization | 18.9% | 18.7% | 19.2% | 19.3% |
| maximum link utilization | 52.8% | 38.8% | 47.1% | 46.5% |
| maximum-utilized link | $2 \rightarrow 4$ | $8 \rightarrow 5$ | $5 \rightarrow 2$ | $9 \rightarrow 4$ |
| solution time | <1s | 57.56s | 11.22s | 7.22s |

Table 2: Decomposition results of the 14 node network

faster than direct optimization. Third, combining interface costs is always faster than resolving incompatible paths. Fourth, the bottleneck links showing the highest utilization values are always adjacent to nodes within the vertex set $S$. And fifth, the RoIP always leads to an improvement of the maximum link utilization – compared to the unoptimized case – whereas the CoIC leads to maximum link utilizations, that can be better but also even worse than in the unoptimized network. This is exactly what happens for the 25 node network as shown in Tab. 1. However, for the 14 node network the value for the maximum link utilization achieved with CoIC is even lower than the one obtained with RoIP.

## 5 Conclusion

In this paper we presented a method for optimizing the routing in large IP networks. As the direct optimization is limited to at most medium-sized networks, we considered the approach of decomposition that follows a *Divide & Conquer* strategy. Thereby, the network to optimize was divided into appropriate subnetworks which were subsequently optimized. The results from both subnetworks' optimizations were then merged in order to achieve a near-optimal routing solution for the original network. For the decomposition we introduced a general approach relying on the MIP technique. Based on parameters, the integer program formulated for decomposition enables the targeted search for particular decomposition schemes.

The results of this study show that the locating of a proper decomposition scheme depends on two network parameters that are the meshing degree and of course the network size as given by the number of nodes. The decomposition algorithm proved to be efficient even for large networks. However, the IP routing optimization based on decomposition is a heuristic and therefore delivers in most of the cases only a sub-optimal solutions. Its application is useful solely, when the networks get too large to be optimized directly. In these cases, the decomposition approach towards IP routing optimization represents a real trade-off between speed and quality of a routing solution.

The future work regarding the decomposition consists of multiple tasks. First, the variation of the weight factor $wf$ and its impact on the decomposition scheme and solution time should be examined. Furthermore, the restriction of the decomposition solution space with help of the targeted use of parameters $maxdif$ and $maxsep$ is to be analyzed with respect to the efficiency of the decomposition algorithm.

## References

[1] S. Köhler, D. Staehle and U. Kohlhaas, “Towards an optimization of the routing parameters for IP networks”, Technical Report 258, University of Würzburg, Mai 2000

[2] D. Staehle, L. Berry, S. Köhler and P. Tran-Gia, “Fast heuristics for optimial routing in large IP networks”, Technical Report 262, University of Würzburg, Juli 2000

[3] J. Milbrandt, “Possibilities of Routing Optimization in IP Networks”, Diploma Thesis, University of Würzburg, April 2001

[4] K. G. Ramakrishnan and M. A. Rodrigues, “Optimal Routing in Shortest-Path Data Networks”, Bell Labs Technical Journal, January-June 2001

[5] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights”, in *IEEE INFOCOM’00,* (Tel-Aviv, Israel), March 2000

[6] C. Huitema, “Routing in the Internet”, Prentice Hall PTR, 1995

[7] D. Bertsekas and R. Gallager, “Data Networks”, Second Edition, Prentice Hall, 1992

[8] A. S. Tanenbaum, “Computer Networks”, Third Edition, Prentice Hall, 1996

[9] A. Dulmage and N. Mendelsohn, “Coverings of Bipartite Graphs”, Canad. J. Math., Vol. 10, pp. 517-534, 1958

[10] A. Bley, M. Grötschel and R. Wessäly, “Design of Broadband Virtual Private Networks: Model and Heuristics for the B-WiN”, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Preprint SC-98-13, 1998

[11] J. Moy, “Experience with the OSPF protocol (RFC 1246)”, July 1991

[12] J. Moy, “OSPF Version 2 (RFC 1247)”, Proteon Inc., July 1991

[13] J. Moy, “OSPF Version 2 (RFC 2328)”, Ascend Communications Inc., April 1998

[14] Charles L. Hedrick, “An Introduction to IGRP”, The State University of New Jersey, Center for Computers and Information Services, Laboratory for Computer Science Research, August 1991

[15] D. Oran, “OSI IS-IS Intra-domain Routing Protocol (RFC 1142)”, Digital Eqnipment Corp., February 1990

[16] K. Calvert and E. Zegura, “GT Internetwork Topology Models (GT-ITM)”, College of Computing, Georgia Institute of Technology, 1996

[17] “Abilene Network Backbone” as of February 2002,
$http : //www.ucaid.edu/abilene/html/maps.html$

[18] “U.S. Sprint IP Backbone Network and Internet Centers” as of Year-End 2001 Planned,
$http : //www.sprintesolutions.com/resources/maps/bb\_map3\_popup.html$

[19] “AT&T IP Backbone Network” as of 2Q2000,
$http : //www.ipservices.att.com/backbone/images/bbone - map.pdf$