

Optimal Routing with Single Backup Path Protection

Simon Spoorendonk

DIKU, University of Copenhagen, Denmark

24th of November

Based on joint work with: Thomas Stidsen (DTU), Bjørn Petersen (DIKU), Kasper Bonne Rasmussen (DTU), and Martin Zachariasen (DIKU) [4]

1 Path Protection

- Path protection schemes

2 Protection Pricing

3 SBPP

- Mathematical Model
- Example

4 QCDPP

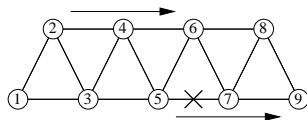
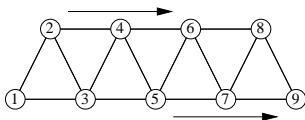
- Mathematical Model
- NP-hard
- Branch-and-Bound Algorithms
- Shortest Path Problem with Resource Constraints

5 Test Results

6 Open Problems

Tele Communication Networks

- Minimize capacity usage on network when meeting customer demand
- Provide reliable service, e.g., in case of cable failure



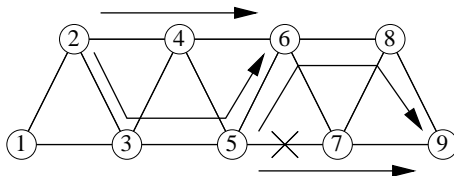
Path Protection

We define path protection in the following way:

- Each circuit contains **one primary connection**. This connection functions in the normal situation
- If the primary connection fails, the signal is routed along **one backup connection**.
- The paths available for the backup connection depends on the path protection scheme

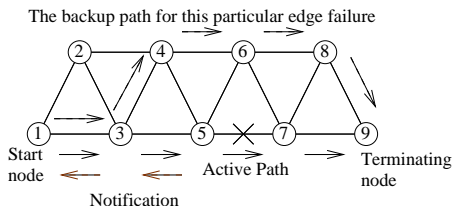
1+1 Path Protection

- Sends packets on both paths
- More than twice the non-failure capacity required



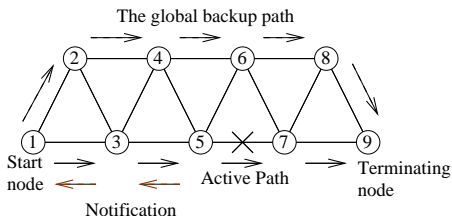
Full Backup Path Protection

- Theoretical most efficient method
- Each possible cable failure is protected by a possible unique backup path
- Lots of capacity sharing between backup paths



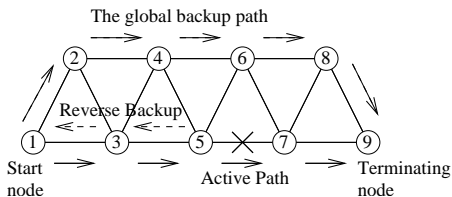
Single Backup Path Protection (SBPP)

- Only sends packets on backup path if a failure occurs
- Allows sharing capacity of backup paths



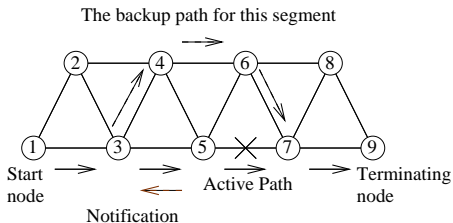
Reverse Single Backup Path Protection (Haskin)

- Variant of SBPP where data is routed back to start node
- Increases capacity requirements but decreases notification time



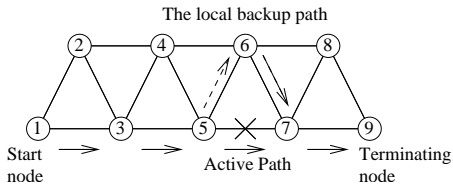
Segment Backup Path Protection

- Segments of primary path cables are protected by same backup path



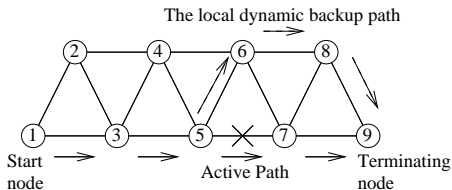
Local Backup Path Protection (LBPP)

- Reroutes the backup path between the failed end nodes of the primary path



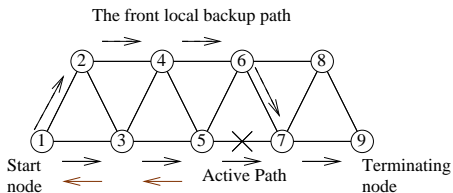
Dynamic Backup Path Protection

- Variant of LBPP where backup paths are routed directly from the start of failed connection to the target node



Front Dynamic Backup Path Protection

- Another variant of LBPP where backup paths are rerouted from start node to end of failed connection



Pricing

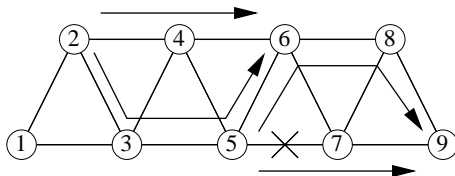
Given we have a network, what should users pay to use it ?

- Obviously the users should pay for the capacity they use
- We will assume the users pay a price linearly dependent on the use of cable capacity.
- The price for a circuit connection is then simply the sum of the capacity costs.
- This leads to an shortest path routing for establishing unprotected circuit connections.

The Problem

First the easy problem, 1+1 protection:

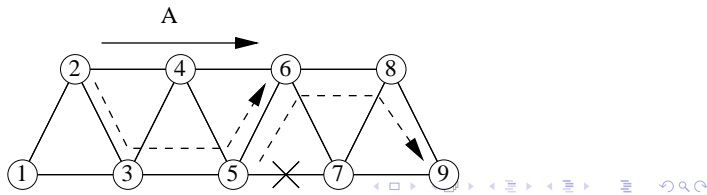
- How much should connection A and connection B pay for the use of the arc a_{56}



The Problem

First the easy problem, 1+1 protection:

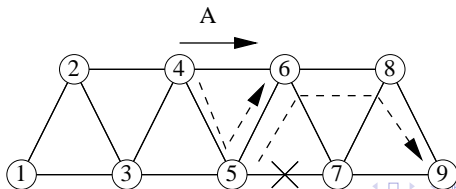
- How much should connection A and connection B pay for the use of the arc a_{56}
- But what about sharing ? How much should connection A and connection B **pay** for the use of the arc a_{56} ?



The Problem

First the easy problem, 1+1 protection:

- How much should connection A and connection B pay for the use of the arc a_{56}
- But what about sharing ? How much should connection A and connection B **pay** for the use of the arc a_{56} ?
- And now, how much should connection A and connection B pay for the use of the arc a_{56} ??



Shared Capacity

In general: How should users pay for shared protection capacity ?

- Wayne Grover: "This leads to a surprisingly difficult problem for exact solution and is currently an open area of research" [1].
- But paying for protection capacity is not a unique telecommunication planning problem, the same issue arise in e.g. electricity production planning ...

Single Backup Path Protection Pros/Cons

We focus on SBPP for the following reasons:

- Simple intuitive and easy to explain because of the close resemblance with 1+1 protection
- Very capacity efficient
- Can be controlled from edge routers
- Seems ideally suited for Multi Path Label Switching [5] implementation used in backbones

There **are** drawbacks with SBPP:

- Not very fast recovery time, because of the notification time.
- Not the most efficient path protection method

The abstract model

We choose to use a very simple model of the path protection problem:

- We assume an oriented network $G(V, A)$ of nodes $i \in V$ and arcs $a \in A$
- We assume a known static demand matrix D^k of the volumes of protected circuit connections from origin node o_k to terminating node t_k
- The (linear) cost for using an arc is c_a
- The objective to be minimized is the sum of costs for the capacity use in each arc (link).

Failure Situations

We model the possible failures with a number of failure situations $s \in S$:

- The set $F_s \subseteq A$ enumerates the arcs which fail in situation s , i.e. the failed arcs are $f \in F_s$
- We model single link failures by creating the failure sets F_s of the two arcs modeling each link
- In principle we can model any failure situation, but only if the network stays connected ...

Single Backup Path Protection

We will formulate the problem directly as a column generation problem where the variables:

- λ_p^k : Correspond to a *path-pair* $p \in P_k$ for demand k .
 - $PRI_{p,a}^k \in \{0, 1\}$: The incidence matrix for the primary path, i.e. 1 when primary path $k \in P_k$ for demand k use arc $a \in A$
 - $BAC_{p,a}^k \in \{0, 1\}$: The incidence matrix for the backup path, i.e. 1 when primary path $k \in P_k$ for demand k use arc $a \in A$
 - The primary path and the backup path needs to be failure disjoint, i.e. $PRI_{p,a}^k + BAC_{p,a'}^k \leq 1 \forall k, p, s : a, a' \in F_s$
- θ_a : Used to record how much capacity is needed

When to backup?

Given the incidence matrixes when should the backup path be used?

- The backup paths should be used when the primary path fails in a failure situation s
- $SWITCH_ON_{p,s}^k$: The incidence matrix for switching on the backup path:

$$SWITCH_ON_{p,s}^k = \left(1 - \prod_{a \in F_s} (1 - PRI_{p,a}^k)\right) \quad \forall k, p, s$$

The Master problem

Min:

$$\sum_{a \in A} c_a \cdot \theta_a$$

s.t.:

$$\sum_{p \in P_k} \lambda_p^k = d_k \quad \forall k$$

$$\sum_{k \in K} \sum_{p \in P_k} PRI_{p,a}^k \cdot \lambda_p^k +$$

$$\sum_{k \in K} \sum_{p \in P_k} SWITCH_ON_{p,s}^k \cdot BAC_{p,a}^k \cdot \lambda_p^k \leq \theta_a \quad \forall a, s : a \notin F_s$$

$$\lambda_p^k, \theta_a \in R_+$$

Exponential paths ...

The number of possible paths grows exponentially with the size of the network (and also if the average node degree increases ...)

- We want to generate the path possibilities on the fly!
- In this way we (in principle) only need a (tiny) subset of the path-pairs, $|K| + \frac{1}{2} \cdot |A| \cdot |A|$
- We will start with dummy path pairs which are guaranteed to be too expensive to be selected in the final solution ...
- ... we simply select **all** links in all failure situations.
- How can we select improving path pairs ?
- **And more fundamentally: What is the price of SBPP path-pairs ?**

The prices

Dual variables are:

- $\alpha_k \geq 0$: The (highest) price we are currently paying for satisfying demand k with a disjoint path pair
- $\beta_a^s \geq 0$: The price the *backup* path has to pay for using arc a in failure situation s

Path pair price

We need to calculate the *reduced cost* of the path pairs:

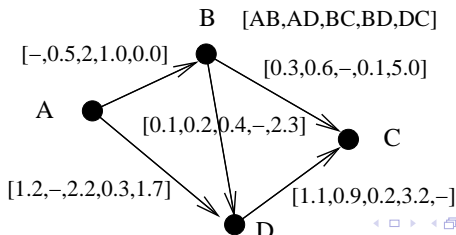
- $C_{reduced} = C_{original} - \sum_i \pi_i a_i$
- In our case it consists of:

$$C_{reduced} = C_{original} - C_{demand} - C_{primary} - C_{backup}$$

- $C_{original} = 0$, there are no direct costs !
- $C_{demand_k} = \alpha_k$ dual variables only participate once
- $C_{primary,k,p \in P_k} = \sum_a PRI_{p,a}^k (\sum_s \beta_a^s)$
- $C_{backup,k} = \sum_a \sum_s SWITCH_ON_{p,s}^k \cdot BAC_{p,a}^k \cdot \beta_a^s$
- Notice we for a given demand k only need to consider the cost of the path-pair since the reward α_k is constant.

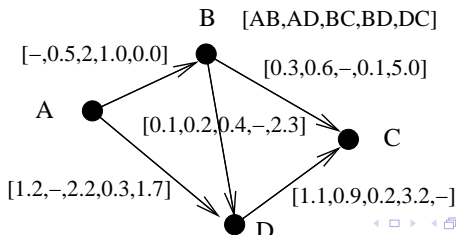
A numerical example

- A very simple example network, with β arrays for each arc (the dash “-”) correspond to the arc it self ...



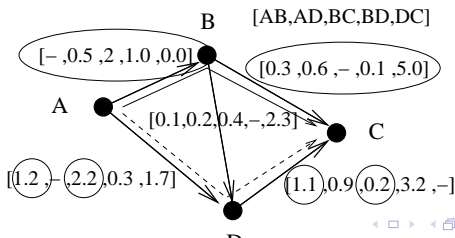
A numerical example

- A very simple example network, with β arrays for each arc (the dash “-”) correspond to the arc it self ...
- There are two possible path pairs:



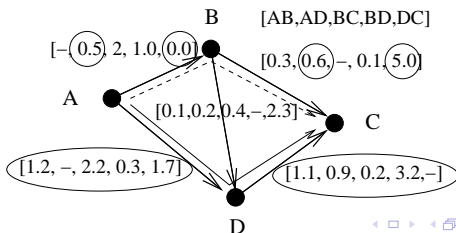
A numerical example

- A very simple example network, with β arrays for each arc (the dash “-”) correspond to the arc it self ...
- There are two possible path pairs:
- $[(AB, BC), (AD, DC)] = 3.5 + 6 + 3.4 + 1.3 = 14.2$



A numerical example

- A very simple example network, with β arrays for each arc (the dash “-”) correspond to the arc it self ...
- There are two possible path pairs:
- $[(AB, BC), (AD, DC)] = 3.5 + 6 + 3.4 + 1.3 = 14.2$
- $[(AD, DC), (AB, BC)] = 5.4 + 5.4 + 0.5 + 1.3 = 12.6$ (The winner)



Quadratic Cost Disjoint Path Problem (QCDPP)

How hard is the QCDPP?

- The problem is NP-complete

How can we solve the QCDPP? Optimal solution methods:

- Formulate MIP and solve with Branch-and-Bound
- Reformulate as an SPPRC and solve with a Label Setting Algorithm

Heuristic approaches:

- Regard reduced networks
- General MIP heuristics on MIP formulation
- Truncated Label Setting Algorithms (relaxed dominans criteria, truncated state space)

The Path-Pair generation problem I

Min:

$$C_{reduced}^k = \sum_{a \in A} \left(\sum_{s \in S} \beta_a^s \right) \cdot x_a + \sum_{a \in A} \sum_{s \in S} \beta_a^s \cdot z_a^s - \alpha_k$$

s.t.:

The Path-Pair generation problem II

s.t.:

$$\sum_{a \in \delta_i} x_a - \sum_{a \in \gamma_i} x_a = \begin{cases} 1 & i = s \\ -1 & i = t \quad \forall i \\ 0 & i \end{cases}$$

$$\sum_{a \in \delta_i} y_a - \sum_{a \in \gamma_i} y_a = \begin{cases} 1 & i = s \\ -1 & i = t \quad \forall i \\ 0 & i \end{cases}$$

The Path-Pair generation problem II

s.t.:

$$|F_a| \cdot u_s \geq \sum_{a \in F_a} x_a \quad \forall s$$

$$|F_a| \cdot v_s \geq \sum_{a \in F_a} y_a \quad \forall s$$

$$u_s + v_s \leq 1 \quad \forall s$$

$$z_a^s \geq u_s + y_a - 1 \quad \forall s, a$$

$$x_a, y_a, u_s, v_s \in \{0, 1\}, \quad z_{ij,qr} \in [0, 1]$$

QCDPP NP-completeness

The decision version of the QCDPP problem states the following question: Does there exist a pair of simple arc disjoint paths (p^{pri}, p^{bac}) from s to t in G such that

$$\sum_{a \in p^{pri}} \sum_{f \in A} \beta_a^f + \sum_{a \in p^{bac}} \sum_{f \in p^{pri}} \beta_a^f \leq C ?$$

QCDPP NP-completeness

To prove NP-completeness (in the strong sense) we have to:

- Prove that we can check a solution in polynomial time (easy)
- Select an existing NP-complete problem to reduce from: 3SAT
- Describe a reduction from the 3SAT [2] problem to the QCDPP problem.

Polynomial checking of QCDPP solutions

Given a solution:

$$(p^{pri}, p^{bac}) = ([o, x_1, x_2, \dots, x_n, t], [o, y_1, y_2, \dots, y_n, t])$$

- Check that primary path and backup path are failure disjoint, i.e. link disjoint.
- Calculate the summed cost of the primary path and the backup path.

3SAT and QCDPP

Given a pair (U, C) where $U = (x_1, x_2, \dots, u_n)$ is a finite set of n variables and $C = (c_1, c_2, \dots, c_m)$ is a finite set of m clauses like: $(x_1 \vee x_2 \vee \overline{x_3})$. The basic question: Is there a setting of the variables U such that all clauses are satisfied, i.e. $c_1 \wedge c_2 \wedge \dots \wedge c_m = \text{TRUE}$

Simpler QCDPP

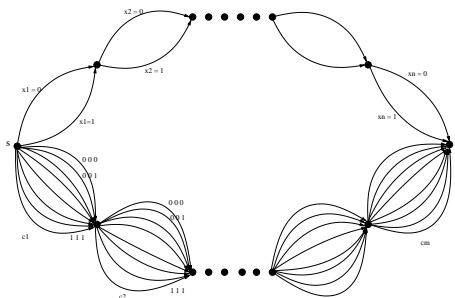
To make the proof simpler we make a few alterations which simplifies the proof:

- Each arc corresponds to a failure situation (easy to change).
- We will construct a multigraph (again, easy to change).
- We only consider 0 or 1 values for the β_a^s dual values.

NP-hard

NP-complete

Given a 3SAT problem construct the following figure:

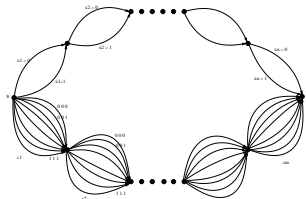


Set the values for β_a^s for all pairs of arcs to zero, except:

NP-complete

Set the values for β_a^s for all pairs of arcs.

- For the bottom (clause) arcs 000, set β_a^s to 1 for the arc a to one randomly chosen variable s
- For the upper (variable) arcs, set $\beta_a^s = 1$ for all pairs of arcs a and s such that a clause containing that variable fails



NP-complete

Given this setup we claim:

- Given a network corresponding to a 3SAT problem solving the QCDPP problem leads to a 3SAT solution, i.e. cost 0
- Given a 3SAT solution and the network we can (in polynomial time) transform the 3SAT solution to a solution of the QCDPP problem by choosing the corresponding arcs.

Brute-force MIP solution

What if we simply solve the subproblem using standard MIP solvers ?

- Feasible approach for small to medium problems only ...
- Possible because problem is quite constrained and we are using sparse networks

Shortest Path Problem with Resource Constraints (SPPRC)

Problem definition:

- A directed graph with edge weights (possible negative)
- Resource constraints, e.g. capacity, time
- Find the least cost path between a pair of nodes.

Very common subproblem in decomposition algorithms

Algorithm

Pseudo-polynomial running time when the number of resources are constant.

Label Setting Algorithm:

- Dynamic programming approach
- Extend partial paths (labels)
- Only maintain pareto-optimal paths through dominance
- Implicit enumeration algorithm

Reformulating the QCDPP into a SPPRC

- Duplicate the graph into a primary and a backup part
- In the backup part all arcs are reversed
- Connect the end nodes of a pair with each other, i.e. this is the link between the two parts of the new graph
- A path from the start node to the copy of the start node corresponds to a primary path and a reverse backup path

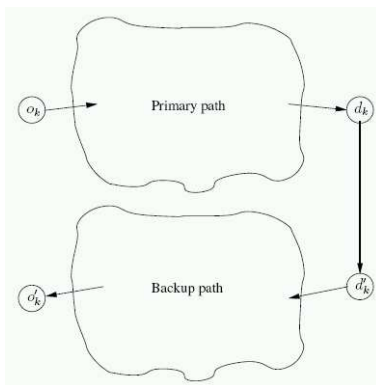
Arc costs

- Arc $a \in A$ on primary path: $c_a = \sum_{s \in S} \beta_a^s$
- Arc $a' \in A'$ on backup path depends on label L :
 $c(a', L) = \sum_{s \in S: s(L)=1} \beta_a^s$
- Arc from end to copy of end node: $c(d_k, d'_k) = -\alpha_k$
- The path cost is:

$$C_{reduced}^k = \sum_{a \in A(p)} \sum_{s \in S} \beta_a^s + \sum_{a' \in A(p)} \sum_{s \in S: s(p)=1} \beta_{a'}^s - \alpha_k$$

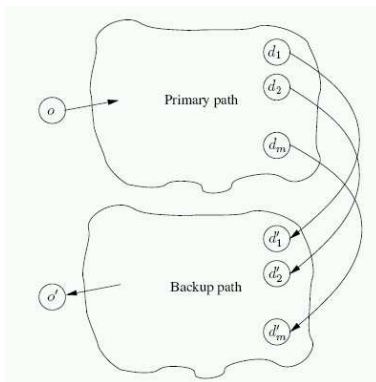
Single pair disjoint paths

We copy the original network to two different parts:



Single to many disjoint paths

Add several links between all terminating node to their copies for a single start node:



Observations

- All arc costs are positive: $\beta_a^f \geq 0$
- If primary path cost more than α_k no need to extend path
- However, hard to dominate paths before switching to backup path
- Branch-bound-Bound is competitive if dominance criteria is weak

Test Networks

- 5 Networks the SNDlib [6]
- RROB: Relative Restoration Over Build, i.e. the network capacity necessary for protection divided by the network capacity necessary for the un-protected case.

	Nodes	Edges	Avg. Node Degree	Number of Demands
newyork	16	49	6.12	240
ta1	24	51 (55)	4.58	396
france	25	45	3.6	300
norway	27	51	3.78	702
cost266	37	57	3.08	1332

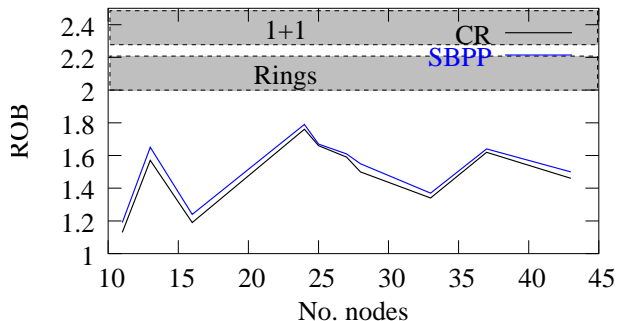
Table: The tested networks

SBPP efficiency

	NF	CR		SBPP	
	Capacity	Abs.	RROB	Abs.	RROB
newyork	412	488.23	1.19	509.9	1.24
tal	732.99	1287.93	1.76	1308.89	1.79
france	9825	16300	1.66	16456.2	1.67
norway	61.41	97.66	1.59	98.67	1.61
cost266	14587.46	23587.58	1.62	23988.5	1.64
avg			1.56		1.59

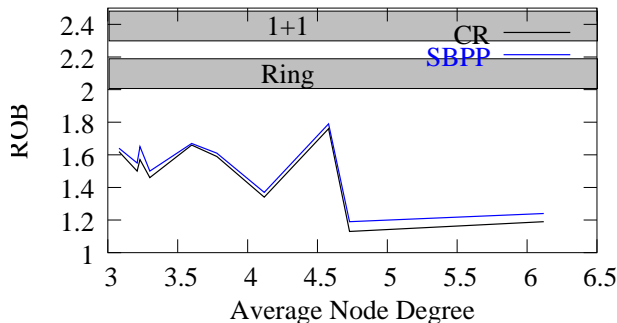
Table: Network protection requirements for SBPP protection compared to NF and CR [3]

Graphical Setup



- Very little difference !
- But also large architecture dependence
- What about the Average node degree ?

Graphical Setup



- It is difficult to conclude something based on so few examples
- RROB seems to be dropping as the average node degree grows ...
- ... but there are large variations ...

Efficiency Comments

The results seems to indicate:

- SBPP protection is **very** efficient ...
- ... can at most be improved by less than 5%

Running time: Brute force version

	It	Total Sec.	Master Sec.	%	Sub Sec.	%
newyork	1622	123284.15	3461.59	2.81	119818.33	97.19
tal	1190	5850.22	840.66	14.37	4998.58	85.44
france	1100	3003.47	722.55	24.06	2271.49	75.63
norway	1593	16612.94	2848.51	17.15	13748.79	82.76
cost266	2079	26075.23	12758.92	48.93	13253.95	50.83

Table: Running times in seconds

Running Time Comments

What can be said about the running times ?

- Solution of the sub-problem (QCDPP) dominates the execution time, as expected ...
- ... **but** for the largest example, the solution time of the master problem is significant

On-line routing problem

QCDPP routing problem is also important in the On-Line case:

- Given a network, how to allocate SBPP connections one by one ?
- If we have the β_a^s values (or approximations of it) we can route SBPP connections by solving the QCDPP

Open Problems/Extension for SBPP

- Stub release SBPP, releasing capacity from primary path on unfailling arcs
- Not Bifurcation, i.e. all circuits for a demand is sent along the same path-pair (req. Branch-and-Price)
- Modular Capacities on edges, discount on large demand (req. Branch-and-Price)
- Capacity limitations on edges
- Node failure protection
- Double edge protection, protection against multiple failures
- Full Backup Path Protection, it is hard just to write the model



Grover W. D.
Mesh-Based Survivable Networks.
Prentice Hall PTR, 2004.



Michael R. Garey and David S. Johnson.
Computers and Intractability. A guide to the theory of NP-completeness.
Freeman, 1979.



T. Stidsen and P. Kjærulff.
Complete rerouting protection.
Journal of Optical Networking, 2005.
accepted.



Thomas Stidsen, Bjørn Petersen, Kasper Bonne Rasmussen,
Simon Spoorendonk, and Martin Zachariasen.
Optimal routing with single backup path protection

Working Paper, 2006.



G. Swallow and L. Andersson.

Mpls working group.

<http://www.ietf.org/html.charters/mpls-charter.html>.



R. Wessly and M. Piro.

Snd-lib: The data source for all people working on optimizations of telecommunications networks.

<http://sndlib.zib.de/home.action>.