

Primary/Secondary Path Generation Problem: Reformulation, Solutions and Comparisons

Quanshi Xia and Helmut Simonis

IC-Parc, Imperial College London
London SW7 2AZ, UK
{q.xia, h.simonis}@imperial.ac.uk

Abstract. This paper considers the primary and secondary path generation problem in traffic engineering. We first present a standard MILP model. Since its size and integrality gap are very large, we then apply a Benders decomposition to isolate the failure case capacity constraints, related linearisation variables and linearisation constraints.

The disaggregated Benders cuts are generated, which is actually the set of *violated* failure case capacity constraints with their linearisation variables and the required linearisation constraints. This corresponds to adding the failure case capacity constraints, their linearisation variables and linearisation constraints only as they are needed.

Some results on generated test cases for different network topologies are given. In comparison with the standard MILP formulation, we reduce execution times on average by a factor of 1000 using the Benders decomposition. We also compare with a scheme of accepting demands one-by-one, which can handle more large-scale problems at the cost of losing optimality.

1 Introduction

It is currently common practice for Internet service providers (ISP) to run their network as a single class of service (CoS) over an IGP routed network. However, this best-effort traffic cannot guarantee quality of service (QoS) [1] parameters.

Using the emerging MPLS technology, some ISPs are switching to a traffic engineered operation, where paths between nodes in the network are specified explicitly. This can help to provide some guaranteed QoS, such as the bandwidth provision or delay and jitter bounds [2].

For services like VoIP (Voice over IP) and video conferencing, the customer needs a reliable service without disruptions even if some elements in the network fail. Fast re-route tunnels can provide an emergency alternative in a very short period. These tunnels can be precomputed and set-up on the network, or example with **Cisco's Tunnel Builder Pro** product. However, as a solution for longer lasting outages a secondary (backup) path needs to be provided as well [3]. This secondary tunnel is activated when the failure of the primary tunnel is signalled to the head router. As the secondary tunnel should not be affected by the element failure that might affect the primary tunnel, we must require that it is disjoint

from the primary path. Most often we only consider link element failures, and the secondary path is not allowed to use the same links as the primary path. If we want to protect against node failure as well, we have to enforce that primary and secondary paths are node disjoint, i.e. except for source and destination they do not use the same nodes. Enforcing this stronger version of disjointness will require more network resources, and may be impossible on certain topology structures.

Only reserving the full bandwidth for the primary path and setting up zero bandwidth tunnels for the secondary paths will minimize the required network resources, but can't guarantee QoS when an element fails. On the other hand, reserving the full bandwidth for both primary and secondary paths at the same time will result in an inefficient usage of the network, i.e. treating the primary path and the secondary path equally will require too much bandwidth. As primary and secondary tunnels for the same demand are never used concurrently, we can account for the bandwidth required in a more intelligent way. In the normal network operation, only the primary tunnels are used and require bandwidth. The sum of all bandwidth demands routed over a given link should be within the capacity of the link. The same principle applies in a failure case. Most of the primary tunnels will be still active, only those routed over failed elements will be replaced by their secondary tunnels. The total bandwidth required by all primary tunnels still in use and all active secondary tunnels should not exceed the total capacity on each link.

To express all capacity constraints for all failure cases leads to a large number of variables and constraints. We use Benders decomposition to avoid stating all these constraints *a priori*. In our master problem we find primary and secondary paths, and then check in our subproblem if any capacity constraints are violated. We add the violated capacity constraints as Benders cuts to the master problem until a solution is found which doesn't violate the capacity constraints. By construction, this is the optimal solution to the global problem.

The paper is structured in the following sections. In section 2 we review a MIP model to solve the path generation problem. We then reformulate this model as a MILP, introducing linearization variables and constraints. In section 4, we show an alternative formulation using a Benders decomposition. Comparative results are given in section 5.

2 Primary/secondary path generation problem

Recently, Papadakos provided a model [4] to find primary and secondary paths, accounting for the bandwidth use in normal operation and in all failure cases separately. The bandwidth required by primary and secondary paths which are active in each failure case should not exceed the edge capacity. In this section we summarize the results of that paper.

The network is modelled as a digraph $\mathbf{G}=(\mathbf{N},\mathbf{E})$, which comprises a set of nodes, \mathbf{N} and a set of edges, \mathbf{E} . Each edge $e = e(i, j) \in \mathbf{E}$ directly connects node

i to node j , associated with a capacity $\text{cap}(e)$. For each node $n \in \mathbf{N}$ there is a set of edges $\mathbf{IN}(n)$ entering n and a set of edges $\mathbf{OUT}(n)$ leaving n .

The set of demands to be routed over the network is called \mathbf{D} . Each demand $d \in \mathbf{D}$ has an origin $\text{orig}(d)$, a destination $\text{dest}(d)$ and a maximum requested bandwidth $\text{bw}(d)$.

For each demand, if accepted, a primary and a secondary path must be found, so that in the normal operation and in all failure cases the bandwidth required by primary and secondary paths active in this failure case do not exceed the available edge capacity.

A model of this problem is derived in the following way: We introduce three sets of (0/1) variables as:

acceptance variable Z_d states whether demand d is accepted or not.

primary path variable X_{de} states whether edge e is used for the primary path of demand d .

secondary path variable W_{de} states whether edge e is used for the secondary path of demand d .

We then introduce four sets of constraints as:

primary path constraint states that for an accepted demand there must be a continuous primary path from the source to the destination.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

secondary path constraint states for an accepted demand there also must be a continuous secondary path from the source to the destination.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} W_{de} - \sum_{e \in \mathbf{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

edge disjoint path constraint states that primary and secondary paths are not allowed to use the same edges.

$$\forall d \in \mathbf{D}, \forall e \in \mathbf{E} : X_{de} + W_{de} \leq 1$$

capacity constraints states that in the normal operation and in all failure cases the bandwidth required by primary and secondary paths active in this failure case do not exceed the edge capacity.

$$\begin{cases} \forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \text{bw}(d) * X_{de} \leq \text{cap}(e) \\ \forall e \in \mathbf{E}, e' \in \mathbf{E}/e : \sum_{d \in \mathbf{D}} \text{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) \leq \text{cap}(e) \end{cases}$$

The capacity constraint can be explained in the following way. The bandwidth use for edge e is limited both by the use for all primary paths routed through it and by the largest use in any of the failure cases considered. Lets assume a failure in edge e' . Then the traffic through e is the sum of all primary paths passing through e which are not passing through e' as well, and the sum of all secondary paths through e for demands where the primary path is routed through e' .

Finally, the MIP formulation is presented as:

$$\begin{aligned}
& \max_{\{Z_d, X_{de}, W_{de}\}} \sum_{d \in \mathbf{D}} \text{bw}(d) * Z_d \\
& \text{st.} \left\{ \begin{array}{l}
\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\
\forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \text{bw}(d) * X_{de} \leq \text{cap}(e) \\
\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \text{OUT}(n)} W_{de} - \sum_{e \in \text{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\
\forall e \in \mathbf{E}, \forall e' \in \mathbf{E}/e : \sum_{d \in \mathbf{D}} \text{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) \leq \text{cap}(e) \\
\forall d \in \mathbf{D}, \forall e \in \mathbf{E} : X_{de} + W_{de} \leq 1
\end{array} \right. \quad (1)
\end{aligned}$$

3 Problem reformulation

The MIP formulation in the previous section was using a number of non-linear constraints. Differing from [4], we propose a new linearisation. For this, we introduce new 0/1 variables $Y_{dee'} = (1 - X_{de'}) * X_{de} + X_{de'} * W_{de}$ (called **linearisation variables**) and a set of linearised inequalities (called **linearisation constraints**):

$$\begin{cases} X_{de} - X_{de'} \leq Y_{dee'} \leq X_{de} + X_{de'} \\ W_{de} + X_{de'} - 1 \leq Y_{dee'} \leq W_{de} - X_{de'} + 1 \end{cases} \quad (2)$$

This results in a new MILP formulation:

$$\begin{aligned}
& \max_{\{Z_d, X_{de}, W_{de}, Y_{dee'}\}} \sum_{d \in \mathbf{D}} \text{bw}(d) * Z_d \\
& \text{st.} \left\{ \begin{array}{l}
\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\
\forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \text{bw}(d) * X_{de} \leq \text{cap}(e) \\
\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \text{OUT}(n)} W_{de} - \sum_{e \in \text{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\
\forall e \in \mathbf{E}, \forall e' \in \mathbf{E}/e : \sum_{d \in \mathbf{D}} \text{bw}(d) * Y_{dee'} \leq \text{cap}(e) \\
\forall e \in \mathbf{E}, \forall e' \in \mathbf{E}/e, d \in \mathbf{D} : \begin{cases} X_{de} - X_{de'} \leq Y_{dee'} \leq X_{de} + X_{de'} \\ W_{de} + X_{de'} - 1 \leq Y_{dee'} \leq W_{de} - X_{de'} + 1 \end{cases} \\
\forall d \in \mathbf{D}, \forall e \in \mathbf{E} : X_{de} + W_{de} \leq 1
\end{array} \right. \quad (3)
\end{aligned}$$

which can be solved by commercial MIP solvers like CPLEX [5]. However, this MILP formulation has (1) a large number of failure case capacity constraints

$(|\mathbf{E}| * (|\mathbf{E}| - 1))$; (2) a huge number of linearisation variables $(|\mathbf{D}| * |\mathbf{E}| * (|\mathbf{E}| - 1))$; (3) a huge number of linearisation constraints $(4 * |\mathbf{D}| * |\mathbf{E}| * (|\mathbf{E}| - 1))$. This means that the *integrality gap* is very big, therefore, limiting the scalability of the model. Although we can solve smaller problem instances, both memory usage and execution time grow very quickly for larger problem instances.

4 Solution by Benders decomposition

Benders decomposition provides an alternative for solving large scale combinatorial optimization problems. It partitions an optimisation problem into two smaller problems, the *master problem* and the *subproblem*. The Benders algorithm iteratively solves the master problem and the subproblem. In every iteration, the subproblem solution provides the *Benders cut*, added to the master problem, narrowing down the search space and leading to optimality [6].

Applying the Benders decomposition to the problem (3), we choose the failure case capacity constraints as the subproblem (SP), and the primary/secondary path generation as the master problem (MP). This decomposition will isolate the failure case capacity constraints and their linearisation variables and linearisation constraints and dramatically reduces the size of the master problem.

4.1 SP – Failure case capacity constraint

Suppose the primary/secondary path solution $\{\tilde{X}_{de}^{(k)}, \tilde{W}_{de}^{(k)}\}$. The capacity constraints for the failure cases are checked by the subproblem:

$$\begin{aligned} \min_{\{S_{ee'} \geq 0\}} \quad & \phi = \sum_{e \in \mathbf{E}} \sum_{e' \in \mathbf{E}/e} S_{ee'} \\ \text{st.} \quad & \begin{cases} \forall e \in \mathbf{E}, \forall e' \in \mathbf{E}/e : \\ S_{ee'} \geq \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (\tilde{X}_{de}^{(k)} - \tilde{X}_{de'}^{(k)} * \tilde{X}_{de}^{(k)} + \tilde{X}_{de'}^{(k)} * \tilde{W}_{de}^{(k)}) - \mathbf{cap}(e) \end{cases} \end{aligned} \quad (4)$$

If the objective solution $\phi^{(k)} = 0$, then all failure case capacity constraints are satisfied. Therefore the optimal solution is obtained as $\{\tilde{Z}_d^{(k)}, \tilde{X}_{de}^{(k)}, \tilde{W}_{de}^{(k)}\}$. However, if $\phi^{(k)} > 0$, which means that not all failure case capacity constraints are satisfied. Then new Benders cuts must be generated, and must be added to the master problem to cut off the solution $\{\tilde{X}_{de}^{(k)}, \tilde{W}_{de}^{(k)}\}$.

In order to generate the Benders cut, the dual of subproblem (4)

$$\max_{\{0 \leq \gamma_{ee'} \leq 1\}} \sum_{e \in \mathbf{E}} \sum_{e' \in \mathbf{E}/e} \left[\sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (\tilde{X}_{de}^{(k)} - \tilde{X}_{de'}^{(k)} * \tilde{X}_{de}^{(k)} + \tilde{X}_{de'}^{(k)} * \tilde{W}_{de}^{(k)}) - \mathbf{cap}(e) \right] * \gamma_{ee'}$$

is solved to get the solution:

$$\tilde{\gamma}_{ee'}^{(k)} = \begin{cases} 0 & \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (\tilde{X}_{de}^{(k)} - \tilde{X}_{de'}^{(k)} * \tilde{X}_{de}^{(k)} + \tilde{X}_{de'}^{(k)} * \tilde{W}_{de}^{(k)}) \leq \mathbf{cap}(e) \\ 1 & \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (\tilde{X}_{de}^{(k)} - \tilde{X}_{de'}^{(k)} * \tilde{X}_{de}^{(k)} + \tilde{X}_{de'}^{(k)} * \tilde{W}_{de}^{(k)}) > \mathbf{cap}(e) \end{cases}$$

Then the Benders cut, which will make the solution $\{\tilde{X}_{de}^{(k)}, \tilde{W}_{de}^{(k)}\}$ infeasible, is generated as

$$\sum_{(e,e') \in \mathbf{C}^{(k)}} \left[\sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) - \mathbf{cap}(e) \right] \leq 0 \quad (5)$$

$$\text{where } \mathbf{C}^{(k)} = \{(e, e') : \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (\tilde{X}_{de}^{(k)} - \tilde{X}_{de'}^{(k)} * \tilde{X}_{de}^{(k)} + \tilde{X}_{de'}^{(k)} * \tilde{W}_{de}^{(k)}) > \mathbf{cap}(e)\}$$

Since the subproblem (4) is decomposable [7], the Benders cut (5) can be disaggregated as:

$$\forall (e, e') \in \mathbf{C}^{(k)} : \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) \leq \mathbf{cap}(e) \quad (6)$$

These new Benders cuts are much *stronger* than Benders cut (5).

By use of the related linearisation variables and the linearisation constraints, the Benders cuts (6) can be linearised as

$$\left\{ \begin{array}{l} \forall (e, e') \in \mathbf{C}^{(k)} : \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * Y_{dee'} \leq \mathbf{cap}(e) \\ \forall (e, e') \in \mathbf{C}^{(k)}, \forall d \in \mathbf{D} : \begin{cases} X_{de} - X_{de'} \leq Y_{dee'} \leq X_{de} + X_{de'} \\ W_{de} + X_{de'} - 1 \leq Y_{dee'} \leq W_{de} - X_{de'} + 1 \end{cases} \end{array} \right. \quad (7)$$

We then add these Benders cuts to the master problem to narrow down its feasible solution space.

Actually, the linearised Benders cuts (7) are those failure case capacity constraints with their related linearisation variables and linearisation constraints which are *violated* by the primary/secondary path solution $\{\tilde{X}_{de}^{(k)}, \tilde{W}_{de}^{(k)}\}$. Note that the Benders cuts do not include the failure case capacity constraints which are not violated, neither their linearisation variables nor their linearisation constraints.

4.2 MP – Generating primary/secondary paths

In the next iteration we collect all Benders cuts generated by subproblem, and construct a new master problem

$$\begin{array}{l} \max_{\{Z_d, X_{de}, W_{de}, Y_{dee'}\}} \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * Z_d \\ \text{st.} \left\{ \begin{array}{l} \forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathbf{dest}(d) \\ Z_d & n = \mathbf{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\ \forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * X_{de} \leq \mathbf{cap}(e) \\ \forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} W_{de} - \sum_{e \in \mathbf{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \mathbf{dest}(d) \\ Z_d & n = \mathbf{orig}(d) \\ 0 & \text{otherwise} \end{cases} \\ \forall (e, e') \in \mathbf{E}^{(k)} = \mathbf{E}^{(k-1)} \cup \mathbf{C}^{(k)} : \sum_{d \in \mathbf{D}} \mathbf{bw}(d) * Y_{dee'} \leq \mathbf{cap}(e) \\ \forall (e, e') \in \mathbf{E}^{(k)} = \mathbf{E}^{(k-1)} \cup \mathbf{C}^{(k)}, \forall d \in \mathbf{D} : \begin{cases} X_{de} - X_{de'} \leq Y_{dee'} \leq X_{de} + X_{de'} \\ W_{de} + X_{de'} - 1 \leq Y_{dee'} \leq W_{de} - X_{de'} + 1 \end{cases} \\ \forall d \in \mathbf{D}, \forall e \in \mathbf{E} : X_{de} + W_{de} \leq 1 \end{array} \right. \quad (8) \end{array}$$

We resolve the master problem to find the new primary/secondary path solution $\{\tilde{X}_{de}^{(k+1)}, \tilde{W}_{de}^{(k+1)}\}$, and then check the satisfaction of the failure case capacity constraints in our subproblem. Initially, the master problem is solved without any failure case capacity constraints, *i.e.* $\mathbf{E}^{(0)} = \emptyset$.

It is worth drawing attention to the difference between the MILP formulation (3) and the master problem (8): only a few failure case capacity constraints and their related linearisation variables and linearisation constraints involved in master problem (8). This makes the master problem (8) easier to solve, at least as long as we do not have to add too many cuts in the process.

4.3 Adding constraints and variables as needed

Instead of stating all failure case capacity constraints *at the very beginning*, we start to solve the primary/secondary path generation problem without the failure case capacity constraints, find a primary/secondary path solution and check which failure case capacity constraints are violated. If no violations are detected, then the solution is optimal. If violations are found, then we add *violated* failure case capacity constraints and related linearisation variables and linearisation constraints, and resolve the master problem. They act as a cutting plane, *i.e.* the previous solution becomes infeasible and a new solution must be found, which does not have the defect of the previous solution. In the worst case, we need to add all failure cases capacity constraints before finding a solution.

This strategy is similar to one which used when solving the Travelling Salesman Problem (TSP) with MILP. In order to efficiently handle the exponential number of sub-tour elimination constraints, such constraint are added only when they are needed. The initial solution consists in solving the problem without any sub-tour elimination constraint and to add such constraints only if the current optimal solution contains such the sub-tour. Usually, only a limited number of sub-tour elimination constraints need to be added [8].

However, the sub-tour elimination constraints are linear, while the failure case capacity constraints are nonlinear, and need a large number of linearisation variables and linearisation constraints to make them linear. This means that in our case, the strategy of *adding constraints and variables as needed* is even more powerful.

5 Implementation, results and comparison

Our current implementation consists of a single MILP master problem (8) and, in principle, LP subproblems for each failure case capacity constraint. However, the subproblem is reduced to a checker for capacity constraint violation. The efficient handling of the MILP master problem and addition of new rows (constraints) to the master problem is supported by the hybrid MILP/CP software platform ECLⁱPS^e [9], which provides interfaces to CPLEX [5] for solving MILP problems.

Three different network topologies have been used to evaluate our algorithm, these networks have between 10 and 38 nodes and 30 to 116 directed edges. We

choose demand sets from 10 to 40 demands, and divided the bandwidth required into 2 classes (small and large). The small bandwidth is randomly chosen from 100 to 500 in increments of 50, and the large bandwidth from 100 to 2000 in increments of 200.

For the small bandwidth demand group, the Benders decomposition can solve all problem instances very easily requiring only between one and three iterations. Table 1 shows the objective as the bandwidth size of the accepted demands, the MILP model solution time (CPU) and size in number of variables (Vars) and constraints (Cstrs), and Benders decomposition solution time (CPU) and iterations (IT). As the demand bandwidth increases, the Benders decomposition requires more CPU time and more iterations to find the optimal solution, results are shown in Table 2. CPU times are given for a Linux based workstation with a 2 GHz Pentium4, OM indicates out of memory on a 1GB machine, and TO indicates a time out after 72000 seconds.

The experiments show that the Benders decomposition is quite sensitive to the problem difficulty. If the failure case capacity constraints are easy to satisfy, then the method will terminate after a few iterations. But when the failure case capacity constraints are hard to satisfy, then we iterate many times and require

Network		Demands Number	Obj	MILP			BD	
Nodes	Edges			CPU	Vars	Cstrs	CPU	IT
10	30	10	2650	2.79	9311	36401	0.58	2
		20	4700	144.96	18621	71901	1.38	2
		30	6950	OM	27931	107401	340.51	2
20	66	10	2650	536.26	44231	177417	1.28	1
		20	4550	5564.54	88461	350477	5.40	2
		30	6800	12989.33	132691	523537	10.88	2
		40	8500	OM	176921	696597	25.08	3
38	116	10	2150	33822.51	132497	536785	4.17	1
		20	3050	OM	266379	1065635	8.88	1
		30	5200	OM	398875	1588963	27.34	2

Table 1. Small Bandwidth Demand

Network		Demands Number	Obj	MILP			BD	
Nodes	Edges			CPU	Vars	Cstrs	CPU	IT
10	30	10	6000	4594.88	8093	31676	5.01	2
		20	8400	TO	16881	65151	419.92	4
		30		OM	25147	96601	TO	
20	66	10	8600	4164.53	39751	159742	10.17	6
		20	13000	OM	82061	325227	3381.68	6
		30		OM	122451	483137	TO	
		40		OM	160921	633472	TO	
38	116	10	7000	11516.08	110041	447409	12.13	3
		20	8400	OM	232583	931235	56.89	6
		30	12600	OM	346095	1379435	6650.73	11

Table 2. Large Bandwidth Demand

more CPU time. However, in our 20 test instances, the Benders decomposition solves all but 3 problem instances within the time limit, requiring between 1 and 11 iterations. Of all 20 test instances, the MILP cannot solve 11 problem instances within the time limit, and for the 9 solved instances, MILP is not efficient and takes on average 1000 times longer than the Benders decomposition.

Note that for large problem instances the master problem may already provide a challenge for a MILP solver. But we can produce an approximation scheme for our problem where the master problem is not solved to optimality or not even solved with MILP, and where the subproblem is generating new constraints that must be added to the master problem [10]. In this scheme we lose optimality and completeness of the method, but can expect good, feasible solutions. The only requirement is that the solver for the master problem can accept new constraints on violated failure case capacities.

Place the demands one by one As an alternative scheme, we can try to place demands one by one. Usually, we will not obtain the optimal solution to the global problem, but we can handle much larger problem sizes. We consider two variants of this scheme.

Suppose that at step k , \mathbf{D}^k is the set of demands which previously have been tried to be placed in the network (\tilde{Z}_d accepted or rejected), therefore we know their primary/secondary path solution $(\tilde{X}_{de}, \tilde{W}_{de})$. And at step k , we try to place a set of new demands, $\mathbf{D}^{(k)}$, keeping already placed ones fixed. We call the corresponding optimisation problem **fXfW**.

We can also allow to reroute all secondary tunnels, keeping only the primary paths of previously accepted demands fixed. Therefore, when placing new demands at step k , we can treat the secondary paths of previous accepted demands as additional variables. We call the corresponding optimisation problem **fXvW**.

Currently, the optimisation problems (**fXfW** and **fXvW**) have been formulated and solved only by MILP solvers, although the Benders decomposition scheme could be applied as well. When we compare this approach with the Benders decomposition results in terms of the number of accepted demands, the bandwidth size and the cpu time, which are given in tables 3 and 4, we can see that (**fXfW** and **fXvW**) cannot guarantee the optimality although they can solve all problem instances.

6 Conclusion

When reformulated as a MILP, the primary/secondary path generation problem needs a large number of linearisation variables and linearisation constraints. Because of the big integrality gap, it is quite difficult to solve with a commercial MILP package. In this paper, we have applied a Benders decomposition to solve the primary/secondary path generation problem. Isolating the failure case capacity constraints and their corresponding linearisation variables and linearisation constraints from the other constraints, and adding them only as needed, allows us to solve more realistic problem instances.

References

1. X. Xiao and L.M. Ni, “Internet QoS: a Big Picture”, IEEE Networks, (1999)
2. D. Awduche et al, “Requirements for Traffic Engineering over MPLS”, Internet Draft - draft-ietf-mpls-traffic-eng.00.txt, (1998)
3. Cisco Inc., “MPLS Bandwidth Protection”, White Paper(2001)
4. N.P. Papadakos, “Optimising Network Utilisation with Backup Routes”, M.Sc. Thesis, Imperial College London (2002)
5. ILOG Inc., “ILOG CPLEX 6.5 User’s Manual”, (1999)
6. J.F. Benders, “Partitioning Procedures for Solving Mixed Variables Programming Problems”, Numerische Mathematik, Vol.4, 238-252, (1962)
7. L.S. Lasdon, “Optimization Theory for Large Systems”, MacMillan, New York (1970)
8. G. Pataki, “Teaching Integer Programming Formulation Using the Travelling Salesman Problem”, SIAM Review 45(1) (2003)
9. Imperial College London, “ECLⁱPS^e 5.6 User’s Manual”, (2003)
10. G. Zakeri, A.B. Philpott and D.M. Ryan, “Inexact Cuts in Benders Decomposition”, SIAM Journal of Optimization, Vol.10, No.3, 634–657, (2000)

Network		Demands Number	BD			fXfW			fXvW		
Nodes	Edges		Nr	Obj	CPU	Nr	Obj	CPU	Nr	Obj	CPU
10	30	10	10	2650	0.58	10	2650	2.44	10	2650	3.09
		20	20	4700	1.38	20	4700	15.39	20	4700	13.05
		30	29	6950	340.51	29	6600	21.97	29	6600	25.34
20	66	10	10	2650	1.28	10	2650	87.11	10	2650	97.71
		20	19	4550	5.40	19	4550	561.25	19	4550	519.25
		30	28	6800	10.88	28	6800	855.19	28	6800	707.69
		40	35	8500	25.08	35	8500	1078.57	35	8500	879.90
38	116	10	8	2150	4.17	8	2150	8327.43	8	2150	6479.58
		20	13	3050	8.88	13	3050	10349.17	13	3050	8112.11
		30	22	5200	27.34	22	5200	10748.35	22	5200	8762.08

Table 3. Small Bandwidth Demand

Network		Demands Number	BD			fXfW			fXvW		
Nodes	Edges		Nr	Obj	CPU	Nr	Obj	CPU	Nr	Obj	CPU
10	30	10	8	6000	5.01	8	5600	2.82	8	5600	3.52
		20	14	8400	419.92	15	8100	7.72	15	8100	9.78
		30	TO			20	10600	10.7	21	11900	17.81
20	66	10	8	8600	10.17	10	8600	85.84	10	8600	112.00
		20	18	13000	3381.68	17	11300	329.55	17	11300	252.51
		30	TO			24	16000	427.23	23	15300	357.21
		40	TO			26	17400	464.63	28	19000	414.72
38	116	10	8	7000	12.13	8	7000	775.22	8	7000	588.78
		20	12	8400	56.89	12	8400	1284.23	12	8400	1262.42
		30	18	12600	6650.73	17	10700	1412.44	18	12600	1602.65

Table 4. Large Bandwidth Demand