Balanced paths in telecommunication networks: some computational results

Paola Cappanera, *Dipartimento di Sistemi e Informatica, Università di Firenze, Italy*
Maria Grazia Scutellà, *Dipartimento di Informatica, Università di Pisa, Italy*

Keywords: node-disjoint balanced paths, color-coding approach, telecommunication instances

1. Introduction

Given a weighted directed network G, we consider the problem of computing k *balanced* paths from given source nodes to given destination nodes of G, i.e. k paths such that the difference in cost between the longest and the shortest path is minimized. The special case in which G is an acyclic network was theoretically investigated in [3]. It was proved that, in such a special case, the balanced path problem is solvable in pseudo-polynomial time. When arc-disjoint or node-disjoint balanced paths are looked for, then the problem is strongly NP-Hard.

Although not yet investigated by the OR scientific community, apart from the preliminary results mentioned above, balanced path problems arise in several interesting applications, both in transportation and in telecommunications.

Inspired by some of the algorithmic ideas outlined in [3], here we examine the balanced path problem in a more general setting, focusing on telecommunication applications. As typical in telecommunications, the focus is on a survivability scenario. The k balanced paths to be determined are thus required to be node-disjoint. A general algorithmic framework is described, which is based on the color-coding method for computing simple paths [1]. Several algorithms were derived from this framework. The family of the proposed methods includes both an exact approach as well as alternative heuristic approaches. The algorithms were tested on a large suite of instances which were generated starting from benchmark telecommunication instances: the instances of the library SNDlib, available at http://sndlib.zib.de/, and the benchmark instances at ftp://ftp.elet.polimi.it/users/Pietro.Belotti/mcf/data/. The computational results are very interesting. They show that in most cases the exact color-coding algorithm produces the optimal solution more rapidly than the commercial software CPLEX. They also show that some of the proposed heuristics are able to generate high quality solutions in a very quick time.

2. The balanced path problem

Let $G = (V, E)$ be a directed network, with $|V| = n$ and $|E| = m$, and c_{ij} be the nonnegative integral cost associated with (i, j), for each $(i, j) \in E$. Given k source nodes $\{s_1, s_2, \ldots, s_k\}$ and k destination nodes $\{t_1, t_2, \ldots, t_k\}$, $k \geq 2$, the *balanced path problem* consists of computing a set of k paths linking the k source nodes to the k destination nodes in such a way that the difference in cost between the longest and the shortest path in the set is minimized. $\{s_1, s_2, \ldots, s_k\}$ are thus ingress nodes of the network, which must be linked to the egress nodes $\{t_1, t_2, \ldots, t_k\}$ in a balanced way. If we introduce two additional nodes s and t, such that s is connected to each node s_i through a zero cost arc (s, s_i), and t is connected to each node t_i through a zero cost arc (t_i, t), then the balanced path problem can be equivalently formulated as the problem of computing k paths from s to t, involving the k arcs of type (s, s_i) and the k arcs of type (t_i, t), which are balanced according to the definition stated above. Hereafter we will impose that the computed balanced paths are node-disjoint. Similar algorithmic results can be obtained if arc-disjoint balanced paths have to be determined. In addition to the above stated *single commodity version* of the problem, a *multicommodity version* is also investigated. In this case, given k origin-destination pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, $k \geq 2$, the problem consists of computing a set of k node-disjoint paths, linking the k origin-destination pairs, in such a way that the difference in cost between the longest and the shortest path in the set is minimized.

Balanced path problems arise in several applications, both in transportation and in telecommunications. In the field of transportation, the use of paths to represent weekly or monthly work assignments is wide spread in crew-scheduling problems; see for instance [2], where balancing aspects refer to an even distribution of the workload among the crew members. In the field of telecommunications, balancing aspects may arise in routing "noninterfering" messages through a communication network. For example, Li, McCormick and Simchi-Levi [6] addressed a related problem, consisting of computing two disjoint paths, from a super origin s to a super destination t, in such a way as to minimize the cost of the longest path. Instead in the balanced path problem the difference in cost between the longest and the shortest path has to be minimized. To the best of our knowledge, this kind of "balanced-type" objective function has never been investigated in the context of (disjoint) path computations. An interesting application of the balanced path problem in the context of telecommunication networks is the load balancing of multiple Quality of Service (QoS) paths in the IP Telephony service, as described in [5]. The authors investigated different kinds of path management schemes, involving either single or multiple (partially) disjoint paths, to guarantee a good robustness against node and/or arc failures of the network. One key observation of the authors is that, in the case of schemes that spread packets over multiple disjoint paths, an important factor is the end-to-end delay difference between the longest and the shortest path in each path set determined. This delay difference should be maintained as low as possible. This factor is related to the so-called jitter factor, and it can be critical especially in contexts such as video/audio conference interconnections and the conduction of time critical experiments in a virtual collaboration. Minimizing the jitter factor over a set of disjoint paths naturally leads to the formulation of the balanced path problem. Note that balanced paths can also be relevant for reasons of equity, in order to guarantee an even response time for all the users of the network, when modelled in terms of given origin-destination pairs.

3. The color-coding algorithmic framework

Let us consider the single commodity version of the problem, making use of a super source s and of a super destination t. Firstly we will assume that no disjointness constraint is imposed on the paths to be balanced and U denotes an upper bound to the cost of the longest path in G. Given U, let us introduce the *cost-expanded network* $G_U = (V_U, E_U)$, which is the unweighted layered network obtained from G by inserting s as well as s_1 ..., s_k in V_U, and replacing each other node i of G by $U+1$ copies i^p, $p = 0, 1, 2, \ldots, U$, where i^p represents node i when reached from the source s in G through paths of cost exactly p, $p = 0, 1, 2, \ldots, U$. Each arc (i, j) of G may therefore generate several arc copies in G_U, i.e. copies of type $(i^p, j^{p+c_{ij}})$, for each feasible index p. By construction, each path from s to t in G of cost p is mapped onto a directed path of G_U from node s to node t^p (i.e., to the copy t^p of the destination node t). G_U is a layered network that has $O(nU)$ nodes and $O(mU)$ arcs: its size is therefore pseudo-polynomial with respect to the size of the original network G. In order to solve the balanced path problem on G, we perform a binary search in the interval $[0, U]$. For each considered integer value δ in $[0, U]$, the approach constructs the family of subgraphs of G_U that are made up of the paths of G_U going from node s to the nodes in $\{t^p, \ldots, t^{p+\delta}\}$, for each suitable p, and it verifies whether at least one of these subgraphs contains k distinct paths. At the termination, the last returned set of paths constitutes an optimum (balanced) set of paths, and the corresponding δ is the optimum objective function value. When node-disjointness is relaxed, then the check in each subgraph of G_U can be performed via a suitable maximum flow computation. The approach thus checks $O(\log U)$ values in $[0, U]$ and, for each considered value δ, it computes a maximum flow in $O(U)$ networks that have $O(mU)$ arcs. Since the maximum flow value is at most k, then each maximum flow can be computed in $O(kmU)$ time. The proposed approach is therefore able to determine a balanced path set in $O(kmU^2 \log U)$ time.

Now, let us reintroduce the node-disjointness constraint. As stated before, this variant is strongly NP-Hard. In such a case, the idea is to impose the node-disjointness of the computed paths via the color-coding method as described in [1]. Let Col denote a set of ν colors (ν is a parameter of the approach), and r indicate a random coloring of the nodes of G with exactly ν colors. Let us define a set of k paths from s to t in G *colorful* if the nodes that are made up of these paths are colored by distinct colors under r. Given r, the idea is to apply the pseudo-polynomial approach previously described (where no disjointness constraint is imposed) using the coloring of the nodes as a means to guarantee, in each considered subgraph of the cost-

expanded network G_U, the computation of node-disjoint paths (recall that several copies of any node i can be present in G_U). More precisely, extend the coloring r to the nodes of G_U so that copies of a same node i in G_U are colored in the same way, i.e. by the same color as i in G under the coloring r. Next, apply a variant of the pseudo-polynomial approach where, for each considered subgraph of G_U generated during the binary search, k node-disjoint paths from s to t are looked for. The idea is to look for a colorful set of k paths in the subgraph considered; i.e. to use the coloring r to guarantee the computation of k paths which are node-disjoint (with respect to r). A colorful set of k paths can be modelled as a unique, colorful path from s to t of cardinality at most ν. We can look for a this kind of path by the same method proposed in [1] for computing simple maximum cost paths of fixed cardinality in a directed graph, in $O(2^\nu mU)$ time. Given a random coloring $r \in R$, the approach can therefore determine a balanced colorful set of paths, with respect to r, in $O(2^\nu mU^2 \log U)$ time. As indicated in [1], a list of random colorings of V exists so that, for each subset V' of V of cardinality at most ν (and thus, for each set of k node-disjoint paths from s to t of cardinality at most ν), there is a coloring in the list which assigns a distinct color to each node in V'. Therefore, if the cardinality of the balanced path solution is bounded from above by ν, and if one applies the above approach for each coloring r in the list, returning the best computed solution, then a deterministic algorithm can be designed which, at least theoretically, gives an exact solution to the node-disjoint balanced path problem. The cardinality of this list is $O(2^\nu \log n)$.

When no hypothesis can be made regarding the structure of G, then the time complexity of the color-coding approach is exponential in the worst case, according to the previously stated time complexity results. However, the proposed ideas can be suitably embedded into a heuristic framework, to efficiently guide the search towards good quality solutions. This can be done as follows. Choose a value for parameter ν, and then generate a list of random colorings with exactly ν colors. For each coloring r in this list, apply the previous approach, which looks for a balanced colorful set of paths with respect to r. At the termination, return the best computed solution. Observe that, if $\nu = n$, then we get an exact approach (in this case, one random coloring is sufficient to produce the optimum solution). On the other hand, when $\nu < n$, then a color-coding heuristic framework is obtained, which is parametric with respect to the cardinality of the list of the generated random colorings, say α. The color-coding approach, both in the exact and heuristic version, can be easily modified in order to solve, with the same time complexity, the case of given origin-destination pairs.

4. Exact and heuristic color-coding algorithms

In addition to the exact algorithm, which is obtained from the general framework described in Section 3. by setting $\nu = n$, several color-coding heuristics were investigated. Here we report the ones that produced the best computational results.

The 5+5 heuristic The *5+5* heuristic is based on the following observation. When performing the binary search, the approach may get trapped in a semi-interval of the binary search. Since a time limit of one hour was imposed for the execution of each algorithm during the computational experimentation, then a (possibly large) portion of the space of the feasible solutions might get left out of the search. To avoid this, a time limit of 5 minutes was imposed on the time elapsed from one improvement to the next. When this time limit is reached the search is stopped, and the algorithm jumps to the next range of the binary search interval. After two trials (*5+5*) the algorithm stops, returning the best computed solution.

The cost scaling heuristic In order to reduce the size of the cost-expanded graph, the following cost transformation was performed. Given a positive constant ϵ, scale the original arc costs c_{ij} so they live in the range $0 \ldots n/\epsilon$, and then truncate them to the nearest integer. That is

$$c'_{ij} = \left\lceil \frac{c_{ij}}{c_{max}} \frac{n}{\epsilon} \right\rceil, \tag{1}$$

where c_{max} denotes the maximum arc cost of network G. The *cost scaling* variant applies the color-coding approach using the transformed costs c'_{ij} instead of the original arc costs c_{ij}, $\forall (i,j) \in E$. As an effect of

the cost scaling, the size of the cost-expanded graph is now a non-increasing function of parameter ϵ. In fact, the cost-expanded graph now consists of $O(\frac{n^2}{\epsilon})$ nodes and of $O(\frac{mn}{\epsilon})$ arcs. As shown in Section 5., the cost scaling heuristic proved to be very effective at computing good quality solutions, which very often proved to be the optimum solutions.

5. The computational study

In this section we provide some computational results for a large suite of balanced path instances. The instances were generated on the basis of two sets of benchmark telecommunication instances. Specifically, two data sets were generated starting respectively from 29 and 52 telecommunication instances available at:

- the ZIB (`http://sndlib.zib.de`)
- the Politecnico of Milano (`ftp://ftp.elet.polimi.it/users/Pietro.Belotti/mcf/data`).

Hereafter, the first group of instances are referred to as *Sndlib* instances, while the second one are referred to as *Belotti* instances. For each original instance, two balanced path instances were generated, related respectively to the single and to the multicommodity scenario. In order to distinguish between the two cases, an *F* is appended to the instance name in the single commodity scenario, whereas, in the multicommodity case, a *T* is used instead. $k = \lceil \log n \rceil$ ingress and egress nodes and $k = \lceil \log n \rceil$ origin-destination pairs were generated for these instances starting from the origin-destination information of the original telecommunication instances.

For both families of instances, whenever possible we validated the quality of the solutions obtained by the proposed color-coding algorithms by a direct comparison with the optimal solutions found by the commercial mixed-integer optimizer CPLEX (version 9.1). CPLEX ran I.L.P. models of the balanced path problem, which were proposed both for the single and for the multicommodity scenario. A time limit (TL) of one hour of CPU time was set both for CPLEX as well as for each tested color-coding algorithm. Specifically, for each instance in the two data sets first we ran the exact color-coding algorithm, referred to as *ExactCC*. We compared ExactCC and CPLEX 9.1 both in terms of quality of the best solution obtained and computational time. This was done in order to characterize the overall behavior of the exact color-coding approach, in an attempt to classify the instances into "easy" or "difficult" (at least for ExactCC). In the following, for the two data sets separately, we report the number of times one solver outperformed the other and how many times they can be considered equivalent. Since these aggregated results seem to be encouraging and show that the exact color-coding approach alone can obtain quite good performances at least on a meaningful number of instances, we decided to focus on much smarter implementations of the color-coding framework just to cope with the most critical instances. In an attempt to reduce the computational burden of ExactCC on these instances, three heuristic variants of ExactCC were implemented, based on the following tools: (i) an additional time limit which results in the *5+5* heuristic variant described in Section 4.; (ii) use of a number of colors $\nu < n$; and (iii) use of the cost scaling device described in Section 4.. These three tools can be used separately or combined together. In our computational experience we observed that the combined use of two or three such tools might often result in much better performances than the ones obtained when only one tool is considered. However, here we prefer to report the gain or loss in efficiency of the three tools separately, to give the reader a clearer picture of their effect and contribution.

Before illustrating some of the results, it is worth mentioning another issue: after running the exact and the heuristic color-coding algorithms on the two groups of instances, we tried to understand what issues characterize the critical instances (at least for our approach), and we attempted to gain some insight as to how the instances could be classified into groups that reveal similar behaviors compared to the others. Indeed, the issues that made an instance critical proved to be quite difficult to ascertain. For these reasons we decided not to report aggregated results (apart from the summarized aggregated results concerning ExactCC), but we

					Cplex		ExactCC		
ProbName	n	m	k	Δ Red	Obj	Time	Init	Obj	Time
janos-euF	37	114	5	3,64	**109**	906,08	**170**	170	3600,00
janos-euT	37	114	5	3,64	134	0,97	170	134	1050,55
janos-usF	26	84	4	4,11	133	11,27	133	133	1402,84
janos-usT	26	84	4	4,11	133	0,39	133	133	198,72
ta1F	24	55	4	4,00	13	2,70	39	13	42,63
ta1T	24	55	4	4,00	36	0,01	∞	36	56,59
janos-us-caF	39	122	5	1,33	**104**	3600,00	143	**20**	3600,00
janos-us-caT	39	122	5	1,33	99	1758,07	143	99	1119,15
sunF	27	102	4	2,00	40	3600,00	44	40	1847,34
sunT	27	102	4	2,00	40	616,31	44	40	237,49

Table 1: ExactCC vs Cplex

preferred to show negative or positive results on subsets of instances. Thus, in the following we will give a snapshot of the computational performances which, in our opinion, provide some insight into the efficiency of the approaches for the two classes of instances. The computed balanced paths also satisfy interesting Quality of Service (QoS) requisites. In particular, on average they are quite "short". For details about the computational investigation and the results, please refer to [4].

All experiments were carried out on a bi-processor AMD Opteron 246 2 GHz, with 2 GB of RAM and under Linux, kernel 2.6.13-1. All algorithms were coded in C++ and compiled with GNU gcc 4.0.1.

The SndLib instances For the 58 instances of the SndLib test bed we got the following aggregated results: ExactCC beat CPLEX 19 times; CPLEX beat our exact approach 9 times; and the two approaches were equivalent 30 times, both in terms of quality of the best solution reported and computational time. The aggregated results are encouraging, which leads us to believe that the three heuristic variants previously described might be able to reduce the computational time required on the most critical instances, whilst keeping low the loss in quality of the best solution found. Next we comment on the subset of 10 particular Sndlib instances which proved to be most critical either for our exact approach or for CPLEX. The results concerning the 10 critical Sndlib instances are shown in Table 1: for each instance the columns of the table give the number n of nodes, the number m of arcs, the number k of paths to be balanced and the percent reduction of the width of the binary search interval, namely Δ Red, which is performed by all the color-coding algorithms during an initialization phase. In the subsequent columns a comparison between the two exact approaches is given in terms of objective function value and time (expressed in seconds); for our approach the feasible solution value, computed by the initialization phase, is also reported in column Init. Note that, for the first 6 instances, CPLEX outperforms ExactCC: in particular 5 out of 6 times ExactCC provides the same solution value as CPLEX in much longer times; in one case, the *janos-euF* instance, ExactCC gives a worse solution (see bold entries in the first row of the table). Apart from this instance the computational times required by CPLEX are limited (not over 12 sec.) while for our approach they can be quite high. The last 4 instances seem to be quite difficult for both solvers; however ExactCC outperformed CPLEX in terms of time and for one instance, namely *janos-us-caF*, it produced a much better quality solution, i.e. a solution with value of 20 instead of 104 (see bold entries in Table 1 - second group of instances). ExactCC was not able to certify the optimality of such a solution since it exceeded the time limit of one hour.

For the first 6 instances, which were the most critical ones, an attempt was made to reduce the computational burden of ExactCC via the *5+5* time limit tool. For all these instances the heuristic, named Heur5+5, gave the same solution value as ExactCC. The reduction of the time required by Heur5+5 with respect to ExactCC was meaningful at least for the first three most critical instances; in fact, the time required by the further time limited version is around 10 minutes. Nevertheless the computational times are still not competitive with the ones reported by CPLEX. Then, for the 10 critical instances in this test bed, we tested the cost

	Cplex	$\varepsilon = 3/2$		$\varepsilon = 2$		$\varepsilon = 5/2$		$\varepsilon = 3$		$\varepsilon = 10$	
ProbName	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
janos-euF	906	0,00	3600	0,00	3600	0,00	3194	3,67	3280	26,61	327
janos-euT	0,97	0,00	254	2,99	161	0,00	79	2,99	73	2,99	9,86
janos-usF	11	0,00	299	0,00	162	0,00	130	0,00	100	3,01	1,73
janos-usT	0,39	0,00	19	0,00	9,09	0,00	5,79	0,00	4,00	**0,00**	**0,20**
ta1F	2,70	**0,00**	**2,18**	0,00	2,14	38,46	0,94	38,46	0,90	176,92	0,19
ta1T	0,01	0,00	3,82	0,00	1,60	0,00	0,91	0,00	0,75	**0,00**	**0,02**
janos-us-caF	3600	-80,77	3600	-80,77	2141	-80,77	1665	-77,88	1431	-60,58	115
janos-us-caT	1758	**0,00**	**333**	0,00	216	0,00	183	0,00	118	5,05	6,47
sunF	3600	**0,00**	**393**	0,00	352	0,00	216	0,00	144	100,00	1,38
sunT	616	**0,00**	**38**	0,00	30	0,00	20	0,00	13	10,00	0,32

Table 2: Cost scaling variant

scaling heuristic with $\varepsilon \in \{3/2, 2, 5/2, 3, 10\}$. The results are reported in Table 2. For each instance the time required by CPLEX is given in addition to the time and the relative error (column Gap) of the solution reported by the heuristic with respect to CPLEX for each value of ε. Moreover, for each instance, in each row, the entries relating to the first value of ε for which the cost scaling heuristic becomes competitive with CPLEX are highlighted in bold. According to the results in Table 2, we can conclude that generally the cost scaling heuristic behaved very efficiently, producing good quality solutions in a very short time.

Concerning the tool number of colors, it should be emphasized that no significant improvement was obtained for the Sndlib test bed by choosing a number of colors $\nu < n$ (precisely, $\nu = 70\%n$ and $\nu = 80\%n$ were investigated). Thus, the computational results concerning the related heuristic variants are not reported here.

The Belotti instances As we did with the SndLib instances, we begin by giving an account of the aggregated results concerning the exact color-coding algorithm. For the 104 instances belonging to the Belotti data set, we got the following figures: ExactCC beat the commercial solver CPLEX 38 times , CPLEX beat the exact version of the approach 57 times, and finally the two approaches were equivalent 9 times both in terms of quality of the best solution reported and computational time involved. The Belotti instances thus appear to be more critical than the SndLib and a more comprehensive analysis is therefore required. Next we report some computational results related to subsets of instances which, in our opinion, highlight the main criticisms or the benefits in using the exact as well as the color-coding heuristic approach.

ExactCC overcomes Cplex We begin by comparing the results given by ExactCC and by CPLEX for a subset of 8 instances of *latadl* and *net* type, where our approach outperformed the commercial solver. These results are illustrated in Table 3. The last three columns show respectively the time required by CPLEX, the time required by ExactCC and the relative error of the solutions reported by ExactCC with respect to CPLEX. It should be noted that for all these 8 instances CPLEX always exceeded the time limit and reported worse solutions than the ones given by our approach (see negative values in column Gap). In particular, for three of these instances (see bold entries in Table 3), CPLEX failed to find a feasible solution, whereas our approach terminated in at most 11.13 sec. by solving the problem to optimality. There are also two instances which seem difficult to solve for the exact color-coding algorithm, as well, namely *net_30_20_F* and *net_30_99_F*. For such instances ExactCC was still able to determine better quality solutions than the ones found by the commercial solver, but it was not able to certify their optimality within the time limit considered. It is also interesting to observe that for these two instances the Heur5+5 variant obtained the same solution values as ExactCC in about 10 minutes. Except for these two instances, the time required by the exact color-coding approach varies between a minimum of 3.06 and a maximum of 76.37 sec.

Cplex prevails over ExactCC Now we will examine the computational results of a subset of 10 instances of *net* type (Table 4). We compared the performances obtained with CPLEX, ExactCC and the cost scaling

ProbName	n	m	k	U	CplexTime	ExactCCTime	Gap
latadl_5F	39	172	5	1950	**3600**	**11,13**	-∞
latadl_5T	39	172	5	1950	3600	4,07	-19,30
latadl_25F	39	172	5	1950	**3600**	**5,12**	-∞
latadl_99T	39	172	5	1950	**3600**	**3,06**	-∞
net_30_20F	30	142	4	13685	3600	3600,00	-37,23
net_30_60F	30	146	4	12052	3600	76,37	-51,04
net_30_60T	30	146	4	12052	3600	48,68	-39,23
net_30_99F	30	146	4	12627	3600	3600,00	-37,56

Table 3: ExactCC vs Cplex

				Cost scaling					
		ExactCC		$\varepsilon = 1/2$		$\varepsilon = 1$		$\varepsilon = 3/2$	
ProbName	CplexTime	Gap	Time	Gap	Time	Gap	Time	Gap	Time
net_15_99F	8,23	0,00	3600	**0,00**	**4,66**	31,37	1,62	11,07	0,64
net_20_5F	886,98	0,00	3600	**0,00**	**133,82**	0,00	45,98	0,00	31,96
net_20_5T	4,04	0,00	3600	0,00	34,71	0,00	7,73	0,00	5,23
net_20_20F	1884,04	0,00	3600	**0,00**	**26,53**	26,47	14,30	26,47	3,56
net_20_20T	15,10	0,00	3600	**0,00**	**10,17**	4,85	3,49	76,70	0,96
net_20_60F	594,74	0,00	3600	**0,00**	**78,59**	0,00	25,03	0,00	18,61
net_20_60T	9,10	0,00	3600	0,00	26,35	**0,00**	**5,88**	0,00	3,05
net_20_99F	1103,86	0,00	3600	**0,00**	**143,29**	0,00	53,85	0,00	35,96
net_20_99T	7,90	0,00	3600	0,00	36,71	0,00	8,89	**0,00**	**4,58**
net_25_60T	263,55	0,00	3600	0,00	851,61	**0,00**	**193,76**	0,00	119,63
AverageData	477,75	0	3600	0	134,64				

Table 4: Results for 10 instances of net type

heuristic for $\varepsilon \in \{1/2, 1, 3/2\}$ in terms of time and of relative error with respect to the optimum value computed by CPLEX (columns Gap). As with the SndLib case, for each row bold entries are used to show the first value of ε for which the cost scaling heuristic version becomes competitive with CPLEX. Interesting results are obtained by the cost scaling heuristic for these 10 instances. Only for one instance, namely *net_20_5T*, does the cost scaling version with $\varepsilon = 3/2$ fail to be more efficient than CPLEX in terms of time, by obtaining the optimum in 5.23 sec. opposed to the 4.04 sec. needed by the commercial solver. The increase in computational time is however very limited. It should be stressed that for these 10 instances the case with $\varepsilon = 1/2$ preserves the optimality of the solution and stops in an average time (see last row of Table 4) of 134.64 sec. as opposed to the 477.75 seconds needed on average by CPLEX.

We conclude this section by giving an idea of the effect of random coloring on the performance of our approach within the Belotti data set. Table 5 gives the performance of CPLEX, ExactCC and the random coloring variant for a set of 3 instances which are quite difficult to solve for CPLEX. As far as the random coloring is concerned, for all three instances we investigated a number of colorings α equal to 5, while the number of colors ν is 70% of the number of nodes n for the first two instances in the table and 80% of n for the last instance. It should be noted that, for the most critical instance among these three, namely *net_30_20F*, our exact approach returned a much better quality solution than the one given by CPLEX (solution value of 86 instead of 137) even though both solvers reached the time limit. The random coloring variant maintained the same good level of solution quality while reducing the time by about 5 times. For the first two instances of the *toronto* type, the effect of the random coloring consisted in further reducing the computational time with respect to ExactCC (which in turn greatly reduced the time with respect to CPLEX), while preserving the optimality of the computed solution.

		Cplex		ExactCC		RandomCol	
ProbName	Colors	Obj	Time	Obj	Time	Obj	Time
toronto_5F	70%	10	3600	10	80,87	10	17,37
toronto_25F	70%	10	1933	10	46,35	10	9,29
net_30_20F	80%	137	3600	86	3600	86	667,53

Table 5: The effect of random coloring

QoS properties of the generated balanced paths In conclusion, we would like to outline some interesting Quality of Service (QoS) properties of the generated balanced paths. One interesting question is whether the computed balanced paths are "sufficiently" short from an absolute point of view, i.e. with respect to the minimum longest path cost achievable by the sets of node-disjoint paths linking the given origins to the given destinations. This is particularly relevant in telecommunication applications, where minimizing the maximum hop cardinality and minimizing the longest path cost are much needed QoS requisites. Indeed, the disjoint balanced paths computed by the color-coding algorithms are quite "short" on average, both considering the maximum hop cardinality and the longest path cost. As far as the first QoS requisite is concerned, the average maximum hop cardinality of the computed balanced paths is very low. In fact, it is 3 both for the Sndlib and the Belotti data sets (the average number of network nodes is 24 for the Sndlib data set, and 23 for the Belotti data set). As far as the second QoS requisite is concerned, i.e. the longest path cost, for both data sets the paths returned by the color-coding algorithms are on average quite short. Specifically, the average percentage deviation from a suitably introduced lower bound is $\leq 20\%$ for the majority of the tested instances (36 over 58 in the first data set, and 64 over 104 in the second data set). Moreover, for 56 out of the overall 162 instances the minimum possible longest path cost was achieved.

References

[1] Alon, N., R. Yuster, and U. Zwick, "Color-coding," *Journal of the ACM* 42, pp. 844–856, 1995.

[2] Cappanera, P. and G. Gallo, "A multi-commodity flow approach to the crew rostering problem," *Operations Research* 52, pp. 583–596, 2004.

[3] P. Cappanera and M.G. Scutellà, "Balanced paths in acyclic networks: tractable cases and related approaches," *Networks* 26, pp. 230–245, 2005.

[4] P. Cappanera and M.G. Scutellà, "Computing balanced paths in telecommunication networks: color-coding approaches and related computational issues," Technical report, 2006.

[5] Lee, S.S. and M. Gerla, "Fault tolerance and load balancing in QoS provisioning with multiple MPLS paths," *Lecture Notes in Computer Science* LNCS 2092, pp. 155–169, 2001.

[6] Li, C-L., S.T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics* 26, pp. 105–115, 1990.