On Minimum Changeover Cost Arborescences

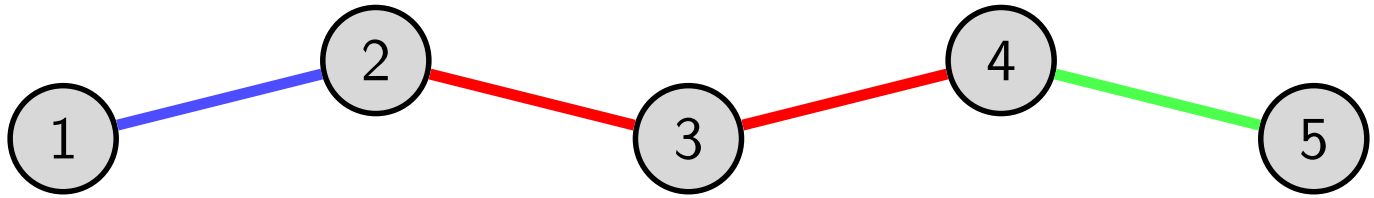
Giulia GALBIATI
University of Pavia

Stefano GUALANDI
University of Pavia

Francesco MAFFIOLI
Politecnico of Milano

SEA 2011
10th International Symposium on Experimental Algorithms
Kolimpari Chania, Greece, May 5-7, 2011

Concept of **reload cost** along a path (or along a cycle).



- It was introduced in the seminal paper:
  Wirth, H., and J. Steffan, *Reload cost problems: minimum diameter spanning tree*, Discrete Appl. Math. **113** (2001).

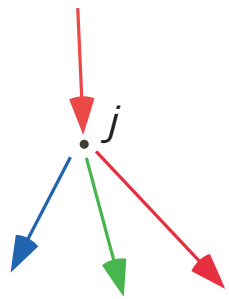
- Very natural concept that models a kind of cost incurred during a transportation/transmission activity.
  - transfer of energy among different carriers
  - loading/unloading of freight between different carriers
  - change of technologies (i.e. fiber, radio links, copper etc.) or of providers participating in a network

The list of **relevant references** is not long.

- Gamvros, I., L. Gouveia and S. Raghavan, *Reload cost trees and network design*. Proc. International Network Optimization Conference (INOC 2007), Spa, 2007.
- Galbiati, G., *The complexity of a minimum reload cost diameter problem*, Discrete Appl. Math., 156 (2008).
- Gourvès, L., A. Lyra, C. Martinhon and J. Monnot, *The minimum reload s-t path, trail and walk problems*, Discrete Appl. Math., 158 (2010).
- Amaldi, E., G. Galbiati and F. Maffioli, *On Minimum Reload Cost Paths, Tours and Flows*, Networks, 57 (2011).
- Galbiati G., Gualandi S. and Maffioli F., *On Minimum Reload Cost Cycle cover*, Internationa Symposium on Combinatorial Optimization (ISCO 2010), Hammamet, 2010.

In some of the problems in the references the objective functions depend on the amount of commodity flowing in the edges, so the reload costs are per unit of flow reload costs. This is not our case.

Concept of **changeover cost at node j** in an arborescence.



It is the sum of the costs to pay at node j for any outgoing arc.

- Very natural concept that comes from the need to model the fixed costs for installing in node j different devices to support the changes of carrier, i.e. of color.

Minimum Changeover Cost Arborescence
(MinCCA)

Given: - a digraph $G = (N, A)$, having n nodes and m arcs
- a function $c : A \to C$ that assigns a color to each arc, where $|C| = k$
- a matrix $D_{k,k}$, where d_{c_1,c_2} denotes the non-negative cost to pay at a node of an arborescence (different from the root), for any outgoing arc of color c_2 if the incoming arc has color c_1.

We assume that G contains at least one spanning arborescence having node 1 as root r.

Find: - an arborescence T, rooted at node 1, that minimizes the changeover cost $d(T)$ of T, i.e. the sum of the changeover cost $d(j)$ at j, over all nodes $j \neq 1$.

Our results

- MinCCA is NPO-complete.
- MinCCA is very hard to approximate.
- We present some Integer Programming formulations.
- We present a greedy approximation algorithm.
- We present an exact branch and cut algorithm.
- We report extensive computational results and exhibit a set of challenging instances.

Complexity

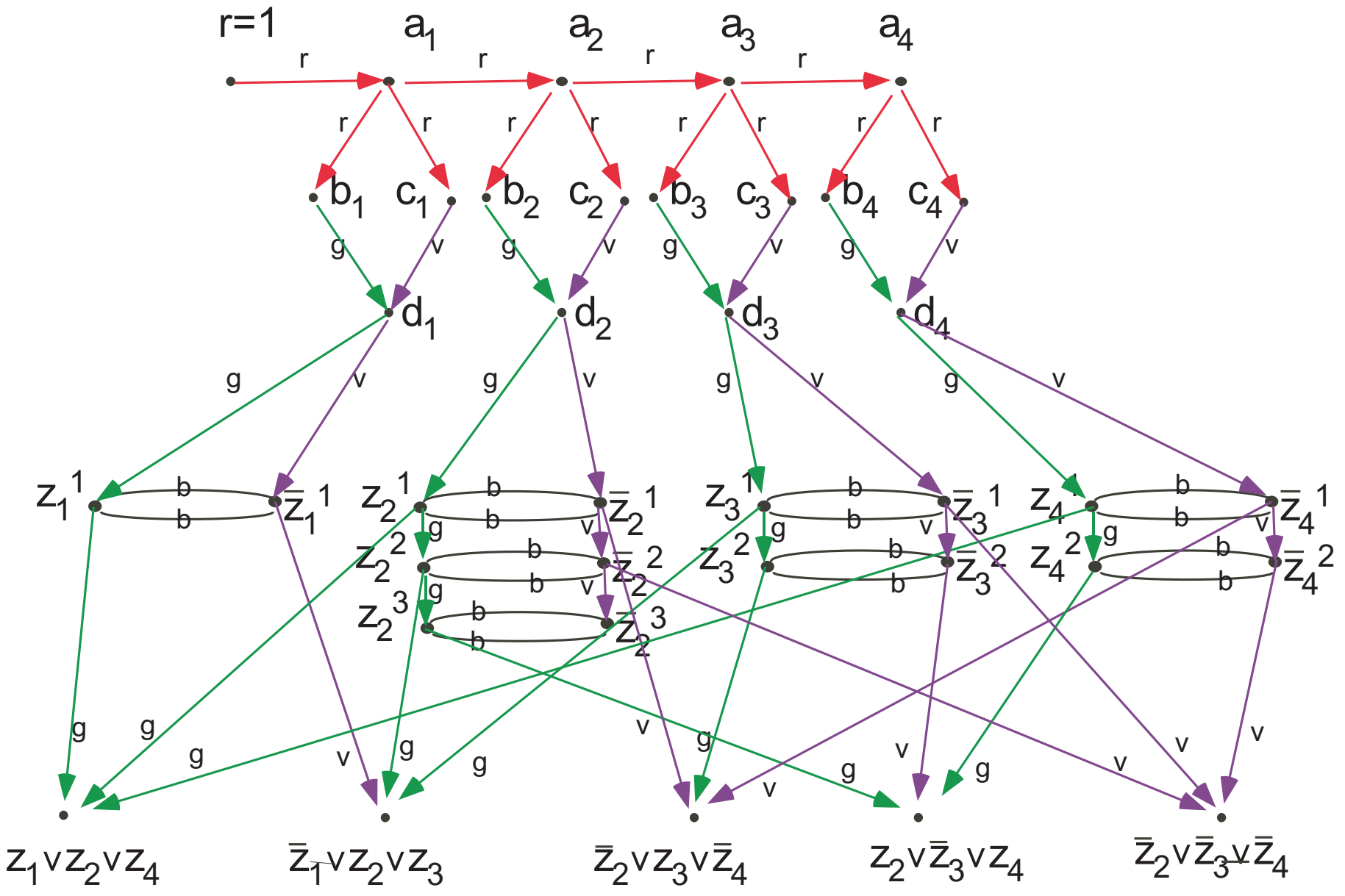
Theorem

Problem MinCCA *is NPO-complete. Moreover, there exists a real constant* α, $0 < \alpha < 1$, *such that the problem is not approximable within* $\alpha\sqrt[3]{n^{1-\varepsilon}}$, *for any* $\varepsilon > 0$, *even if formulated on graphs having bounded in and out degrees, costs satisfying the triangle inequality and at most 4 colors on the arcs.*

A reduction from the Minimum Ones problem:

Given: - a set $Z = \{z_1, ..., z_n\}$ of n boolean variables
- a collection $C = \{c_1, ..., c_m\}$ of m disjunctive clauses with at most 3 literals

Find: - a subset $Z' \subseteq Z$ having minimum cardinality and such that the truth assignment for Z that sets to true the variables in Z' satisfies all clauses in C.
If the collection of clauses is unsatisfiable the trivial solution returned is set Z.


r=1
a1
a2
a3
a4
b1
c1
b2
c2
b3
c3
b4
c4
d1
d2
d3
d4
z1 1
z̄1 1
z2 1
z̄2 1
z3 1
z̄3 1
z4 1
z̄4 1
z2 2
z̄2 2
z3 2
z̄3 2
z4 2
z̄4 2
z2 3
z̄2 3
z1∨z2∨z4
z̄1∨z2∨z3
z̄2∨z3∨z̄4
z2∨z̄3∨z4
z̄2∨z̄3∨z̄4

The matrix D of costs is defined so that:

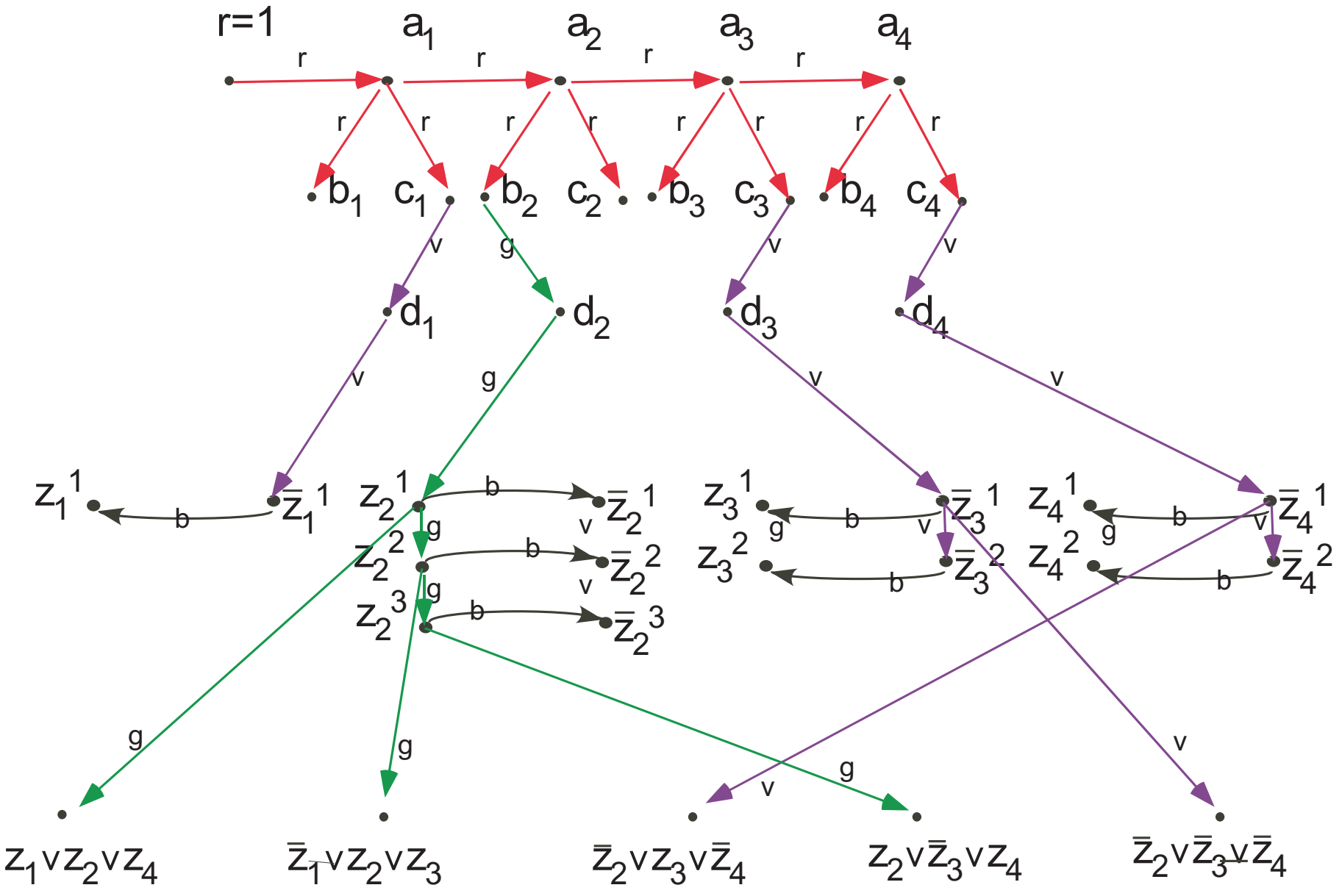
- $d(r, g) = 1$,
- $d(b, g) = d(b, v) = d(g, v) = d(v, g) = M$, with $M > n^2$
- all other costs equal to 0.

If n' denotes the number of nodes of the directed graph, we have that $n' \leq \beta n^3$, with $\beta > 1$ independent from n.

Proof. Based on:

- $opt(I) \leq opt(I')$ and equality holds iff I is a satisfiable instance of MINIMUM ONES.
- if there exists an $\bar{\varepsilon} > 0$ and MINCCA is approximable within $\alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}}$ then we could approximate MINIMUM ONES within $n^{1-\bar{\varepsilon}}$, contrary to the result in [Jonsson P. 1997].

Given instance I, the algorithm would construct instance I' and find a spanning arborescence T having a changeover cost $d(T)$ satisfying $\frac{d(T)}{opt(I')} \leq \alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}}$. If $d(T) \geq M$ the algorithm would return the trivial solution, otherwise, if $d(T) < M$, the algorithm would use T to return a solution Z' for I having $|Z'| = d(T)$.


r=1
a1
a2
a3
a4
r
b1
c1
b2
c2
b3
c3
b4
c4
v
g
d1
d2
d3
d4
b
z1 1
z̄1 1
z2 1
z̄2 1
z2 2
z̄2 2
z2 3
z̄2 3
z3 1
z̄3 1
z3 2
z̄3 2
z4 1
z̄4 1
z4 2
z̄4 2
z1∨z2∨z4
z̄1∨z2∨z3
z̄2∨z3∨z̄4
z2∨z̄3∨z4
z̄2∨z̄3∨z̄4

Combinatorial Lower and Upper Bounds

If an arc $e = (i, j)$ is part of the optimal solution, then the smallest possible contribution to the objective function given by this arc is:

$$w_e = w_{(i,j)} = \min_{f \in \delta^-(i)} d_{c(f),c(e)}$$

Computing the weight w_e of arc e takes time $O(d_i)$, with d_i in-degree of vertex i.

Once we have collected the values of w_e for every $e \in A$, we can solve the Minimum Weight Rooted Arborescence problem (with root $r = 1$), in time $O(m + n \log n)$ [Gabow1986].

The cost of the optimal solution T of this problem provides a lower bound for the optimal value of MinCCA, whereas the changeover cost $d(T)$ of arborescence T yields an upper bound.

Integer Programming formulation

Our problem and the Minimum Weight Rooted Arborescence problem have the same feasible region: The set of spanning r-arborescences.

The latter problem can be solved in polynomial time by classical methods (Edmonds 1967) or by Linear Programming.

Proposition (Schrijver 2003, pg. 897). Let $G = (N, A)$ be a digraph, r a vertex in N such that G contains a spanning arborescence rooted at r. Then the r-arborescence polytope of G is determined by:

$$\left\{ x \mid x \geq 0, \sum_{e \in \delta^+(X)} x_e \geq 1, \forall X \subset N, r \in X, \sum_{e \in \delta^-(j)} x_e = 1 \; \forall j \in N \setminus \{r\} \right\}$$

Hence we formulate MinCCA as an Integer Program using the same polyhedral description. However our objective function is quadratic and the continuous relaxation is no longer integral.

Integer Programming models

Bilinear Integer Programming formulation

$$\min \quad \sum_{j \in N \setminus \{r\}} \sum_{a \in \delta^-(j)} \sum_{b \in \delta^+(j)} d_{c(a),c(b)} x_a x_b \tag{1}$$

$$\text{s.t.} \quad \sum_{a \in \delta^+(X)} x_a \geq 1, \qquad \forall X \subset N, r \in X, \tag{2}$$

$$\sum_{a \in \delta^-(j)} x_a = 1, \qquad \forall j \in N \setminus \{r\}, \tag{3}$$

$$x_a \in \{0,1\}, \qquad \forall a \in A. \tag{4}$$

Linearized Integer Programming formulation

The objective function (1) can be linearized in a standard way by introducing binary variables z_{ab} instead of the bilinear term $x_a x_b$, and the linking constraints:

$$z_{ab} \geq x_a + x_b - 1, \quad z_{ab} \leq x_a, \quad z_{ab} \leq x_b. \tag{5}$$

A Stronger ? Linearization

In order to obtain stronger lower bounds we applied some of the techniques presented in [Caprara 2008] and [Buchheim 2010] to:

- the redundant constraint $\sum_{a \in A} x_a = n - 1$,
- the constraints $\sum_{a \in \delta^-(j)} x_a = 1, \quad \forall j \in N \setminus \{r\}$.

By multiplying both sides of the constraints by a binary variable x_b, we obtain, since $x_b^2 = x_b$, quadratic constraints whose linearized version, using the z_{ab} variables, become:

- $\sum_{a \in A, a \neq b} z_{ab} = (n-2)x_b, \quad \forall b \in A$,
- $\sum_{a \in \delta^-(j),\ a \neq b} z_{ab} = 0, \quad \forall j \in N \setminus \{r\}, \forall b \in \delta^-(j)$.

If the variables z_{ab} were already introduced to linearize (1) they can be reused. Otherwise new variables z_{ab} need to be introduced, along with their corresponding linking constraints given in (5).

Heuristic Algorithm

- Start with an arborescence T (not necessarly spanning) rooted in node 1, having arcs of a single color and maximizing the number of nodes.
- Repeat until T becomes a spanning arborescence:
  - for each color h find an arborescence T_h extending the arcs of T with arcs of color h, at minimum changeover cost d_h (using the minimum weight rooted arborescence algorithm)
  - replace T with the T_h that minimizes the ratio $\frac{d_h}{|T_h|-|T|}$.

Computational Results

- Collection of instances generated using the data in a collection of Steiner tree problems on graphs (http://steinlib.zib.de/). Used the data sets named B, I080, es10fst, and es20fst.
- All instances used for the experiments are available at http://www-dimat.unipv.it/ gualandi/Resources.html.

Since the library contains undirected graphs, we have introduced for each edge i,j two arcs (i,j) and (j,i). Each new arc gets a color randomly drawn from 1,...,k, where k ranges in the set $\{2,3,5,7\}$.

- uniform costs, i.e., all costs equal to 1 in any change of color
- non-uniform costs, randomly drawn from $\{1,..., 10\}$

Algorithms implemented in C++, using the g++ 4.3 compiler and the CPLEX12.2 callable library. Tests are run on a standard desktop with 2Gb of RAM and a i686 CPU at 1.8GHz.

Lower bounds on small instances

- LB_1 simple combinatorial lower bound
- LB_2 bound obtained solving the continuous relaxation (1)–(5)
- LB_3 bound obtained with the continuos relaxation (1)–(5) plus the constraints of the stronger linearization.

The separation problem for constraints (2) is solved using the Hao-Orlin algorithm in the LEMON graph library (whose time complexity is $O(n^2\sqrt{m})$.

Tabella: Comparison of lower bounds on small instances.

Instance	n	m	k	Opt	LB_1	LB_2	LB_3
ex-0	7	11	6	17	13	15.5	**17**
ex-1	7	12	6	2	**2**	1.5	**2**
ex-2	7	20	6	6	0	3	3.3
ex-3	7	10	6	11	9	10	**11**
ex-4	7	14	6	11	3	7.3	7.5
ex-5	7	12	6	15	11	10	13
ex-6	7	12	6	12	7	10	11.7
ex-7	7	13	6	11	10	9.5	**11**
ex-8	7	15	6	3	**3**	2	2.5
ex-9	7	16	6	7	3	3.5	4

Lower bounds on medium size instances

Tabella: Instances with uniform changeover costs and 5 colors.

Istance	n	m	Opt	UG	Gap	UB_1	Gap	LB_1	Gap	LB_2	T_{LB_2}	LB_3	T_{LB_3}	Gap
es10fst10	18	42	12	14	0.17	14	0.17	8	0.33	3.3	0.3	3.3	1.0	0.72
es10fst11	14	26	11	11	0.00	11	0.00	11	0.00	5.0	0.1	5.0	0.1	0.55
es10fst12	13	24	8	8	0.00	8	0.00	6	0.25	3.0	0.1	3.0	0.1	0.63
es10fst13	18	42	9	11	0.22	9	0.00	6	0.33	1.8	0.3	1.8	0.6	0.80
es10fst14	24	64	11	15	0.36	13	0.18	10	0.09	3.7	1.2	3.7	4.3	0.67
es10fst15	16	36	9	9	0.00	11	0.22	7	0.22	5.0	0.2	5.0	0.4	0.44
es20fst10	49	134	28	33	0.18	48	0.45	19	0.32	3.7	19.8	3.7	96.6	0.87
es20fst11	33	72	28	28	0.00	32	0.14	26	0.07	8.6	2.6	8.6	7.1	0.69
es20fst12	33	72	22	25	0.14	32	0.28	18	0.18	6.3	2.7	6.3	4.9	0.71
es20fst13	35	80	20	22	0.10	34	0.55	16	0.20	6.0	2.9	6.0	9.7	0.70
es20fst14	36	88	18	22	0.22	35	0.59	13	0.28	2.5	4.5	2.5	25.4	0.86
es20fst15	37	86	22	22	0.00	36	0.64	16	0.27	6.8	5.2	6.8	17.0	0.69
Averages:					0.12		0.27		0.21		3.3		13.9	0.69

Branch-and-cut

With the help of the CPLEX callable library, we implemented two versions of a standard Branch-and-Cut algorithm using the two linearizations proposed.

The first algorithm uses LB_2, the second LB_3. Both algorithms also embed the combinatorial bounds LB_1, and UB_1, UG.

In the variable selection strategy we favor variables associated to arcs whose head has a small degree (each variable x_a has a priority equal to $|n - degree(head(a))|$)

Results of the Branch-and-cut Algorithms

- For the es20fst data set.

  The table shows easy instances that both linearizations were able to solve within a time limit of 1000 sec.

- For the B data set and the I080 data set.
  These instances are easy for Steiner tree problems, but are demanding for the MinCCA problem. Both linearizations were not able to solve them within a time limit of 1000 sec.

  The table shows, in column Best, the best upper bound we were able to compute with our branch-and-cut algorithm and a timeout of 1000 seconds. We propose them as

Challenging Instances of the MinCCA problem

.

Results for the es20fst data set

Tabella: Results of the branch-and-cut algorithm: Easy instances.

Istance	n	m	Opt	Cuts	Nodes	Time	Cuts	Nodes	Time
es20fst11	33	72	122	188	723	12	147	667	35
			42	115	277	7	80	348	15
			85	374	1252	19	435	1048	55
			96	103	741	10	113	790	37
es20fst12	33	72	55	86	191	6	112	227	15
			78	63	327	7	62	354	34
			95	105	414	8	122	415	27
			86	57	772	9	55	708	39
es20fst13	35	80	102	58	280	8	60	732	63
			92	42	244	7	48	392	39
			62	112	843	16	118	920	64
			107	42	665	12	52	597	74
es20fst14	36	88	78	94	1395	35	104	1464	252
			40	86	850	35	88	692	186
			54	98	1376	41	92	737	119
es20fst15	37	86	110	242	2233	45	330	2073	174
			55	549	3344	71	592	5246	509
Averages:				142	937	20	154	1024	102

The first three columns Cuts / Nodes / Times refer to LB_2, the other three to LB_3.

Results for the B data set and the I080 data set

Tabella: Challenging instances with uniform costs.

Instance	n	m	Best	UG	UB_1	LB_1	Gap	LB_2	Gap	T_{LB_2}
b01	50	126	28	35	34	18	0.36	11.7	0.58	21
b02	50	126	30	31	34	20	0.33	8.8	0.71	15
b03	50	126	28	35	36	15	0.46	5.3	0.81	19
b04	50	200	24	26	35	6	0.75	1.6	0.93	66
b05	50	200	18	21	31	5	0.72	3.1	0.83	68
b06	50	200	20	23	31	4	0.80	1.0	0.95	65
b07	75	188	53	53	54	31	0.42	13.4	0.75	83
b08	75	188	51	55	56	31	0.39	13.8	0.73	59
b09	75	188	47	50	47	29	0.38	13.5	0.71	75
b10	75	300	36	36	46	8	0.78	3.5	0.90	212
b11	75	300	39	39	52	8	0.79	1.3	0.97	177
b12	75	300	35	35	44	5	0.86	1.2	0.97	179
b13	100	250	62	65	62	31	0.50	22.1	0.64	168
b14	100	250	59	59	70	33	0.44	18.7	0.68	150
b15	100	250	68	68	71	36	0.47	19.2	0.72	154
b16	100	400	42	42	63	6	0.86	0.3	0.99	1143
b17	100	400	46	46	63	7	0.85	1.2	0.97	350
b18	100	400	54	54	66	7	0.87	2.3	0.96	436
I080-001	80	240	47	47	59	28	0.40	15.2	0.68	159
I080-002	80	240	49	49	58	20	0.59	8.0	0.84	173
I080-003	80	240	43	43	46	24	0.44	7.3	0.83	131
I080-004	80	240	48	48	51	22	0.54	10.1	0.79	139
I080-005	80	240	41	41	56	13	0.68	6.0	0.85	228
Averages:							0.60		0.82	

Here Best denotes the best upper bound we were able to compute with a timeout of 1000 seconds and T_{LB_2} denotes the time spent at the root node to compute LB_2.

Thank you for your attention