Thinning out Steiner Trees

Matteo Fischetti
Markus Leitner
Ivana Ljubic
Martin Luipersbeck
Michele Monaci
Max Resch
Domenico Salvagnin
Markus Sinnl

University of Padova and Vienna

- **The 11th DIMACS challenge was on Steiner Tree Problems**

Problems Addressed

The Challenge addresses the Steiner Tree problem and its variants, including

1. classical Steiner problem in graphs;
2. classical geometric Steiner problem (including rectilinear, Euclidean
3. prize-collecting Steiner trees;
4. dynamic Steiner trees;
5. node-weighted Steiner trees;
6. generalized Steiner trees (Steiner forest);
7. obstacle-avoiding Steiner trees;
8. directed Steiner trees (arborescences);
9. rooted Steiner trees with height (or hop) constraints;
10. ... trees;

- Mean Value
  - geometric mean for primal bound
  - shifted geom. mean (shift 1.0) for primal integral and time
  - arithmetic mean for gap
- Points-based method

- Quality Challenge
  - Primal Bound upon termination
- Pareto Challenge
  - Primal Integral
- Gap-based Exact Challenge
  - primal-dual gap upon termination: $\frac{|pb-db|}{abs(|pb|,|db|)} \in [0,1]$
- Time-based Exact Challenge
  - time to solve (or time limit)

- "old" Formula 1 point scheme (1991 – 2002)
- Winner gets 10 points, second 6, then 4,3,2,1

Vienna and Padua together

- A bunch of effective codes initially provided by Vienna's team
- All codes re-engineered and re-tuned to improve performance
- **New exact and heuristic codes implemented (this talk)**
- Initial filter to select the actual code to be run based on the instance type

- **Four codes** (all based on Cplex 12.6) finally submitted to DIMACS's challenge

#MozartBalls	exact, single & multi-threaded	SPG, (R)PCSPG, MWCS, DCST
#StayNerd	heuristic, single & multi-threaded	SPG, PCSPG
#MozartDuet	heuristic, multi-threaded	SPG, PCSPG
#HedgeKiller	heuristic, multi-threaded	SPG, PCSPG

DIMACS 11: Competition Results

Gerald Gamrath

Zuse Institute Berlin · Berlin Mathematical School

Challenge results

Detailed final results posted by Gerald Gamrath at

http://dimacs11.cs.princeton.edu/contest/challenge-results.pdf

Many variants / scores → not a single winner…

… however #MozartBalls ranked first in many categories

Steiner Tree Problem

Definition (The prize-collecting Steiner tree problem (PCSTP))

Given an undirected graph $G = (V, E)$ with a (possibly empty) set of *real terminals* $T_r \subset V$, edge costs $c : E \mapsto \mathbb{R}^+$ and node revenues $p : V \mapsto \mathbb{R}^+$, the PCSTP is to find a subtree $\mathcal{T}$ that spans all real terminals and such that the cost

$$c(\mathcal{T}) = \sum_{e \in E[\mathcal{T}]} c_e + \sum_{i \notin V[\mathcal{T}]} p_i$$

is minimized.

Definition (Potential terminals)

Among the nodes $i \in V \setminus T_r$, only those with revenues $p_i > 0$ such that at least one adjacent edge is strictly cheaper than p_i are considered as potential leaves. These nodes are referred to as *potential terminals*, and the associated set is denoted by T_p:

$$T_p = \{v \in V \setminus T_r \mid \exists \{u, v\} \text{ s.t. } c_{uv} < p_v\}.$$

Hypercube instances

- Since the beginning of our study we focused on the (in)famous **hypercube** instances introduced by

> Isabel Rosseti, Marcus Poggi de Aragao, Celso C. Ribeiro, Eduardo Uchoa, and Renato F. Werneck. New benchmark instances for the Steiner problem in graphs. In Extended Abstracts of the 4th Metaheuristics International Conference (MIC'2001), pages 557–561, Porto, 2001.

2.1 Hypercubes (hc)

Graphs in this series are d-dimensional hypercubes, with $d \in \{6, \ldots, 12\}$. For each value of d, the corresponding graph has 2^d nodes and $d \cdot 2^{d-1}$ edges. These graphs are bipartite, and both partition subsets have the same cardinality. The vertices in one of such subsets become terminals ($|X| = 2^{d-1}$).

- Real terminals only
- All edges have cost 1
- Very symmetrical

  very difficult LPs

  very hard even for heuristics

Name	|V|	|E|	|T|	DC	Opt
hc6u	64	192	32	NPm	**39**
hc7u	128	448	64	NP?	**77**
hc8u	256	1024	128	NP?	**148**
hc9u	512	2304	256	NP?	*292*
hc10u	1024	5120	512	??	*582*
hc11u	2048	11264	1024	??	*1154*
hc12u	4096	24576	2048	??	*2275*

Looking for more effective models

- Many hard instances (including hc*u) involve **uniform edge costs**
- For these instances, edge costs can easily be moved to nodes …

  … hence edge variables become **redundant** (actually, **harmful** as they add a lot of symmetry and overload the model with useless cuts)
- We better work on the space of node-variables only …

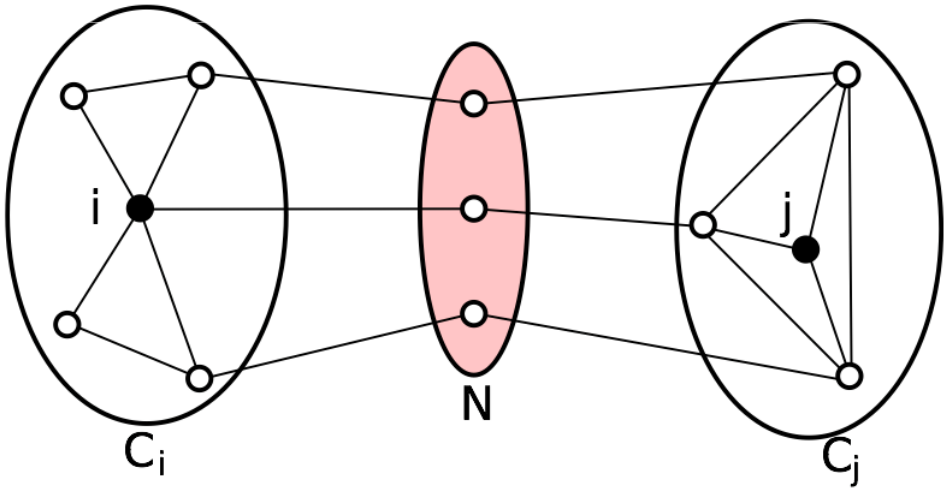
  … and only impose **connectivity of the subgraph** induced by the selected nodes, as in

  Eduardo Alvarez-Miranda, Ivana Ljubic, and Petra Mutzel. **The maximum weight connected subgraph problem**. In *Facets of Combinatorial Optimization: Festschrift for Martin Groetschel*, 245–270, Springer, 2013.

Node separators

Definition (Node Separators)

For $i, j \in V$, a subset $N \subseteq V \setminus \{i, j\}$ is called (i, j) **node separator** iff after eliminating N from V there is no (i, j) path in G.
N is minimal if $N \setminus \{i\}$ is not a (i, j) separator, for any $i \in N$.
Let $\mathcal{N}(i, j)$ denote the family of all (i, j) separators.



Model for uniform edge costs

Node-based MIP model

Shift uniform edge costs c into node revenue:

$$\tilde{c}_v = c - p_v, \qquad \forall v \in V$$

Let

$$T = T_r \cup T_p \qquad\qquad P = \sum_{v \in V} p_v$$

$$\begin{aligned}
\min \quad & \sum_{v \in V} \tilde{c}_v y_v + (P - c) && && (1)\\
\text{s.t.} \quad & y(N) \geq y_i + y_j - 1 && \forall i,j \in T, i \neq j, \forall N \in \mathcal{N}(i,j) && (2)\\
& y_v = 1 && \forall v \in T_r && (3)\\
& y_v \in \{0,1\} && \forall v \in V \setminus T_r && (4)
\end{aligned}$$

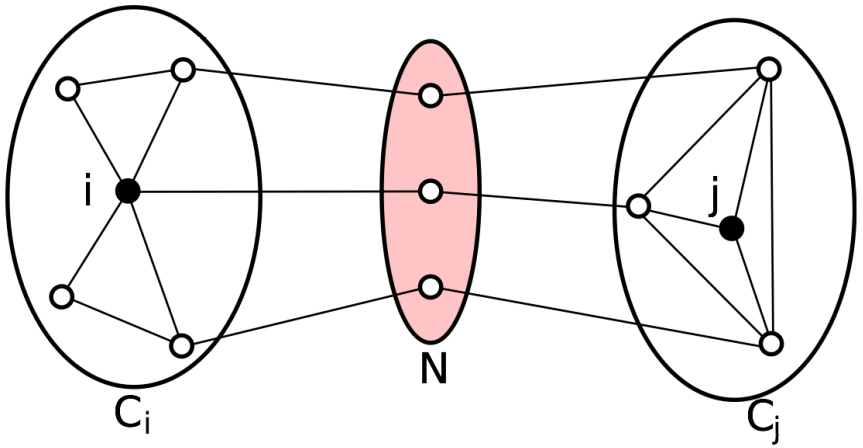
where $y(N) = \sum_{v \in N} y_v$.

Connectivity cut separation

Data: Infeasible solution defined by a vector $\tilde{y} \in \{0,1\}^n$ with $\tilde{y}_i = \tilde{y}_j = 1$, C_i being the connected components of $G_{\tilde{y}}$ containing i, and $j \notin C_i$.

Result: A minimal node separator N that violates inequality (2) with respect to i, j.

Delete all edges in $E[C_i \cup A(C_i)]$ from G
Find the set R_j of nodes that can be reached from j
Return $N = A(C_i) \cap R_j$



Basic model

- Node-degree inequalities:

$$y(A_i) \geq \begin{cases} y_i, & \text{if } i \in T \\ 2y_i, & \text{otherwise} \end{cases} \tag{5}$$

Lifted version for potential terminals:

$$2 \sum_{v \in A_i : c_{vi} < p_i} y_v + \sum_{v \in A_i : c_{vi} \geq p_i} y_v \geq 2y_i \quad \forall i \in T_p.$$

- 2-Cycle inequalities:

$$y_i \leq y_j \quad i \in V, j \in T_p, c_{ij} < p_j \tag{6}$$

Branch & Cut code

- Built on top of IBM ILOG **Cplex 12.6**
- Initial compact relaxation → **basic model** (quite tight for some instances)
- Very basic implementation with connectivity cuts **separated for integer points** only (lazycut callback)
- Cplex's **cuts** at default level (but 0-1/2 cuts that are set to *aggressive*)
- Cplex's **primal heuristics** at default level (local-branching set to on)
- More elaborated implementations tried → **worse performance**
- **Local branching heuristic using the B&C itself as a black-box solver**

$$\sum_{j \in V : \tilde{y}_j = 1} y_j \geq \left| \{ j \in V : \tilde{y}_j = 1 \} \right| - r \qquad (7)$$

  - invoked a few times before attempting the exact solution
  - used to gather **primal information** (better and better feasible solutions) as well as **dual information** (connectivity cuts) → **restart policy**

Exact solver: basic framework

Data: Input graph G, instance of the STP/PCSTP/DCST/MWCS, iteration and time limits.
Result: A (sub)-optimal solution *Sol*.
$\mathcal{S}_{\text{init}}$ =**InitializationHeuristics**()
$k = 1$, $CutPool = \emptyset$, $\mathcal{S}_{\text{lb}} = \emptyset$
Choose *Sol* from the solution pool $\mathcal{S}_{\text{init}}$.
while *(k ≤ maxLBiter) and (time limit not exceeded)* **do**
  $(Sol, CutPool, \mathcal{S}_{\text{lb}})$= **LocalBranching**$(Sol, CutPool, \mathcal{S}_{\text{lb}}, seed)$
  Choose *Sol* from the solution pool $\mathcal{S}_{\text{init}}$. Change *seed*.
  if $k \bmod 10 == 0$ **then**
    Sol_{recomb} = **Recombine**$(\mathcal{S}_{\text{lb}})$
    if $cost(Sol_{recomb}) < cost(Sol)$ **then**
      $Sol = Sol_{recomb}$
    end
  end
  $k = k + 1$
end
Sol = **BranchAndCut**$(CutPool, Sol, TimeLim)$
return *Sol*

Comparing exact models

Table 4. Uniform STP instances (GAPS and SP datasets). Proven optimal solutions in boldface. Previous best known solutions given in brackets. Column Time gives the computing time for proving the optimality (or, the time limit, otherwise). Columns UB and LB show upper and lower bounds obtained by the (x, y)-model, within the time limit of two hours, respectively.

				y-model		(x, y)-model			
name	$\|V\|$	$\|E\|$	$\|T\|$	OPT	Time (s.)	UB	LB	gap	Time (s.)
s1	64	192	32	**10**	0.03	10	10	0.0%	0.01
s2	106	399	50	**73**	0.10	73	73	0.0%	1.67
s3	743	2947	344	**514**	0.29	514	508	1.19%	7200.00
s4	5202	20783	2402	**3601**	1.72	3601	3515	2.39%	7200.00
s5	36415	145635	16808	**25210**	30.06	25210	24448	3.02%	7200.00
w13c29	783	2262	406	**507** (508)	0.61	507	507	0.00%	167.17
w23c23	1081	3174	552	**689** (694)	196.19	689	689	0.00%	1114.00

Benders-like heuristic

- Steps
  1. Extract a relaxation of current model
  2. Solve relaxation heuristically
  3. Repair solution through local branching centered at the (infeasible) solution found
  4. Add the connectivity cuts generated by local branching to the relaxation
  5. Repeat

Benders-like heuristic

Data: Time/iteration limits.
Result: Feasible solution Sol, cut pool $CutPool$.
Sol = dummy solution of very large cost
$CutPool = \emptyset$
while *(time/iteration limit not exceeded)* **do**
  Heuristically solve a relaxation of the current model (including all cuts in $CutPool$) and let Sol^R be the possible disconnected solution found
  Add the local branching constraint (7) centered on Sol^R to the unrelaxed model
  $(Sol', CutPool') =$ `BranchAndCut`$(CutPool, Sol, TimeLim, SolLim)$
  Remove the local branching constraint from the unrelaxed model
  if $cost(Sol') < cost(Sol)$ **then**
    $Sol = Sol'$
  end
  $CutPool = CutPool \cup CutPool'$
end
return Sol

Set covering is everywhere!

- For pure instances (real terminals only) the node model can be interpreted as a huge **set covering problem**

$$y(N) \geq y_i + y_j - 1 \text{ with } i, j \in T_r \Rightarrow y(N) \geq 1$$

- Given the basic model + pool of connectivity cuts, the **relaxation** in the Benders-like framework can then be solved by a **specialized set-covering heuristic** → we used the CFT heuristic of

  Alberto Caprara, Matteo Fischetti, and Paolo Toth. A heuristic method for the set covering problem. Operations Research, 47(5):730–743, 1999.

- For non-uniform instances, **blurred version** → in the “relaxation”, edge costs are heuristically split among adjacent nodes (so the node model applies)

- **Very fast and effective, in particular, for hc*u instances**

Set-covering heuristic results

Table 3. Our very preliminary results for unsolved uniform STP instances of the PUC class

name	PreviousBest	NewBest	TURNI time (s.)	Repair time (s.)
bip52u	234	234	60.0	2.14
bip62u	220	**219**	60.0	0.03
bipa2u	341	**338**	60.0	0.05
hc9u	292	292	60.00	1.53
hc10u	581	**575**	60.00	4.52
hc11u	1154	**1145**	1800.00	0.07
hc12u	2275	**2267**	60.00	6.68

problem	instance	best UB	Time	#threads
STP	hc11u	**1144**	474	8
STP	hc12u	**2256**	4817	8
STP	hc12p	**236158**	4411	4
PCSTP	hc11u2	**751**	298	8
PCSTP	hc12u2	**1492**	632	16

Thanks for your attention

- Full paper

  M. Fischetti, M. Leitner, I. Ljubic, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl, **Thinning out Steiner trees: a node-based model for uniform edge costs**, Technical Report DEI, University of Padua, 2014.

  and slides available at

  http://www.dei.unipd.it/~fisch/papers/

  http://www.dei.unipd.it/~fisch/papers/slides/