On the Steiner k-edge-connected network design problem

Ibrahima Diarrassouba[1], Mohamed Khalil Labidi[2,3], Ali Ridha Mahjoub[3]

[1] Le Havre University, LMAH, FR-CNRS-3335, Le Havre, France
{ibrahima.diarrassouba}@univ-lehavre.fr
[2] Faculty of Science of Tunis, URAPOP, UR13ZS38, Tunis, Tunisia
{mohamed-khalil.labidi}@dauphine.fr
[3] Paris Dauphine University, PSL Research University, CNRS UMR 7243, LAMSADE, 75016 Paris, France
{ridha.mahjoub}@dauphine.fr

Keywords : *Lagrangian relaxation, large scale, genetic algorithm, hybridization, parallel computing, SkESNDP.*

1 Introduction

Let $G = (V, E)$ be an undirected graph, a subset of nodes $S \subseteq V$ called *terminals*, a weight function $\omega : E \to \mathbb{R}$ which associates the weight ω_e with each edge $e \in E$. The *Steiner k-Edge-Connected Network Design Problem* (SkESNDP for short) is the problem of finding a minimum weight subgraph of G spanning S such that between every two nodes $u, v \in S$, there are at least k-edge-disjoint paths.

The SkESNDP is a special case of a more general model, introduced by [6] and later called *generalized Steiner problem.* This problem is well known to be NP-hard and has been largely investigated in the literature. For an exhaustive description of Steiner survivability problems variants we refer the reader to [2], and for more complete surveys to [1, 3, 5].

2 Integer programming formulation

The SkESNDP can be formulated using a polynomial number of variables and constraints using $|S|(|S| - 1)/2$ minimum cost st-flow problems on a directed graph obtained by exchanging each edge $uv \in E$ by two arcs (u, v) and (v, u). This formulation is called *Undirected Flow Formulation* (see [4]). The SkESNDP is equivalent to

$$\min \sum_{uv \in E} \omega_{uv} x_{uv}$$

$$\sum_{v \in V\setminus\{u\}} f^{st}_{uv} - \sum_{l \in V\setminus\{u\}} f^{st}_{lu} = \left\{ \begin{array}{rll} k, & if & u = s, \\ -k, & if & u = t, \\ 0, & if & u \in V \setminus \{s,t\}, \end{array} \right\} \text{for all } u \in V \text{ and } \{s,t\} \in D, \qquad (1)$$

$$\left. \begin{array}{l} f^{st}_{uv} \\ f^{st}_{vu} \end{array} \right\} \leq x_{uv}, \qquad \text{for all } uv \in E \text{ and } \{s,t\} \in D, \qquad (2)$$

$$f^{st}_{uv}, f^{st}_{vu} \geq 0, \qquad \text{for all } uv \in E \text{ and } \{s,t\} \in D, \qquad (3)$$

$$x_{uv} \leq 1, \qquad \text{for all } uv \in E, \qquad (4)$$

$$x_{uv} \in \{0,1\}, \qquad \text{for all } uv \in E, \qquad (5)$$

$$f^{st}_{uv} \in \{0,1\}, \qquad \text{for all } uv \in E, \{s,t\} \in D. \qquad (6)$$

3 Parallel hybrid optimization algorithm

Our algorithm relies on the usage of parallel computing for solving the SkESNDP and taking advantage from the diagonal structure of the formulation presented above. Namely, we devise an optimization approach for the problem based on three algorithms

- *a greedy heuristic (SH)*;
- *a Lagrangian relaxation algorithm (RLA)*;
- *a genetic algorithm (GA).*

For our purpose, we run these three algorithms in a parallel computing framework. Also, each iteration of each algorithm is used to improve the other algorithms, and hence, improve the whole algorithm. Moreover, we solve the Lagrangian relaxation, and the genetic algorithms using parallel computing.

Instances			RLA				SH		PHA				CPLEX			
name	$\|V\|$	$\|S\|$	*UB*	*LB*	*Gap*	*CPU*	*UB*	*CPU*	*UB*	*LB*	*Gap*	*CPU*	*UB*	*LB*	*Gap*	*CPU*
berlin	30	3	2539	2409.13	5.12	00:00:03	2800	00:00:00	2503	2285.98	8.67	00:00:00	2489	2489	0	00:00:00
	52	7	20400	3127.03	84.67	00:00:37	4761	00:00:00	4187	3006.06	28.2	00:03:19	3802	3389.61	10.85	02:00:00
st	70	9	2920	201.558	93.1	00:01:13	552	00:00:00	507	222.73	56.07	00:07:10	688	297.11	56.82	02:00:00
kroA	100	9	151739	5358.9	96.47	00:04:12	21614	00:00:00	19846	5270.00	73.45	00:24:35	49051	7821.06	84.06	02:00:00
	200	9	194591	6294.12	96.77	00:17:02	22947	00:00:00	20403	6088.26	70.16	01:33:39	117741	5458.25	95.36	02:00:00
	200	13	375741	9256.4	97.54	00:35:33	29989	00:00:01	26397	8525.46	67.7	02:00:00	294825	7890.25	97.32	02:00:00
lin	318	13	56637	3553.51	93.73	01:00:24	8051	00:00:05	7660	2656.36	65.32	02:00:00	93116900	0	100	02:00:00
	318	15	75722	3369.12	95.55	01:37:53	9331	00:00:07	9017	2749.56	69.51	02:00:00	–	–	–	–

TAB. 1: Numerical results for the algorithms RLA, SH, PHA and CPLEX for $k = 3$

We can notice that the approach is able to improve, for all the instances, the upper bounds given by SH and RLA. Comparing PHA to CPLEX we can see that our algorithm produces better upper bounds for the large scale graphs while CPLEX is able to solve to optimality the small ones. We can also see that for 4 instances, CPLEX produces a better lower bound than PHA. Also, the gap produced by PHA is better than that produced by CPLEX for 6 instances.

Finally, we can see that the CPU time of PHA is relatively small, while CPLEX reaches the maximum CPU time for almost all the instances (6 instances over 8). Moreover, PHA has been able to produce an upper bound for instance lin318-15 while CPLEX was not able to produce even a feasible solution due to lack of memory.

References

[1] Ding-Zhu Du, JM Smith, and J Hyam Rubinstein. *Advances in Steiner trees*, volume 6. Springer Science & Business Media, 2013.

[2] Mathias Hauptmann and Marek Karpiński. *A compendium on steiner tree problems.* Inst. für Informatik, 2013.

[3] Hervé Kerivin and A Ridha Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[4] Thomas L Magnanti and S Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.

[5] Hans Jürgen Prömel and Angelika Steger. *The Steiner tree problem: a tour through graphs, algorithms, and complexity.* Springer Science & Business Media, 2012.

[6] Kenneth Steiglitz, Peter Weiner, and D Kleitman. The design of minimum-cost survivable networks. *IEEE Transactions on Circuit Theory*, 16(4):455–460, 1969.