DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Comparison of Two Loss-Contracting Algorithms for Network Procurement

Philipp Michael Sauter

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Comparison of Two Loss-Contracting Algorithms for Network Procurement

Vergleich zweier Loss-Contracting Algorithmen für Network Procurement

Author:	Philipp Michael Sauter
Supervisor:	Prof. Martin Bichler
Advisor:	Richard Littmann
Submission Date:	15.08.2020

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.08.2020 Philipp Michael Sauter

Abstract

In economics there is an interest to find incentive compatible and efficiently computable auction mechanisms that give good approximations for optimal social welfare. This thesis will try to find such an approximation by using algorithms for the Steiner Tree Problem on Procurement Auctions.

The environment for the Steiner Tree Problem is a weighted, undirected graph G = (V, E), where V is the set of nodes and E is the set of edges within G. There is also a subset T $\subseteq$ V called "terminals" and a cost function c: E $\rightarrow \mathbb{R}$, that returns the added path cost for a given set of edges. The Steiner Problem now asks for the minimal tree connecting all terminals. This problem is NP-complete and the algorithms presented in this paper give approximations for the solution. The first approximation algorithm we're going to look at was presented in a paper by Berman and Ramaiyer [1] and constructs Steiner minimal trees for subsets of T with at most k elements, adding them to the solution greedily and finally removing the redundant edges. It was proven, that increasing the value of parameter k improves the approximation rate, which converges to 1.746 for k $\rightarrow \infty$. The second algorithm from Hougardy and Prömel [3] improves the previous approximation rates by using a generalized version of the parameterized relative greedy heuristic (RGH) by Karpinski and Zelikovsky [6] and iteratively applying it with different parameters α_i (i $\in$ [1, k]) to the previous iteration's output. They achieved the approximation ratio of 1.598 after 11 iterations and indicated the limit at 1.588 for k $\rightarrow \infty$. In a 2007 paper, Blumrosen and Nisan [2] proved that a mechanism for single-minded bidders is incentive compatible iff it satisfies monotonicity and critical payment. We'll therefore assess monotonicity for both implementations and afterwards compute critical payments. To find the critical payment for Edge e we'll change the cost of e in the input graph and check whether or not it is still included in the resulting Steiner Tree after applying the algorithm on this changed input graph. Using binary search we're going to find the maximum prize at which e is still part of the tree, which is therefore the critical payment. These critical payments should vary for both algorithms as well as the total edge cost and the runtime. To conclude this thesis we will analyse these statistics and compare them between the two algorithms.

Contents

1 Introduction

Citation test [10].
See Table 1.1, Figure 1.1, Figure 1.2, Figure 1.3.

Table 1.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

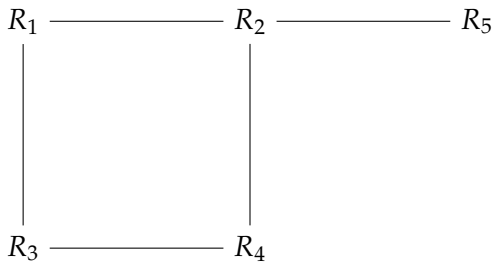

Figure 1.1: An example for a simple drawing.

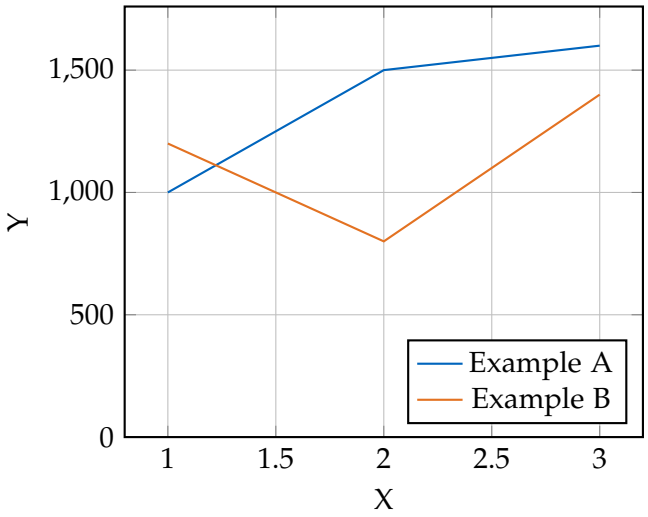

Figure 1.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 1.3: An example for a source code listing.

2 Problem

2.1 Setting

In the following we are going to model a Network Auction, where the participants bid on edges within the network. The single-minded bidders are supposed to form a network containing a certain set of nodes. In order to achieve this they can submit bids on any connection between two nodes. The true valuations of these connections are known only by the provider of it. The goal is to minimize the cost, by buying the cheapest set of edges, that spans the required nodes. This is in essence the Steiner Tree Problem in Graphs, which is NP-Complete and has multiple greedy Approximation Algorithms presented for it. In order to find out a decent approximation, that suits our additional needs for incentive compatibility and welfare optimization, we are going to implement two advanced Approximation Algorithms and try and prove that they are incentive compatible by using the Lemma 1.9 by Blumrosen and Nisan [2]. In order to meet the requirements we will have to prove Monotonicity, as well as finding critical payments for every edge included in the approximated solution.

2.2 Steiner Tree Problem in Graphs

The Steiner Problem, which was named after Jakob Steiner, has application in a lot of settings, but the most common setting is the Steiner Tree Problem in Graphs. It takes an connected, undirected, weighted Graph with non-negative edge-weights, as well as a set of vertices called "terminals" or "terminal points" and askes for the minimum weights tree connecting all terminals, called a Steiner Minimum Tree (SMT). Vertices that aren't terminals are called Steiner points and they can be included in a Steiner Tree, but their inclusion is optional as opposed to terminal points. The Steiner Tree Problem in Graphs is one of Karp's 21 NP-complete problems [5] and we will therefore only look at Approximation Algorithms for it. The input Graph for the Steiner Tree Problem is assumed to be complete and the cost of each edge between two vertices u and v is assumed to be the the length of the shortest path between u and v. This assumption is without loss of generality since if a connected Graph doesn't fulfill these criteria, we can just use the metric closure of it.

2.3 Definitions

The Environment for the Steiner Tree Problem in Graphs is a weighted undirected Graph G = (V, E), where V is the set of vertices in the Graph and E the set of edges. The set T $\subseteq$ V marks the terminal vertices of G and V\T is the set of Steiner points. The length of a given set of edges E or a given Tree X d(E)/d(X) is the sum of all edge costs. For any Set of vertices S any tree connecting S is called a Steiner Tree for S. The minimum spanning tree of S is denoted by MST(S) and its length by mst(S). Similarly a Steiner minimal tree for S is denoted by SMT(S) and its length by smt(S). A Steiner Tree is called full if all terminals are leaves in the tree and a k-restricted Steiner Tree is a full Steiner Tree that has at most k terminals. The loss of a Steiner Tree X l(X) is the length of a minimum forest spanning all vertices of X, where every component of the forest contains at least one terminal node. The contraction of a set of vertices S means to set the length of edges between vertices in S to zero. The Optimal Solution is denoted by OPT(S) and its cost by opt(S). Finally the performance ratio of an approximation algorithm is the supremum on the length of the minimum Steiner tree found by the algorithm divided by the length of an optimal solution.

$$PerformanceRatio = sup(\frac{smt(T)}{opt(T)}) \tag{2.1}$$

2.4 Known Algorithms and their Performance Ratios

The first approximation for the Steiner Tree problem, which reached a performance ratio of 2 is the MST-Approximation-Algortihm by Takahashi and Hiromitsu [10]. Since all other algorithms have built upon the existing Foundation and therefore every other approximation algortihm is built around this MST-Approximation. For a full overview of the approximation algorithm leading up to Hougardy and Proemel's approximation algorithm and their respective performance ratios see Table 2.1

Table 2.1: Known Approximation Algorithms and their Performance Ratios [3]

Author(s)	Performance Ratio	Year
Takahashi, Hiromitsu	2.000	1980
Zelikovsky	1.834	1993
Berman, Ramaiyer	1.734	1994
Zelikovsky	1.694	1995
Proemel, Steger	1.667	1997
Karpinski, Zelikovsky	1.644	1997
Hougardy, Proemel	1.598	1998

3 Implementation

3.1 MST-Algorithm

To implement the MST-Algorithm we first build the metric closure $\bar{G}$ of the input Graph G. Then we take the subgraph $\bar{G}_T$ of this metric closure containing all required terminal nodes T. Using the subgraph as input for a minimum spanning tree algorithm, we receive a tree which now consists of only terminals and edges, that represent the shortest paths between them. In this implementation we are going to use Joseph Kruskal's minimum spanning tree algorithm [8], since it's simple, intuitive and is also easily reusable for building the minimum spanning forest we need to compute the loss of a Steiner tree. The final step is for us to replace the edges of the tree by the corresponding shortest paths in G and the outcome is our Steiner tree approximation.

3.2 Berman-Ramaiyer-Algorithm

The approximation algorithm by Berman and Ramaiyer is split into two phases which both maintain a spanning tree M of T, which is initialized to the output of our MST-Algorithm. The first phase is called the evaluation phase and it uses the *prepareChange* procedure, which we are going to look at shortly, to compute two sets of edges called the Add-Set A and the Remove-Set R for every j-element subset $\tau \subseteq T$, with j being increased up to a maximum size bounded by the input number k. It then uses these sets to compute a *gain* of τ, by subtracting the cost of a SMT(τ) from the cost of the Remove-Set. If this *gain* is greater than zero the Remove-Set is removed from M and every edge of the Add-Set has its cost reduced by *gain* and is added to M right afterwards. This marks a tentative preference to add SMT(τ). The subset τ, the Remove-Set R and the Add-Set A are added to a stack σ_j to be read in the construction phase.

The construction phase works on the output of the evaluation phase and starts by initializing the second spanning tree N, which will end up being the submitted solution, with the current version of M. It moves through the stacks σ_j in opposite direction and pops entries from it until the stack is empty. Every entries data is used to revert the changes made to M by subtracting the Add-Set and adding the Remove-Set. If all edges from the Add-Set are still present in N, they are removed and the SMT(τ) is

```
M = SMT(T)
for j=3 to k {
  σ_j=∅
}
for j=3 to k {
  for each j-element subset τ ⊆ T {
    prepareChange(M, τ, R, A);
    gain= cost(R) - smt(τ);
    if gain>0 {
      Decrease the cost of each edge if A by gain;
      M = M \ R ∪ A
      σ_j.push(τ, R, A);
    }
  }
}
    }
```

Figure 3.1: Evaluation Phase from Berman, Ramaiyer (Fig.2 [1])

added in their place. If there are only some but not all edges from A remaining in N, each of these edges $e \subseteq (A \cap N)$ is replaced in N with the minimal cost edge in M, that connects the two components created by removing e. After these changes have been applied for every entry of every stack the approximated minimal Steiner tree is present in N, while M should have reverted to the original input of the evaluation phase, which was an MST-approximation of the minimal Steiner tree. TODO prepareChange

3.3 Hougardy-Proemel-Algorithm

4 Monotonicity and Critical Payments

4.1 Monotonicity

4.2 Critical Payments

4.3 Incentive Compatibility

5 Comparison

5.1 Average Treecost

5.2 Included Steiner Nodes

5.3 Runtime

6 Conclusion

List of Figures

List of Tables

Bibliography

[1] P. Berman and V. Ramaiyer. "Improved Approximations for the Steiner Tree Problem." In: *Journal of Algorithms 17* (1994), pp. 381–408.

[2] Liad Blumrosen and Noam Nisan. "Combinatorial auctions." In: *Algorithmic game theory* 267 (2007), p. 300.

[3] S. Hougardy and H. Prömel. "A 1.598 Approximation Algorithm for the Steiner Problem in Graphs." In: *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 448–453.

[4] Stefan Hougardy. "The Floyd–Warshall algorithm on graphs with negative cycles." In: *Information Processing Letters* 110.8-9 (2010), pp. 279–281.

[5] Richard M Karp. "Reducibility among combinatorial problems." In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[6] M. Karpinski and A. Zelikovsky. "New approximation algorithms for the Steiner problems." In: *Journal of Combinatorial Optimization 1* (1997), pp. 47–65.

[7] Marek Karpinski and Alexander Zelikovsky. "New approximation algorithms for the Steiner tree problems." In: *Journal of Combinatorial Optimization* 1.1 (1997), pp. 47–65.

[8] Joseph B Kruskal. "On the shortest spanning subtree of a graph and the traveling salesman problem." In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[9] Chin Lung Lu, Chuan Yi Tang, and Richard Chia-Tung Lee. "The full Steiner tree problem." In: *Theoretical Computer Science* 306.1-3 (2003), pp. 55–67.

[10] Hiromitsu Takahashi et al. "An approximate solution for the Steiner problem in graphs." In: (1980).

[11] Florian Walch. URL: `https://github.com/fwalch/tum-thesis-latex`.