Online Steiner Tree and Forest Problem

Xinyu Fei Huiwen Jia Jingwen Tang

IOE 691 Course Project April 21, 2021

1 Algorithm

1.1 Steiner tree problem

We consider the natural greedy algorithm for Steiner tree problem. The solutions set S is initialized as an empty set. When a new terminal t arrives, we connect it to the current solution set S using edges of minimum cost. Following from the triangle inequality, the solution set S is always a spanning tree.

Theorem 1. *The greedy algorithm is* $(1 + \log_2 |R|)$ *competitive for Steiner tree where* R *is the terminal set.*

1.2 Steiner forest problem

We consider the natural greedy and a tight algorithm for Steiner forest problem. The solutions set F is initialized as an empty set. When a new terminal pair (s, t) arrives, we find the path connecting s and t using edges of minimum cost on the graph $G|_F$. By G contracting F, we mean we set the cost of the edges in F to be zero. Then we add the new edges in the path we found into F. There is an improved algorithm for Steiner forest problem which achieved a better competitive ratio. The idea is presented in Algorithm 1.

Algorithm 1: A Tight Algorithm

```
Result: F
initialization F = ∅
while pair (s, t) arrives do
    Find the minimum s − t path p in G|_F;
    Augment F ← F ∪ p;
    Let g ∈ ℤ be such that 2^g ≤ d(p) < 2^{g+1};
    for i ≤ j do
        if d(s, N_j) ≥ 2^{j−1} then
            N_j ← N_j ∪ {s};
            π_j(s) ← s;
        else
            F ← F ∪ (s, π_j(s)) where π_j(s) ∈ N_j is the node closest to s;
        end
        Repeat if with s replaced by t;
        Update E_j ← E_j ∪ {(π_j(s), π_j(t))};
    end
end
```

Theorem 2. *The greedy algorithm is* $O(\log_2^2 |R|)$ *competitive for Steiner forest where* R *is the terminal pair set.*

Theorem 3. *The tight algorithm is $O(\log_2 |R|)$ competitive for Steiner forest where R is the terminal pair set.*

2 Numerical Experiments

2.1 Experimental Setting

Steiner tree problem. We test the natural greedy algorithm on 83 instances from SteinLib dataset [1]. These instances are various in the scale of the graph, the sparsity of the graph and the number of terminals. For each instance, we randomly shuffled the order of the arrival of the terminals and test the algorithm for multiple replications. We compare out objective values with the optimal values collected from SteinLib.

Steiner forest problem. We run the Greedy and Tight algorithms on the same instances for several replicates as tree problem. We don't have an optimal solution for the Steiner forest problem instances and it is time-consuming to compute an optimal solution for the IP model. Therefore, we compute a lower bound for each instance. Here, we apply the "ball packing" lower bound, which is the objective value of the dual formulation of the Steiner forest LP relaxation, i.e. $OPT \geq LP \geq r|K|$. For some instances with large number of nodes, if we iterate all subsets, the running time will be out of control. Thus for those large-scale instance, we used only part of all subsets of terminals and picked the maximum dual objective value, which is also a valid lower bound.

2.2 Results of Steiner Tree Problem

In the results table, the column "Instance" presents the names of instances. The columns "$|V|$", "$|E|$" and "$|R|$" show the number of vertices, edges and terminals. The column "opt" shows the optimal value. The columns "min obj" and "max obj" show the minimum and maximum objective values among all the replications and the columns "best ratio" and "worst ratio" show the ratio between objective values and optimal value respectively. The column "theoretical ratio" is computed by the bound $\log(k)$ where k in the number of terminals. We select some representative instances for various kinds of graphs and sizes of terminals. The results are presented in Table 1.

The results show that when the graphs and the number of terminals are small, the online greedy algorithm performs well. Especially in some instances such as "es10fst01", "es10fst02", and "b09", the algorithm successfully obtains the optimal solution. With the increase of the number of vertices, edges and terminals, the ratio between the minimum objective value and the optimal value also increases, which means that it is harder to find the optimal solution by the algorithm in large-scale instances. On the other hand, the ratio between the maximum objective value and the optimal value keeps stable for various instances, indicating that the algorithm has a good upper bound for all the instances and arrivals of terminals. Furthermore, the best ratio and worst ratio are both much less than the theoretical ratio, demonstrating the excellent performance of the algorithm in practice.

Considering the computational time, all the experiments for small and medium graphs complete within 1 second. For instances with large graphs and numerous terminals, we take the dataset C as an example. We present the figures about how the computational time varies with the number of edges and the number of terminals in Figure 1. From the figures, we find that the computational time grows approximately linearly with the number of edges and grows more significantly with the number of terminals.

Table 1: Summary of results of Steiner Tree Problem

Instance	$\|V\|$	$\|E\|$	$\|R\|$	opt	min obj	max obj	best ratio	worst ratio	theoretical ratio
Small, Sparse, Few terminals									
es10fst01	18	20	10	22920745	22920745	29814814	1.00	1.30	3.32
es10fst02	14	13	10	19134104	19134104	19134104	1.00	1.00	3.32
es10fst03	17	20	10	26003678	26496603	34696845	1.02	1.33	2.30
Medium, Sparse, Numerous terminals									
b09	75	94	38	220	220	267	1.00	1.21	5.25
b12	75	150	38	174	179	219	1.03	1.26	5.25
b15	100	125	50	318	326	387	1.03	1.22	5.64
b18	100	200	50	218	228	281	1.05	1.29	5.64
Large, Sparse, Numerous terminals									
es100fst02	339	522	100	75176630	91310527	102139494	1.21	1.36	6.64
es100fst06	301	452	100	74414990	89678033	102492118	1.21	1.38	6.64
Large, Dense, Numerous terminals									
c15	500	2500	250	556	620	685	1.12	1.23	7.97
c20	500	12500	250	267	301	318	1.13	1.19	7.97

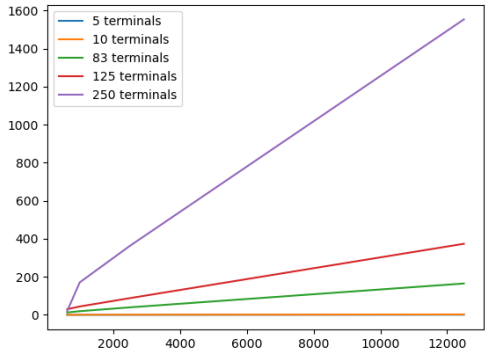
2.3 Results of Steiner Forest Problem

We present the results of the Steiner forest instances in Table 2. The result show that the actual competitive ratios accords with theoretical results. Notice that the lower bound for the large scale graph is not as tight as other instances, which is because, due to the running time constraints, we only used part of the subsets to compute the dual values as lower bound set.

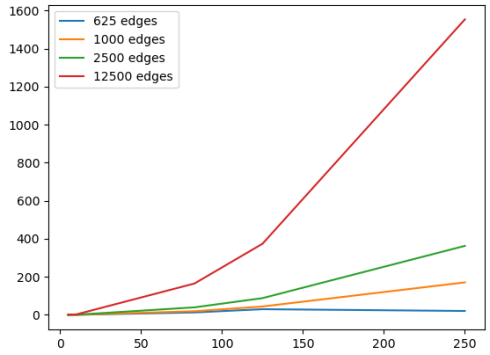
Table 2: Summary of results of Steiner Forest Problem

Instance	$\|V\|$	$\|E\|$	$\|R\|$	lb	greedy max	greedy theo cr	greedy cr	tight max	tight theo cr	tight cr
Small, Sparse, Few pairs										
es10fst01	18	20	5	11536360.00	15959780.00	2.59	1.38	15959780.00	1.61	1.38
es10fst02	14	13	5	13252106.00	18912422.00	2.59	1.43	18912422.00	1.61	1.43
es10fst03	17	20	5	9515702.00	23419028.00	2.59	2.46	23419028.00	1.61	2.46
Medium, Sparse, Numerous pairs										
b09	75	94	18	59.50	190.00	8.51	3.19	189.00	2.92	3.18
b12	75	150	18	45.00	127.00	8.51	2.82	125.00	2.92	2.78
b15	100	125	24	78.00	254.00	10.23	3.26	254.00	3.20	3.26
b18	100	200	24	48.00	190.00	10.23	3.96	186.00	3.20	3.88
Large, Sparse, Numerous pairs										
c05	500	625	124	117.50	1161.00	23.27	9.88	1155.00	4.82	9.83
c10	500	1000	124	125.00	898.00	23.27	7.18	868.00	4.82	6.94
Large, Dense, Numerous pairs										
c15	500	2500	124	109.00	440.00	23.27	4.04	426.00	4.82	3.91
c20	500	12500	124	122.00	221.00	23.27	1.81	214.00	4.82	1.75

Specifically, each instance here denotes a graph and a set of terminal pairs. We shuffle the order of the arrival of pairs randomly and study the worst performance among them. We shuffle for



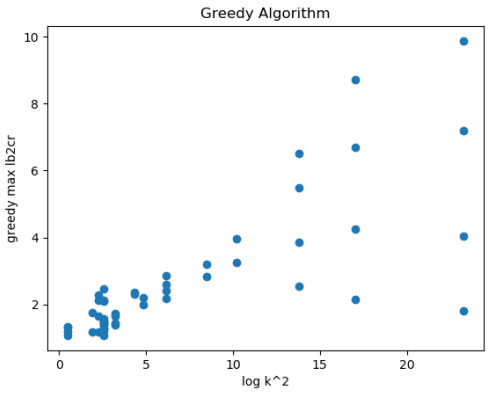
(a) Time varying with edges



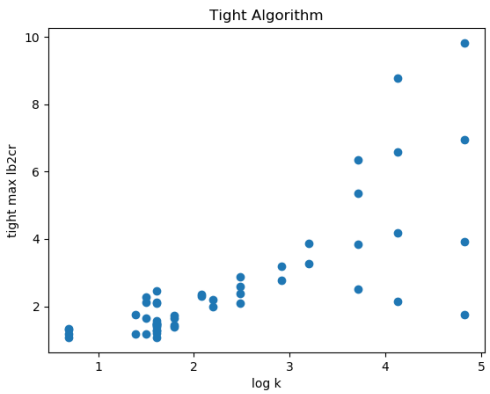
(b) Time varying with terminals

Figure 1: Figures of computational time

$\min\{150, |R|!\}$ times for each instance to set a restriction on running time. Meanwhile, "greedy cr" and "tight cr" are the actual worst case competitive ratio among all shuffled order of pair arrivals of the two algorithms for each instance. The "greedy theo cr" is $\log^2(|R|)$ while the "tight theo cr" is $\log(|R|)$. We can see that for each instance, the actual competitive ratio is within a constant multiplier of the theoretical ratio, which will be further shown in the figures below.



(a) C.r. of Greedy Alg v.s. $\log^2(|R|)$



(b) C.r. of Tight Alg v.s. $\log(|R|)$

Figure 2: Competitive ratio for S.forest of two algs

The two figures show the empirical competitive ratios of two algorithms respectively. From the figures, we can see that all the instances (nodes) lie below a line with a constant slope, which validates the theoretical competitive ratios respectively. Moreover, for our tested instances, the tight algorithm always does slightly better than the greedy algorithm, though the difference is not significant compared to the solution objective value.

References

[1] Koch, T., Martin, A., and Voß, S. (2001). Steinlib: An updated library on steiner tree problems in graphs. In *Steiner trees in industry*, pages 285–325. Springer.