INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 4 (1): 11-15, 2005
ISSN 1812-5638
© 2005 Asian Network for Scientific Information

Improved Steiner Approximation Algorithm Based on Topology Analysis

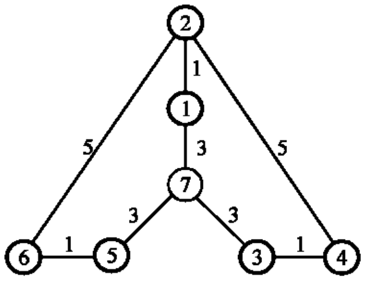
Gao Lei, Zhang Deyun, Wang Lei and Shi Hongfeng
School of Electronics and Information Engineering,
Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China

Abstract: To the case k = 3 of the Berman's approximation algorithm, the improved algorithm is proposed in this study. It constructs a Voronoi region to get the cost of triple subtree on the basis of using Fibonacci heap to count the shortest distance of each pair of points in the corresponding set. Then, to decrease useless triples by the topology analysis of Steiner tree, it simplifies the topology and reduces the time complexity. In the experiment results, the filter factor β is above 0.9 for each example, some even up to 0.999. It shows that many useless triples have been filtered before the beginning of the evaluation phase and the construction phase.

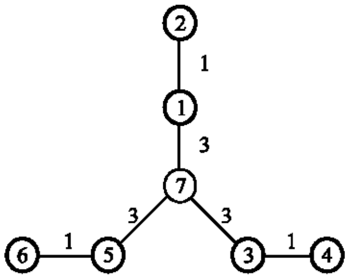
Key words: Steiner tree, Voronoi region, topology analysis, time complexity

INTRODUCTION

Let G = (V, E) be an undirected graph, let each edge (i, j)$\in$E have a nonnegative real-valued cost and let S be a subset of V. A tree T of G (that is, the vertex set V(T) is a subset of V and the edge set E(T) is a subset of E) is a Steiner tree of S in graph if S is contained in the vertex set of T, V(T)$\supseteq$S. The problem is to find a Steiner tree T whose cost $c(T) = \sum_{(i,j)\in E(T)} c(ij)$ is minimum among all Steiner trees; an optimal solution T is called a Steiner minimal tree (SMT for short) and the vertexes in S are called given points (Fig. 1). The problem has been widely applied in the routing in VLSI layout, design and optimization of communication network and so on. Especially, in the application of multicast in network, SMT reduces the total cost and saves bandwidth.



(a) S = {1,2,3,4,5,6}, the vertexes in S are circled in bold



(b) SMT for G
Fig. 1: An example of the Steiner tree problem

It is known that the Steiner tree problem is NP-complete[1,2]. So it is valuable for finding the approximation solution in polynomial time. Berman[3] introduced a good idea to solve it. The algorithm reduces the approximation ration and time complexity.

When $|\tau| = 3$, the ration will be 11/6 and time complexity will be $O(\alpha+|V||S|^2+|S|^{7/2})$ where, α is the time complexity of the all-pairs-shortest-paths problem. Analyses the topology of G, it shows that not all vertexes in G are useful for the solution. With topology analysis of Steiner tree to decrease useless triples, it can simplify the topology and reduce the time complexity. According to the feature, the improved algorithm is presented as the time complexity is $O(|V||S|\log|V|+|E||S|+|V||S|^2+(1-\beta)|S|^{7/2})$ and the ratio is still 11/6. In the experiment results, the value of the filter factor β is very high. It shows the many useless triples are filtered before the beginning of the evaluation phase and construction phase.

BERMAN's ALGORITHM

Lemma 1: Let τ denote a subset of S, assume that $|\tau|>1$. Let e be an edge of maximum cost such that both the connected components of M-{e} contain a vertex of τ. Then there exists a maximum cost removal set for τ in M that contains e.

Corresponding Author: Gao Lei, School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China E-mail: gl@mail.xanet.edu.cn

Lemma 2

1. R is a removal set for τ in M with the maximum cost.
2. A is a spanning tree of τ.
3. cost (A) = cost(R).
4. if f = (u,v)$\in$A, then cost (f) is equal to the largest cost of an edge on the path in M between u and v.

Corollary 1: The edge set M-R $\cup$A is a spanning tree of S.

The computation is divided into to phases, namely, the evaluation phase and the construction phase.

Initially the evaluation phase is executed. The phase processes every subset τ of S that has size between 3 and k, progressing from smaller to larger sets. For every such τ, the removal set R and the add set A are computed by calling the procedure PrepareChange. The gain is computed for the subset τ, the value equals cost(R)-scost(τ). If gain is not positive, then the sets R and A are discarded; τ is never considered again by the algorithm. Otherwise, the cost of every edge in A is decreased by gain and M is modified by actually removing R and adding A; moreover, the tuple [τ,R,A] is pushed onto the stack $\sigma_{|\tau|}$. By Corollary 1, M remains a spanning tree of S after this modification.

After the evaluation phase is completed, the construction phase is executed. The purpose of this phase is to use the entries stored on the stacks to remove all the artificial edges from M and construct the output which is a Steiner tree of S. In particular, this phase will ensure that no edge that belongs to an add set will be present in the final solution.

DESCRIPTION OF IMPROVED ALGORITHM

In Berman's algorithm, every subset τ of S that has size between 3 and k is processed in evaluation phase and construction phase. Finally, approximation SMT is constructed. When $|\tau|$ = 3, the total of triples τ = (i,j,k), i,j,k$\in$S, is $|S|^3$. In the computing proceeding of approximation SMT for this case, there are some features in the computation of scost(τ) and the triples which are needed to be processed in the two phases.

The cost computation of subtree τ with Voronoi region

Definition 1: For any two vertexes v and w of G, let c(v,w) denote the cost of a shortest-path between v and w in G, i.e., $c(v,w) = \min \sum_{i=1}^{L} c(v_{i-1}v_i)$, where, the minimization is over all paths P = $v_0v_1\ldots\ldots v_L$ in G from v = v_0 to w = v_L.

Definition 2: For each given point i$\in$S, define V(i) to be the set containing i and all the vertexes v$\in$V\S that are

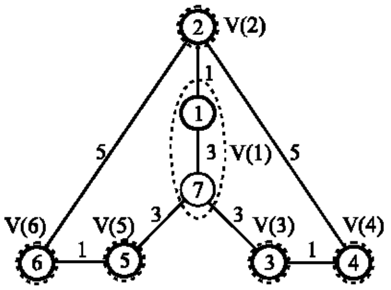


Fig. 2: The division of Voronoi region

closer to i than to any other given point j, j$\neq$i; if v is closest to two or more given points i,j,..., i.e., if c(v,i) = c(v,j) = ... = $\min_{k\in S} c(v,k)$, then v is included in the set of the given point having the smallest index among i,j,.... In symbols, the Voronoi Region of every given point shows as follows:

$$V(i) = \{i\} \cup \{v \in V\backslash S:\ c(v,i) = \min_{k\in S} c(v,k) \text{ and }$$

$$i = \min\{j = 1,2.\ldots.,|S|:\ c(v,i) = \min_{k\in S} c(v,k)\}\}$$

With the following two steps, the cost of subtree τ is computed.

Step 1: Computing the shortest-path between every vertex of S and every vertex of V with Fibonacci Heap[4].

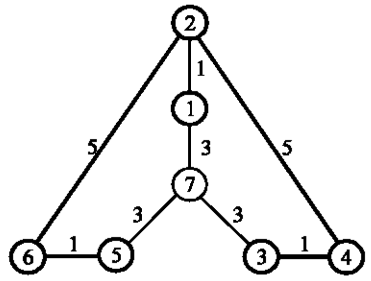
Step 2: For every triples τ = (i,j,k), computing the Steiner subtree, scost(τ), with Voronoi Region.

Figure 2 shows the division of Voronoi Region for Fig. 1 as an example.

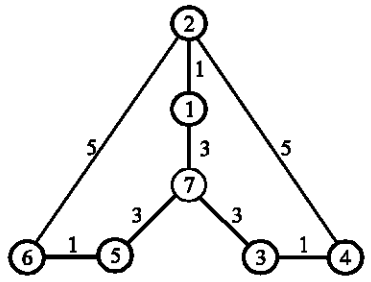
The description of step 2 shows as follows:

```
Triples ← { $\tau \subset S$: $|\tau| = 3$}
For every $\tau\in$Triples do
Find $v \in \cup_{s\in\tau} V(s)$ which minimizes $\sum_{s\in\tau} c(v,s)$
 $v(\tau) \leftarrow v$
 $scost(\tau) \leftarrow \sum_{s\in\tau} c(v(\tau),s)$
```

Topology simplify: In the evaluation phase, the minimum spanning tree M turn to $M^{(n)}$ in the n-th step of the iteration. For the triple τ = (i,j,k), let (...,i,...,j,...,k,...) denote the traversing path of minimal spanning tree M and R_{ijk} denote the removal edge set of τ in the m-th iteration of evaluation phase, R_{ijk} = {save$^{(n)}$(i,j), save$^{(n)}$(j,k)}. We denote save$^{(n)}$(i,j) as the max edge from i to j in $M^{(n)}$, the weight is save$^{(n)}_{ij}$. Similarly, Let save$^{(n)}$(j,k)



(a) Minimum spanning tree is indicated by thick lines



(b) SMT is indicated by thick lines

Fig. 3: Comparison between minimum spanning tree and SMT for Fig. 1

denote the max edge from j to k in $M^{(n)}$, the weight is $save^{(n)}_{jk}$.

In Fig. 3, (a) is the minimum spanning tree M and (b) is SMT, the final solution we get. It is obviously that only the triple (1,3,5), not all the triples, will contribute to the evaluation phase and construction phase. How to discard the useless triples before the beginning of evaluation phase? The following lemma will show the way.

Lemma 3: For the triple $\tau = (i,j,k)$, if $save_{ij}+save_{jk} \le scost(\tau)$, τ will not be under consideration in construction phase.

Proof: Berman's algorithm shows that the triple τ is discarded in evaluation phase and will not be processed in construction phase, iff, $save^{(n)}_{ij}+save^{(n)}_{jk} \le scost(\tau)$ is true. Observing the triple τ that has the feature $save_{ij}+save_{jk} \le scost(\tau)$, if the inequality $save^{(n)}_{ij}+save^{(n)}_{jk} \le scost(\tau)$ is true in evaluation phase, then the triple will be discarded and absent in construction phase.

Assume the triple $\tau' = (u,v,w)$ is the former of τ. Consider such triple τ' that it is one part of the tuple $[\tau',R,A]$ which is pushed onto the stack σ_3 since it will change the set R_{ijk}. Assume that the triple is processed in the m-th iteration of evaluation phase, there is $R_{uvw} = \{save^{(m)}(u,v), save^{(m)}(v,w)\}$.

If for every τ' exits $R_{uvw} \cap \{save(i,j), save(j,k)\} = \phi$, obviously, the edge $save^{(m)}(i,j)$ is still save(i,j) and $save^{(m)}(j,k)$ is save(j,k) too. Until the processing of τ, there is $R_{ijk} = \{save^{(n)}(i,j), save^{(n)}(j,k)\} = \{save(i,j), save(j,k)\}$. Therefore, $save^{(n)}_{ij}+save^{(n)}_{jk} = save_{ij}+save_{jk} \le scost(\tau)$ and $gain(\tau) \le 0$. Then τ is discarded and absent in construction phase.

Inversely, if $R_{uvw} \cap \{save(i,j), save(j,k)\} \neq \phi$, that means $save^{(n)}(i,j)$ will be save'(i,j) and $save^{(n)}(j,k)$ will be save'(j,k). It is now clear that in order to prove the lemma it suffices to show the following: For the triple τ that $save_{ij}+save_{jk} \le scost(\tau)$, $save'_{ij}+save'_{jk} \le scost(\tau)$. The rest of this section is devoted to proving this.

Case 1: $|R_{uvw} \cap \{save(i,j), save(j,k)\}| = 1$. Without loss of generality, we can only consider the case that $save^{(m)}(u,v)$ is save(i,j). The topology of $M^{(m-1)}$ has four cases: (1){…,u,…,i,…,v,…j,…}, (2){…,u,…,i,…, j,…,v,…},(3){…,i,…,u,…,v,…,j,…},(4){…,i,…,u,…,j,…v, …}.After the processing of τ', $M^{(m)}$ will be only one topology, {…,i,…,u,v,…, j,…}. Obviously, there is $save'_{ij} = \max\{save'_{iu}, c(u,v), save'_{vj}\}$, $save'_{iu} \le save_{ui} \le save_{uv} = save_{ij}$, $save'_{vj} \le save_{vj} \le save_{uv} = save_{ij}$, $c(u,v) = save^{(m)}_{uv} - gain(\tau') = save_{ij} - gain(\tau')$. Since there is $gain(\tau') > 0$, we have $c(u,v) < save_{ij}$. Hence $save'_{ij} \le save_{ij}$. Similarly, $save'_{jk} \le save_{jk}$, if $save^{(m)}(v,w)$ is save(j,k). Now, for Case 1, we can draw the conclusion that $save'_{ij}+save'_{jk} \le save_{ij}+save_{jk} \le scost(\tau)$.

Case 2: $|R_{uvw} \cap \{save(i,j), save(j,k)\}| = 2$, that means $save^{(m)}(u,v)$ is save(i,j) and $save^{(m)}(v,w)$ is save(j,k). There're eight cases for the topology of $M^{(m-1)}$: (1){…,u,…,i,…, v,…,j,…,w,…, k,…},(2){…,u,…,i,…,v,…, j,…,k,…,w,…},(3){…,i,…,u,…,v,…,j,…,w,…, k,…},(4){…,i,…,u,…,v,…,j,…,k,…,w,…},(5){…,u,…,i,…,j ,…,v,…,w,…,k,…},(6){…,u,…,i,…,j,…,v,…,k,…,w,…},(7) {…,i,…,u,…,j,…,v,…,w,…,k,…},(8){…,i,…,u,…,j,…,v,…, k,…,w,…}. At the end of the processing for τ', the topology of $M^{(m)}$ has two cases: {…,i,…,u, v,…,j,…,v,w,…,k,…}and{…,i,…,u,w,…,k,…,w,v,…,j,…}. Without loss of generality, we can just consider the first result. Obviously, there is $save'_{ij} = \max\{save'_{iu}, c(u,v), save'_{vj}\}$, $save'_{iu} = save_{ui} \le save^{(m)}_{uv} = save_{ij}$, $save'_{vj} = save_{jw} \le save^{(m)}_{uv} = save_{ij}$, $c(u,v) = save^{(m)}_{uv} - gain(\tau') < save^{(m)}_{uv} = save_{ij}$. As a result, $save'_{ij} \le save_{ij}$. Similarly, we can get that $save'_{jk} \le save_{jk}$. Therefore, $save'_{ij}+save'_{jk} \le save_{ij}+save_{jk} \le scost(\tau)$.

From the analysis of Case 1 and Case 2, it shows the following: if there is $|R_{uvw} \cap \{save(i,j), save(j,k)\}| \neq \phi$, it will be sure that $save'_{ij}+save'_{jk} \le scost(\tau)$ after processing of τ'. Process the triples till τ. At this point, the removal edge set of τ is $R_{ijk} = \{save^{(n)}(i,j), save^{(n)}(j,k)\}$ with the feature $save^{(n)}_{ij}+save^{(n)}_{jk} \le scost(\tau)$. Therefore, $gain(\tau) \le cost(R_{ijk}) - scost(\tau) \le 0$ and τ is discarded.

This completes the proof of Lemma 3.
We now describe the procedure Topology-Cut:

```
precedure Topology-Cut (M,Triples)
Begin
for every $i,j \in S$ do
    Find $save_{ij}$ in spanning tree M
for every $\tau = \{i,j,k\} \in$Triples
    if $save_{ij}+save_{jk} \le scost(\tau)$ then
        remove $\tau$ from Triples
End
```

COMPLEXITY ANALYSIS

In the computation of scost(τ) with Voronoi Region, step1 requires $O(|V||S|\log|V|+ |E||S|)$ steps and one can execute step 2 in $O(|V|\log|V|+|E|+|S|^3+|V||S|^2)$. So the running time is $O(|V||S|\log|V|+|E||S|+ |S|^3+|V||S|^2)$.

To the Topology-Cut procedure, it requires $O(|S|^2\log|S|)$ for finding $save_{ij}$ in spanning tree M and $O(|S|^3)$ to remove the useless τ from Triples. So the running time is $O(|S|^3)$. With the processing of the Topology-Cut procedure, the total of Triples is reduced to $(1-\beta)|S|^3$.

In conclusion, the total time complexity of the improved algorithm is $O(|V||S|\log|V|+ |E||S|+|V||S|^2+(1-\beta)|S|^{7/2})$. The experiments show that the filter factor β is very high. After the filtering, there are only few triples left to be processed in the continued computation, evaluation phase and construction phase.

NUMERICAL RESULTS

In the experiments of the improved algorithm, the test data comes from the SteinLib library[5], which is built by Thorsten Koch and Alexander Martin. PC for the experiments is Intel PIII 800 with 128M SDRAM.

From Table 1, it shows the following results:

1. The improved algorithm has good performance. The approximation solution is closed to the optimal solution, even equals to latter for some examples.

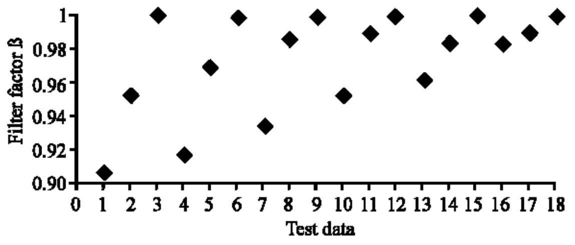


Fig. 4: The value of the filter factor β

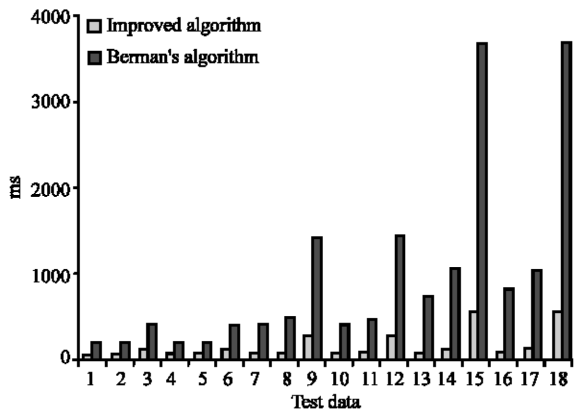


Fig. 5: Comparison of running time

Table 1: Results of the improved algorithm

Test data	|V|	|E|	|S|	Optimal solution	Approxi-mation solution	Total of triples before filtering	Total of triples after filtering	β	Running time of Berman (ms)	Running time of improved algorithm (ms)
b01	50	63	9	82	82	84	8	0.9048	135	7
b02	50	63	13	83	85	286	14	0.9510	141	13
b03	50	63	25	138	139	2300	2	0.9990	354	66
b04	50	100	9	59	59	84	7	0.9167	147	8
b05	50	100	13	61	62	286	9	0.9685	137	13
b06	50	100	25	122	124	2300	6	0.9974	355	68
b07	75	94	13	111	111	286	19	0.9336	346	15
b08	75	94	19	104	104	969	14	0.9856	458	36
b09	75	94	38	220	222	8436	11	0.9987	1390	228
b10	75	150	13	86	87	286	14	0.9510	357	16
b11	75	150	19	88	88	969	11	0.9886	409	33
b12	75	150	38	174	174	8436	11	0.9987	1409	235
b13	100	125	17	165	172	680	26	0.9618	695	27
b14	100	125	25	235	236	2300	39	0.9830	1030	75
b15	100	125	50	318	318	19600	24	0.9988	3689	549
b16	100	200	17	127	133	680	12	0.9824	781	31
b17	100	200	25	131	133	2300	23	0.9900	1026	76
b18	100	200	50	218	218	19600	19	0.9990	3682	555

2. For every example, β is above 0.9. It even reaches 0.999 especially for some examples. Be the performance of the Topology-Cut procedure, there are only few triples left to be processed in evaluation phase and construction phase after the filtering.
3. Observing each example group with invariable |V| and |E| (b01~b03, b04~b06, b07~b09, b10~b12, b13~b15, b16~b18), it is obviously that bigger the size of S, higher the β (Fig. 4). That means the Topology-Cut procedure has better performance as the topology of Steiner tree turn to more complex when the size of S increases.
4. The decrease of time complexity reduces the running time (Fig. 5). The improved algorithm runs faster.

CONCLUSION AND FUTURE WORK

This study we analyzed the topology of Steiner tree in case k = 3 and presented an improved algorithm. The proposed algorithm reduces the total time complexity with the ration 11/6. In the experiment results, filter factor β is above 0.9, even 0.999 for some examples. It shows that many useless triples are filtered before the beginning of the evaluation phase and construction phase. Future work will address how to simplify topology in other cases (k>3).

REFERENCES

1. Warme, D.M., P. Winter and M. Zachariasen, 2000. Exact Algorithms for Plane Steiner Tree Problems: A Computational Study, Advances in Steiner trees. In: Du, D.Z., J.M. Smith and J.H. Rubinstein (Eds.). Advances in Steiner Trees. Norwell, Massachusetts: Kluwer Academic Publishers, pp: 81-116.
2. Ganley, J.L., 1999. Computing optimal rectilinear Steiner trees: A survey and experimental evaluation. Discrete. Applied Math., 90: 161-171.
3. Berman, P. and V. Ramaiyer, 1994. Improved appoximation for the Steiner tree problem. J. Algorithms, 17: 381-408.
4. Fredman, M.L. and R.E. Tarjan, 1987. Fibonacci heaps and their uses in improved network optimization algorithms. J. Assoc. Computer. Mach., 34: 596-615.
5. http://elib.zib.de/steinlib/steinlib.php