

Fast Approximation Algorithm for Restricted Euclidean Bottleneck Steiner Tree Problem

Zimao Li* and Wenying Xiao

College of Computer Science, South-Central University for Nationalities, Wuhan City, Hubei Province, P. R. China

*Corresponding author, Email: lizm@mail.scuec.edu.cn, 1047464075@qq.com

Abstract—Bottleneck Steiner tree problem asks to find a Steiner tree for n terminals with at most k Steiner points such that the length of the longest edge in the tree is minimized. The problem has applications in the design of wireless communication networks. In this paper we study a restricted version of the bottleneck Steiner tree problem in the Euclidean plane which requires that only degree-2 Steiner points are possibly adjacent in the optimal solution. We first show that the problem is NP-hard and cannot be approximated within $\sqrt{2}$ unless $P=NP$, and provide a fast polynomial time deterministic approximation algorithm with performance ratio $\sqrt{3}$.

Index Terms—Bottleneck Steiner Tree; Approximation Algorithm; Performance Ratio; Wireless Networks

I. INTRODUCTION

In the 1990s, along with the conquest of a series of famous conjectures, the traditional *Steiner tree* problem [1] attracted the scientists' considerable attention and interests from both theoretical point of view and its applicability and once occupied a central place in the emerging theory of approximation algorithms.

Given a weighted graph $G=(V,E;W)$ and a subset $S \subset V$ of required vertices, the traditional Steiner tree problem asks a shortest tree spanning S . The tree may use additional points (called Steiner points) in $V-S$. We call such a tree a Steiner tree. The problem is MAX-SNP hard even when the edge weights are only 1 or 2 [2]. For the Steiner tree problem in Euclidean plane, it is still NP-hard and there is a polynomial-time approximation scheme (PTAS) [3] for Euclidean Steiner trees, i.e., a near-optimal solution can be found in polynomial time [4].

New applications of Steiner tree problem in VLSI routing [5], wireless communications [6] and phylogenetic tree reconstruction in biology [7] have been found and studied deeply. These applications generally need to do some modification to the classic Steiner tree problem. Therefore, in recent years the study of variations of traditional Steiner tree problem become a hot issue.

For example, in the design of wavelength division multiplexing (WDM) optical network, suppose we need to connect the n nodes located at p_1, p_2, \dots, p_n by WDM optical network, due to transmit power limit, signal can only transmit a limited distance to ensure correct transmission. If the distance between some nodes in the

connection tree is large, signal amplifiers are required to place at proper positions to shorten the connection distance. WDM optical network design leads us to consider minimizing the maximum edge length problem and minimizing the number of Steiner points problem, implying the two variations of the traditional Steiner tree problem: the *bottleneck Steiner tree problem* [8] and the *Steiner tree problem with minimum number of Steiner points and bounded edge-length* [9, 10, 12, 13].

In this paper, we consider the bottleneck Steiner tree problem, which is defined as follows: given a set $P=\{p_1, p_2, \dots, p_n\}$ of n terminals and a positive integer k , we want to find a Steiner tree with at most k Steiner points such that the length of the longest edges in the tree is minimized.

The problem has applications in the design of wireless communication networks, multifacility location, VLSI routing, network routing and optical switching networks.

The problem is NP-hard. In [8], D.-Z Du and L. Wang proved that unless $P=NP$, the problem cannot be approximated in polynomial time within performance ratios 2 and $\sqrt{2}$ in the rectilinear plane and the Euclidean plane, respectively. Moreover, they gave an approximation algorithm with performance ratio 2 for both the rectilinear plane and the Euclidean plane. For the rectilinear plane, the performance ratio is best possible, that is, the performance ratio is tight. For the Euclidean plane, however, the gap between the lower bound $\sqrt{2}$ and upper bound 2 is still big. Based on the existence of a 3-restricted Steiner tree, we presented a randomized polynomial approximation algorithm with performance ratio $1.866 + \epsilon$, for any positive number ϵ for the Euclidean plane [11]. Later I. Cardei, M. Cardei, L. Wang, B. Xu, and D.-Z Du improved the performance ratio to $\sqrt{3} + \epsilon$, for any positive number ϵ [12, 13]. This is so far the best results possible.

In 2004, we considered a restricted version of the problem in the Euclidean plane which requires that no edge connects any two Steiner points in the optimal solution, we proved that the problem is NP-hard and cannot be approximated in polynomial time within performance ratio $\sqrt{2}$ and proposed a randomized polynomial approximation algorithm with performance ratio $\sqrt{2} + \epsilon$, for any positive number ϵ [14].

S. Bae, C. Lee, and S. Choi studied the Euclidean bottleneck Steiner tree problem when k is restricted to 1

or 2, they gave exact solutions to this problem [15]. M. Li, B. Ma and L. Wang studied the bottleneck Steiner tree problem in String space when $k = 1$ (also called the closest string problem). They proved the problem to be NP-hard and present a PTAS for it, and hence solved it perfectly in theory [16].

Although in [12, 13], the authors presented approximation algorithm with performance ratio $\sqrt{3} + \epsilon$ for the general Euclidean bottleneck Steiner tree problem, the algorithm is not deterministic but randomized, which increased the uncertainty and difficulty of implementation, and hence it is not practical. In this paper, we study the bottleneck Steiner tree problem in the Euclidean plane by allowing only degree-2 Steiner points to be possibly adjacent in the optimal bottleneck Steiner tree and present a polynomial deterministic approximation algorithm with performance ratio $\sqrt{3}$. The case we consider is more general than the restricted version in [14], which requires that no two Steiner points are allowed to be adjacent. We denote the problem *restricted-BST* for short.

The rest of the paper is organized as below. In Section II, we show that the restricted-BST problem is NP-hard and cannot be approximated within $\sqrt{2}$ in the Euclidean plane unless $P=NP$. Based on the notion of full Steiner component, we prove the existence of performance ratio $\sqrt{3}$. In Section III, we provide an intuitive polynomial time approximation algorithm with performance ratio $\sqrt{3}$. Section IV improves the approximation algorithm's time complexity by introducing a heap. The concluding remarks appear in Section V.

II. THE EXISTENCE OF RATIO- $\sqrt{3}$

In this section, we first show that the restricted-BST problem is MAX-SNP hard, then prove the existence of performance ratio $\sqrt{3}$.

Directly following the proof of Theorem 1 in [8], that is, a reduction from planar graph vertex cover with all vertices of degree at most 4 to the restricted-BST problem, where the former problem is known to be NP-hard, we have the following NP-hard result:

Theorem 1: Unless $P=NP$, the restricted-BST problem in the Euclidean plane cannot be approximated within performance ratio $\sqrt{2}$ in polynomial time.

Next we try to design a polynomial time approximation algorithm with performance ratio $\sqrt{3}$ for the problem. Similar to the idea of the algorithm in [8, 14], we first construct a minimum spanning tree for the set of n terminals in P , then we add degree-2 Steiner point to long edges in the minimum spanning tree. We call the resulting tree a *steinerized spanning tree*.

The following two lemmas (Lemma 1 and Lemma 2) show that the length of the longest edges in an optimal steinerized spanning tree is at most $\sqrt{3}$ times the optimum, and the optimal steinerized spanning tree can be found among the steinerized minimum spanning trees.

Usually, every leaf in a bottleneck Steiner tree is a terminal. However, a terminal may not be a leaf.

A bottleneck Steiner tree is *full* if all terminals are leaves. Thus, if a Steiner tree is not full, there must exist a terminal which is not a leaf, we can decompose the tree at this terminal into several edge-disjoint small trees, and these small trees share a common terminal. In this way we can always decompose any Steiner tree into the union of several edge-disjoint small trees, in each of them a vertex is a leaf if and only if it is a terminal. These small trees are called *full Steiner components*, or formally,

Definition 1: A full Steiner component of a bottleneck Steiner tree is a subtree in which each terminal is a leaf and each internal node is a Steiner point.

The following lemma indicates the existence of performance ratio $\sqrt{3}$.

For convenience, let a and b be two terminals in the plane, we denote ab an edge and $|ab|$ the length of ab . Without loss of generality, we assume the length of the longest edges in the optimal bottleneck Steiner tree is 1.

Lemma 1: Given a set of n terminals P in the Euclidean plane, let T be an optimal bottleneck Steiner tree for the restricted-BST problem. Then, there exists a steinerized spanning tree T' for P with the same number of Steiner points as T such that the length of the longest edges in T' is at most $\sqrt{3}$.

Proof: Let T be an optimal bottleneck Steiner tree with k Steiner points for the restricted-BST problem. Because only degree-2 Steiner points are possibly adjacent in the optimal solution, as described above, T can be decomposed into the union of its full components, each of which is either a star with a Steiner point as center or just a line-segment path connecting two terminals with $l \geq 0$ intermediate degree-2 Steiner points (See Figure 1 and Figure 2). Figure 1 illustrates a star with 4 terminals centered at Steiner point v , and Figure 2 present an example of a line-segment path connecting two terminals a and b with 2 intermediate degree-2 Steiner points.

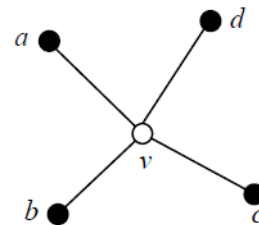


Figure 1. A star with a Steiner point

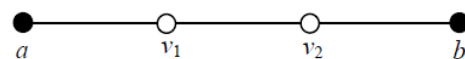


Figure 2. A line-segment path with 2 degree-2 Steiner points

For a star T_s with at least 3 terminals, we can always decrease the degree of the Steiner point step by step to 2 and guarantee the length of the longest edges in the modified tree is at most $\sqrt{3}$. The procedure is as below:

Suppose the Steiner point is labeled as v , there must exist two terminals a and b satisfying $\angle avb \leq 120^\circ$, by directly connecting a and b and removing the longer edge of va and vb , the degree of v is decreased by 1, and it is easily seen that

$|ab| = \sqrt{|va|^2 + |vb|^2 - 2|va| \cdot |vb| \cos \angle avb} \leq \sqrt{3}$ (remember the assumption that the length of the longest edges in the optimal Steiner tree is 1). Repeat the procedure until the degree of v becomes 2. Figure 3 gives an example to illustrate the procedure.

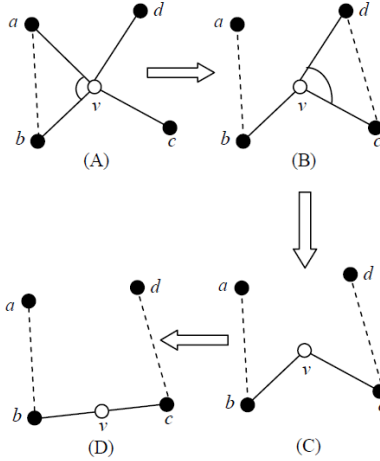


Figure 3. Transform a star to a Steiner subtree

Thus we transform the star T_s into a Steiner subtree in which the length of the longest edges is at most $\sqrt{3}$ and the number of Steiner points in the Steiner subtree does not increase. For the line-segment path full Steiner component, no transformation work is needed because the length of its edges is at most 1. The proof completes by union all the Steiner subtrees to form a steinerized spanning tree T^* with the same number of Steiner points as the optimal bottleneck Steiner tree T .

In the following we study the problem that given a set of terminals and a positive integer k , find a steinerized spanning tree T with k degree-2 Steiner points such that the length of the longest edge in T is minimized. This problem is a bottleneck steinerized spanning tree problem. For simplicity, the optimal solution of this problem is called the *optimal steinerized spanning tree*. We will show that the optimal steinerized spanning tree can be computed in polynomial time, which therefore is a polynomial-time approximation with performance ratio $\sqrt{3}$ by Lemma 1. First we show that the optimal steinerized spanning tree can be found among steinerized minimum spanning trees. The following important fact on the minimum spanning tree indicating that an optimal steinerized spanning tree can be found among steinerized minimum spanning trees.

Lemma 2 [8]: Let e_1, e_2, \dots, e_{n-1} be all edges in a spanning tree T and $e_1^*, e_2^*, \dots, e_{n-1}^*$ be all edges in a minimum spanning tree T^* for the same set P of terminals. Suppose $c(e_i) \leq c(e_{i+1})$ and $c(e_i^*) \leq c(e_{i+1}^*)$ for all $1 \leq i \leq n-1$ where $c(e)$ denotes the length of edge e . Then, $c(e_i^*) \leq c(e_i)$ for all $1 \leq i \leq n-1$.

III. APPROXIMATION ALGORITHM

It follows immediately from Lemma 2 that when we use the same number of Steiner points to steinerize a spanning tree and a minimum spanning tree, the result from the latter has a longest edge of length not exceeding

that from the former. That is, an optimal steinerized spanning tree can be found among steinerized minimum spanning trees. Since only degree-2 Steiner points are possibly adjacent, we only need to add k Steiner points to a minimum spanning tree in order to obtain an optimal steinerized spanning tree.

The idea is explained as follows: for each edge $e_i = (u, v)$ in the minimum spanning tree, if we use l_i degree-2 Steiner points to steinerize it, then the length of the longest edge in the resulting path from u to v has the minimum value $c(e_i)/(l_i+1)$. This minimum value is achieved when the l_i Steiner points divide e_i evenly. Denote $l(e_i) = c(e_i)/(l_i+1)$.

At the beginning of the algorithm, $l(e_i) = c(e_i)$. The basic idea is to add a degree-2 Steiner point to the edge e_i with the largest $l(\cdot)$ value at a time. After e_i receives one more degree-2 Steiner point, l_i is updated by $l_i = l_i + 1$ and $l(e_i)$ is updated by $c(e_i)/(l_i+1)$. The process is repeated until k degree-2 Steiner points are added. In fact, each time we add a Steiner point to some edge, instead of computing the positions of the Steiner points on that edge, we just update the number of Steiner points and the even partition length $l(\cdot)$ of that edge. The positions of all Steiner points can be computed in the last step. The complete algorithm is given in Figure 4.

Algorithm restricted-BST(P, n, k)

Input: A set P of n terminals in the Euclidean plane and an integer k .

Output: A bottleneck Steiner tree T for P with at most k Steiner points.

1. Compute a minimum spanning tree for P , suppose e_1, e_2, \dots, e_{n-1} are all edges in it.
2. Initialize $l(e_i) \leftarrow c(e_i)$ and $l_i \leftarrow 0$ for $1 \leq i \leq n-1$, where $c(e_i)$ denote the length of edge e_i , l_i denote the number of Steiner points added to edge e_i .
3. Find $e_i = (u, v)$ with the largest $l(\cdot)$ value. //add a steiner point to e_i .
4. Update l_i by $l_i + 1$ and $l(e_i)$ by $c(e_i)/(l_i+1)$.
5. Repeat step 3 to 4 until k Steiner points are added, that is $\sum l_i = k$.
6. Organize the l_i Steiner points by evenly partition edge $e_i = (u, v)$ for $1 \leq i \leq n-1$.

Figure 4. Approximation Algorithm for Restricted BST Problem

To confirm our algorithm runs in polynomial, next we analyze our algorithm's time complexity in details.

Step 1 of algorithm restricted-BST can be implemented as below: first, compute the Delaunay triangulation [18] in $O(n \log n)$ time and $O(n)$ space, this will generate $O(n)$ edges; then we label each edge with its length; finally run Prim's minimum spanning tree algorithm on this triangulation to find a minimum spanning tree in $O(n \log n)$ time. So step 1 can be implemented in $O(n \log n)$ time.

Step 2 uses linear time to finish initialization.

In each loop, Step 3 uses linear time to find the longest edge, while Step 4 uses constant time.

As Step 5 only loops k times, Step 3 and Step 4 run in $O(kn)$ in total.

Step 6 runs in $O(n+k)$ time to determine the position of Steiner points.

So the algorithm's time complexity is $O(n \log n + kn)$.

Combined with Lemma 2 and the above analysis, we have the following theorem.

Theorem 2: The algorithm restricted-BST is an $O(n\log n + kn)$ time approximation algorithm with performance ratio $\sqrt{3}$ for the restricted bottleneck Steiner tree problem in the Euclidean plane.

IV. FASTER ALGORITHM

The most time consuming step in the loop of the algorithm restricted-BST is Step 3 because each time we have to scan the edge list to find the edge with largest $l(\cdot)$. But if it is a good idea to sort the edges by $l(\cdot)$ in non-increasing order? The sorting indeed results in Step 3 to run in constant time, but it will sacrifice the time of Step 4 to run in linear time to keep the order. Thus, the algorithm's time complexity will not change.

In algorithm restricted-BST, the step to find an edge with the largest $l(\cdot)$ and step to update $l(\cdot)$ are frequently executed, which inspires us to use a priority queue to maintain all the edges associated with priority $l(\cdot)$ [17]. The priority queue should support two operations efficiently: including finding an edge with the largest priority and update an edge's priority.

According to our case, a *binary max-heap* [17] is suitable to implement the priority queue. A binary max-heap is a heap data structure created using a binary tree with two additional constraints: (1) shape property, the tree is a complete binary tree; that is, all levels of the tree, except possibly the last one are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right; (2) heap property, the key at each node is greater than or equal to that of its children.

A max-heap supports the operations of a priority queue efficiently. We can construct a heap in linear time, and a max-heap returns a node with the largest key in $O(1)$ time, and updates a node key in $O(\log n)$ time.

Now we can formulate our improved algorithms as below (Figure 5).

Algorithm faster-restricted-BST (P, n, k)

Input: A set P of n terminals in the Euclidean plane and an integer k .

Output: A bottleneck Steiner tree T for P with at most k Steiner points.

1. Compute a minimum spanning tree for P , suppose e_1, e_2, \dots, e_{n-1} are all edges in it.
2. Initialize $l(e_i) \leftarrow c(e_i)$ and $l_i \leftarrow 0$ for $1 \leq i \leq n-1$, where $c(e_i)$ denote the length of edge e_i , l_i denote the number of Steiner points added to edge e_i .
3. Bottom-up manner to construct a max-heap for e_1, e_2, \dots, e_{n-1} by keys $l(\cdot)$.
4. Find $e_i = (u, v)$ with the largest key $l(e_i)$ from the constructed max-heap.
5. Update l_i by $l_i + 1$ and $l(e_i)$ by $c(e_i)/(l_i + 1)$ in the max-heap. //add a steiner point to e_i .
6. Repeat step 4 to 5 until k Steiner points are added, that is $\sum l_i = k$.
7. Organize the l_i Steiner points by evenly partition edge $e_i = (u, v)$ for $1 \leq i \leq n-1$.

Figure 5. Faster algorithm for Restricted BST Problem

In the above algorithm,
Step 1 is in $O(n\log n)$;

Step 2 uses linear time;

Step 3, the construction of a max-heap can be done in linear time;

Both Step 4 to return an edge with the largest key and Step 5 to update an edge's key use $O(\log n)$ time in each loop;

As Step 6 only loops k times, Step 4 and Step 5 run in $O(k\log n)$ time in total.

Step 7 runs in linear time.

So the algorithm's time complexity is $O(n\log n + k\log n)$, thus we have Theorem 3.

Theorem 3: There is an $O(n\log n + k\log n)$ time approximation algorithm with performance ratio $\sqrt{3}$ for the restricted bottleneck Steiner tree problem in the Euclidean plane.

V. CONCLUSION

We mainly considered a restricted version of the bottleneck Steiner tree problem in the Euclidean plane. The problem asks to find a Steiner tree with n fixed terminal nodes and up to k Steiner nodes such that the length of the longest edge in the tree is minimized and only degree-2 Steiner points are possibly to be adjacent in the optimal tree.

We first showed that the restricted version is NP-hard and cannot be approximated within $\sqrt{2}$ unless $P=NP$, then we presented a $O(n\log n + kn)$ algorithm with performance ratio $\sqrt{3}$ and based on the heap data structure, we improved the algorithm's time complexity to $O(n\log n + k\log n)$. This faster deterministic algorithm beats the randomized approximation algorithm with performance ratio $\sqrt{3} + \varepsilon$, for any positive number ε [12, 13], whether in efficiency or in accuracy.

As an application, the algorithm can be used to improve the lifetime of wireless networks by minimizing the length of the longest edge in the interconnecting tree by deploying additional relay nodes at specific locations.

ACKNOWLEDGMENT

The author wish to thank Dr. Rongbo Zhu for his double-check and suggestions on this paper. This work was fully supported by National Natural Science Foundation of China (Project 61103248 and 61379059).

REFERENCES

- [1] M. R. Garey, R. L. Graham and D. S. Johnson, "The Complexity of Computing Steiner Minimal Trees," *SIAM Journal on Applied Mathematics*, vol. 32, pp. 835-859, 1977.
- [2] M. Bern and P. Plassmann, "The Steiner Problem with Edge Lengths 1 and 2," *Information Processing Letters*, vol. 32, pp. 171-176, 1989.
- [3] M. R. Garey and D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness," W. H. Freeman and Company, New York, 1979.
- [4] S. Arora, "Polynomial Time Approximation Scheme for Euclidean TSP and Other Geometric Problems," *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, Burlington VT, pp. 2-11, Oct. 1996.

- [5] A. Kahng and G. Robins, "On Optimal Interconnections for VLSI," Kluwer Publishers, 1995.
- [6] A. Caldwell, A. Kahng, S. Mantik, I. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-based Placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 18, pp. 1265-1278, 1999.
- [7] F. K. Hwang, D. S. Richards and P. Winter, "The Steiner Tree Problem," North-Holland, 1992.
- [8] L. Wang and D. -Z Du, "Approximations for a Bottleneck Steiner Tree Problem," *Algorithmica*, vol. 32, pp. 554-561, 2002.
- [9] M. Sarrafzadeh and C. Wong, "Bottleneck Steiner Trees in the Plane," *IEEE Transactions on Computers*, vol. 41, pp. 370-374, 1992.
- [10] G. Lin and G. Xue, "Steiner Tree Problem with Minimal Number of Steiner Points and Bounded Edge-length," *Information Processing Letters*, vol. 69, pp. 53-57, 1999.
- [11] L. Wang and Z. Li, "An approximation algorithm for a bottleneck k -Steiner tree problem in the Euclidean plane," *Information Processing Letters*, vol. 81, pp. 151-156, 2002.
- [12] D. -Z Du, L. Wang and B. Xu, "The Euclidean Bottleneck Steiner Tree and Steiner Tree with Minimum Number of Steiner Points," in *Proceedings of the 7th Annual International Conference on Computing and Combinatorics*, Guilin, China, LNCS vol. 2108, pp. 509-518, August 2001.
- [13] I. Cardei, M. Cardei, L. Wang, B. Xu, D. -Z. Du, "Optimal relay location for resource-limited energy-efficient wireless communication," *Journal of Global Optimization*, vol. 36, pp. 391-399, 2006.
- [14] Z. Li, D. Zhu and S. Ma, "Approximation algorithm for bottleneck Steiner tree problem in the Euclidean plane," *Journal of Computer Science and Technology*, vol. 19, pp. 791-794, 2004.
- [15] S. Bae, C. Lee, and S. Choi, "On Exact Solutions to the Euclidean Bottleneck Steiner Tree Problem," in *Proceedings of the Third Annual Workshop on Algorithms and Computation*, Kolkata, India, LNCS vol. 5431, pp. 105-116, Feb. 2009.
- [16] M. Li, B. Ma and L. Wang, "On the Closest String and Substring Problems," *Journal of the ACM*, vol. 49, pp. 157-171, 2002.
- [17] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms (3rd edition.)," MIT Press and McGraw-Hill, 2009.
- [18] B. Delaunay, "Sur la sphère vide. A la mémoire de Georges Voronoi," *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, no. 6, pp. 793-800, 1934.



Zimao Li, Ph.D., associate professor and vice dean, born in Linqing, P.R. China, 1974, received his Bachelor's degree in Mathematics, Master's degree in Computer Science from Shandong University in 1996 and 1999, respectively, and his Ph.D. degree in Computer Science from City University of Hong Kong in 2002. His research interests include computational complexity, approximation algorithms and design and analysis of algorithms.

From 2002 to 2005, he was a lecturer of College of Computer Science and Technology, Shandong University, P.R. China; from 2005, he is an associate professor of College of Computer Science, South-Central University for Nationalities, P.R. China; from 2012, he is the vice dean of College of Computer Science, South-Central University for Nationalities. He has been supported by 1 national fund and 3 provincial level funds and published more than 10 research papers in journals such as SIAM Journal on Computing, IEEE Transactions on SMC, Journal of Computer Science and Technology, Information Processing Letters, etc.



Wenying Xiao, born in Anguo, P.R. China, 1988, received his Bachelor's degree in Computer Science and Technology from South-Central University for Nationalities in 2012, currently he is a Master degree candidate. His research interest is design and analysis of algorithms.