Wireless Sensor Network, **2009, 1, 446-452**
doi:10.4236/wsn.2009.15053 Published Online December 2009 (http://www.scirp.org/journal/wsn).

Group Target Tracking in WSN Based on Convex Hulls Merging

Quanlong LI, Zhijia ZHAO, Xiaofei XU, Tingting ZHOU
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
Email: *liquanlong@hit.edu.cn*
Received July 22, 2009; *revised September* 21, 2009; *accepted September* 22, 2009

Abstract

When a mass of individual targets move closely, it is unpractical or unnecessary to localize and track every specific target in wireless sensor networks (WSN). However, they can be tracked as a whole by view of group target. In order to decrease the amount of energy spent on active sensing and communications, a flexible boundary detecting model for group target tracking in WSN is proposed, in which, the number of sensors involved in target tracking is adjustable. Unlike traditional one or multiple individual targets, the group target usually occupies a large area. To obtain global estimated position of group target, a divide-merge algorithm using convex hull is designed. In this algorithm, group target's boundary is divided into several small pieces, and each one is enclosed by a convex hull which is constructed by a cluster of boundary sensors. Then, the information of these small convex hulls is sent back to a sink. Finally, big convex hull merged from these small ones is considered as the group target's contour. According to our metric of precision evaluation, the simulation experiments confirm the efficiency and accuracy of this algorithm.

Keywords: Sensor Networks, Target Tracking, Group Target, Convex Hull

1. Introduction

Target tracking is a basic application of WSN, and hence has received a considerable amount of attentions in research community [1–5]. However, most of the previous works are limited to investigating point target tracking, where a target is regarded as a point and the localization result is also a point. There are also several methods which have discussed to solve multi-target tracking in WSN, e.g. [4]. However, in order to distinguish different targets, more complicated computations and communications are required, especially when the number of targets is relatively large. In fact, when a group of targets move closely to each others (e.g., motorcade, tank column, or a herd of buffalo) in a sensor network, it is unpractical or pointless to localize every specific target. Therefore, the schemes for tracking point targets in WSN are not suitable for the cases where there are large quantities of targets moving within a huge area. Instead, it's more valuable and feasible to track these targets as a whole called group target.

Recently, there are several works on group target tracking in WSN [6–9]. In order to detect chemical pollution-like events, [6] proposed a scheme for contour tracking by constructing and maintaining a contour network which tightly surrounds the contours and captures the important topological features. A fully distributed protocol, named CollECT, is proposed in [7] to detect and track events in a wireless heterogeneous sensor network. A dynamic cluster-based structure is introduced in [8] to detect and track the movements of boundaries of continuous objects in a sensor network. The authors in [9] estimate a group target's boundary as a circle covering all border sensors.

Unlike all the above mentioned revelatory works, we proposed a group target tracking protocol on the boundary detecting model, clustering model and group target's boundary construction algorithm. As a first attempt, the idea of convex hulls merging is used to track group target.

In this method, the target tracking process is separated into two steps: boundary dividing and boundary merging. In the first step, the sensors chosen to detect the boundary of a group target are divided into multiple clusters and each cluster is responsible for tracking a partial boundary of the group target. In each cluster, there is a cluster head (CH) which gathers information from its cluster members (CM) and aggregates these data to form a local convex hull. Then, the aggregated data is sent back to the sink which is usually monitored by humans. In the second step, when sufficient information of local convex hulls is collected at the sink, it will execute the merging algorithm to combine those convex hulls into a global convex hull which is considered as the whole

contour of the group target.

The rest of the paper is organized as follows. In Section 2, two analytical models for group target tracking in WSN are proposed. In Section 3, the group target tracking algorithm is designed. The details of simulations are discussed in Section 4. Finally, Section 5 concludes this paper.

2. Analytical Model

Unlike the point target tracking, the problem of group target tracking is much more challenging because of several reasons. Firstly, if considering target as a point, it's easier to measure the location result and obtain the localization accuracy. However, in the group target tracking case, the location result is an area. Then, the question is how to measure the accuracy of localizing group target? Secondly, the group target always occupies a region which is covered by numerous sensors, but some of them may be too far to communicate. So how could they cooperate with each other to efficiently track the same group target? Thirdly, if we want to divide the tracking nodes into several groups, what are the rules to categorize them and how could these segments be merged to form a global contour? All these questions will be clarified by the novel methods which will be stated in this and following sections.

2.1. Boundary Detecting Model

In [6], there is a simple method to detect object's boundary, but the width of boundary is fixed. In the following, a more flexible detecting model called Boundary Detecting Model (BDM) will be proposed, where the width of boundary is adjustable. The model is built upon the binary sensing model [3,5], which is famous for its minimal requirements of sensing and processing capabilities. Moreover, it can be easily extended to be applied on other types of sensors.

Definition 2.1 Discovering Neighbors Ratio (DNR): The Discovering Neighbors Ratio (DNR) is defined as a function η of node i:

$$\eta : I \to [0,1]$$
$$\eta(i) = \frac{\varphi(i)}{\theta(i)}, \text{ for all } i \in I \quad (1)$$

where I is set of sensors, $\varphi(i)$ is the number of node i's neighbors that are discovering targets (named *discovering neighbors*) and $\theta(i)$ is the number of node i's neighbors.

DNR denotes the ratio between the number of discovering neighbors and that of all neighbors. Further, when $0 < \eta(i) < 1$, some of node i's neighbors are discovering neighbors, while others not. Then it is reasonable to conclude that i is closed to group target's "boundary". There is a *neighbors table* in each node, where the neighbors' discovery information is stored. According to the Formula (1), the *boundary status* can be defined as:

Definition 2.2 Boundary Status (BS): The Boundary Status is a function χ of node i:

$$\chi : I \to \{INNER,\ BOUNDARY,\ OUTER\}$$
$$\chi(i) = \begin{cases} OUTER, & \eta(\mathrm{i}) \le H_0 \\ BOUNDARY, & H_0 < \eta(\mathrm{i}) < H_1, \text{ for all } i \in I \\ INNER, & \eta(\mathrm{i}) \ge H_1 \end{cases} \quad (2)$$

where H_0 and H_1 are parameters to adjust the width of boundary. The BS describes whether one node is on the boundary of a region which contains a group target.

Definition 2.3 Boundary Detecting Model (BDM): The Boundary Detecting Model (BDM) is a sensing model, in which every sensor can calculate its BS by communicating with its neighbors.

Figure 1 illustrates a snapshot of group target appearing in a region deployed with numerous BDM sensor nodes. The white circle represents OUTER sensor node, gray circle represents BOUNDARY sensor node, black circle represents INNER sensor node and red triangle represents target.

Every sensor is in the OUTER status initially. Once the signal strength of events it captured exceeds a certain threshold, the node turns into discovering status. Meanwhile, it also has the responsibility of notifying its neighbors to update their neighbors' tables. Based on the updated neighbors table, a node could calculate its DNR and decide which BS it will become. There are three different situations: i) if DNR$\leqslant H_0$, it gets into the OUTER status; ii) if DNR$\geqslant H_1$, it gets into the INNER

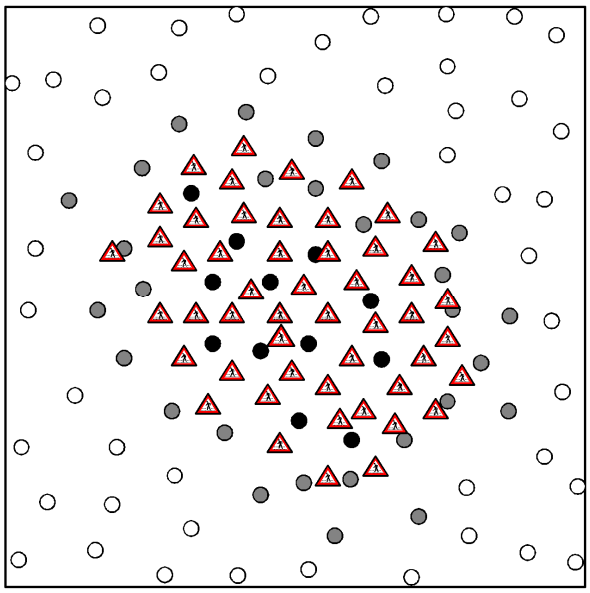

Figure 1. A scene of group target in WSN.

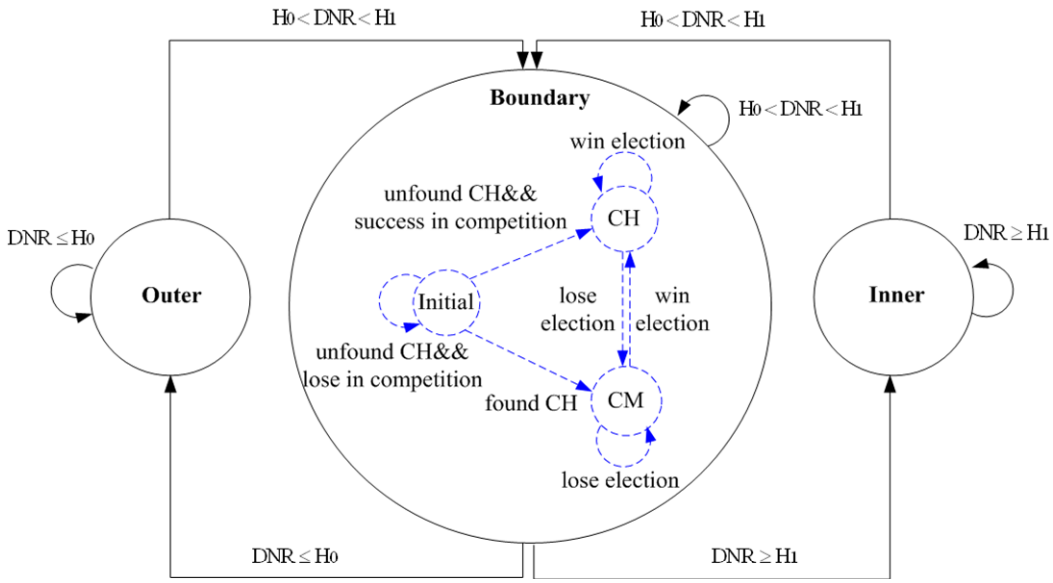

Figure 2. Node's complete status transition diagram.

status; iii) otherwise, it gets into the BOUNDARY status. The whole process is completely distributed and only needs local communications among sensors.

2.2. Boundary Clustering Model

Unlike other clustering methods [1] used in WSN, we only needs the BOUNDARY nodes in the clustering process in our clustering model. We divide BOUNDARY nodes into several clusters without considering the INNER and OUTER nodes. Because the ratio between the number of BOUNDARY nodes and the number of discovering nodes decreases with the group target's scale increasing, the energy consumption of the whole WSN can be reduced accordingly.

At the beginning, all BOUNDARY Nodes stay in the initial status. Then, they get into an election phase. After that, the winner changes into the CH status, while loser gets into the CM status after it chooses one CH as its head.

In this section, we introduced two useful models which are essential to support our algorithm for tracking group target. In Figure 2, complete status transition diagram of a node is shown. The dotted lines represent the *cluster status* transition that runs on each BOUNDARY node, while the solid lines represent the BS transition that runs on every node.

3. Tracking Algorithm Based on Convex Hulls Merging

3.1. Group Target Localization Algorithm

In the previous work [9], a group target is localized as a rough circle. In this paper, our localization result is a convex hull which is more accurate compared with the circle in [9]. However, considering the limitation of the capability of sensors, we cannot get the accurate convex hull contains all the targets in a group. Instead, we can calculate the convex hull enveloped by all BOUNDARY nodes and it approximates the accurate convex hull.

There are some methods [10–12] to calculate the convex hull of a set of vertexes. To reduce the computation complexity and energy consumption, two algorithms are selected carefully and some variations and integrations are made necessarily. One of the convex hull algorithms is known as *Graham-Scan* [10], and the other is an on-line approximate convex hull algorithm [12]. These two algorithms run on the sink and deployed nodes respectively. The main idea about the group target localization algorithm can be illustrated as follows. First, the convex hull algorithm runs on each CH, which collects the information from all its members and is responsible for sending the result (local convex hull) back to the sink. When the sink gathers enough data of local convex hulls, the merging algorithm will be executed to generate the entire convex hull. The complete algorithm can be described as follows, named *Convex Hulls Merging Localization* (*CHML*).

***Algorithm 1*: Convex Hulls Merging Localization**
Divide:1:CH collects position information from its members;
2:executes on-line algorithm to compute local convex hull;
3:sends local convex hull back to a sink;
Merge:4:**IF** (the sink receives sufficient local convex hulls)
Execute merging algorithm to compute global convex hull;
ELSE
Receive local convex hulls.

Considering the dynamic case, when the group target moves the clusters around it will keep changing. If the convex algorithm is re-executed for each change, more computation and communication resources will be consumed. Therefore, it is reasonable to use an on-line algorithm on CHs. In the sink, we use the convex hulls merging algorithm which is a divide-and-conquer algorithm [11] for convex hull. In this paper, we choose an approximation of the on-line algorithm for its simplicity. The error of this algorithm is guaranteed to be very low. The time-complexities of these two algorithms are $O(n)$ and $O(n\log n)$ respectively, where n is the number of BOUNDARY nodes in total. So the total time-complexity of our algorithm is $O(n\log n)$. Refer to appendix for details of the algorithms.

3.2. Cluster Maintenance Algorithm

When a group target is moving, some previously constructed clusters are destroyed and some new ones will be formed. In order to track the group target continuously, clusters must be maintained so that out-of-date clusters are eliminated and new clusters are dynamically created along the trajectory of the group target.

Consider a newly formed cluster. When the group target is moving, some new nodes will join the cluster. Meanwhile, some old ones quit. So the topology of the cluster is changing and the original CH may not be able to continue playing as a cluster head. For instance, some new nodes may be too far away to achieve the communication with the CH. In this case, a new CH will be selected.

Following criterions need to be considered when a new CH is elected. First, the nearer a node is to the center of the cluster, the more suitable it is to become a CH [8]. Therefore, the sum of the distances from the CH to its members will be the smallest one and it will help to reduce the communication consumption and delay. Furthermore, it will ensure the cluster moves in a direction which the group target towards. Second, the convex hull information owned by the original CH is useful to the new CH. For this reason, a node should better be chosen if it could inherit this information easily.

Based on the two criterions, a weighted election method is proposed. Let C_j be a cluster, $C_j \subseteq I$, then for any i, $i \in C_j$, the distance from i to its cluster's topology center is d_i,

$$d_i = \sqrt{(a_i.x - (\frac{1}{|C_j|}\sum_{i \in C_j} a_i.x))^2 + (a_i.y - (\frac{1}{|C_j|}\sum_{i \in C_j} a_i.y))^2} \text{ d10} \quad (3)$$

where a_i is the position of node i. Then the weight of node i is W_i,

$$W_i = w_1 \times d_i + w_2 \times \frac{1}{n+1} \quad (4)$$

where w_1 and w_2 are weights of factors, n is the number of lost nodes if node i is elected as a new CH. So when a new BOUNDARY node participates, or an old one quits, the CH will compute the weight of each node in its cluster. If there is a node whose weight is bigger than that of the previous CH, this node will take the old CH's place and become the new CH.

4. Performance Evaluation

4.1. Simulation Setup

In the simulation, binary sensors are randomly scattered with a uniform distribution in the monitoring region which is a rectangle area with the size of $1000m \times 700m$. The communication radius and sensing radius are changed according to the deployment density of sensors to guarantee enough coverage of the monitoring region.

We simulate a group target moving at the speed of $5m/s$. If there is any target gets into one sensor's sensing area whose sensing radius is R_{sense}, the sensor will discover it without knowing the number of targets or their accurate positions.

4.2. Evaluation Criteria

In a point target tracking evaluation, the localization result is a point and the error can be defined as the distance between the actual position and estimated position. However, the group target localization result is a polygon area. To measure the localization accuracy, we propose the following evaluation criteria. Let polygon P_A be a group target's convex hull and polygon P_E be its estimated convex hull. Then, the error is defined as:

$$e = \frac{S_{P_E \cup P_A} - S_{P_E \cap P_A}}{S_{P_E \cup P_A}} \times 100\% \quad (5)$$

where S represents the area of a polygon. Apparently, $0 \le e \le 1$ is satisfied. This error definition not only reflects the coverage extent of P_E over P_A, but also avoids P_E to be too large. So the lower the e is, the better the performance can be achieved. By varying the number of deployed sensors and their sensing radiuses, the impact of sensor density and sensing radius on the accuracy of estimated convex hull is investigated.

4.3. Simulation Results

Figure 3(a) shows the tracking accuracy during a complete tracking process. It can be seen that, from time 0s to 10s, the accuracy increases sharply. But at the end of the process, the different situations happen. In fact, these variations reflect the process of a group target getting into the monitoring region and that of leaving the area.

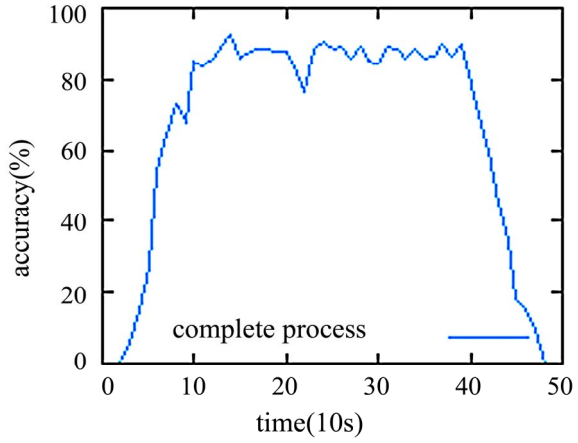

Figure 3(a). Tracking accuracy during a complete tracking process.

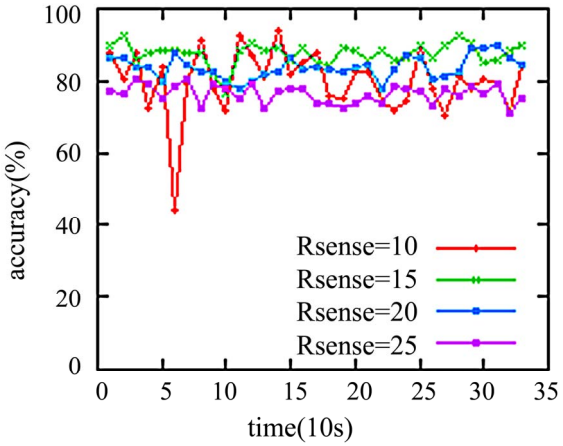

Figure 3(b). Accuracies of group target tracking under different sensor deployment densities.

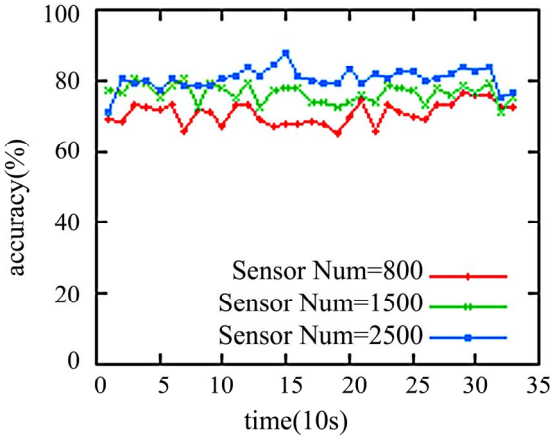

Figure 3(c). Impact of sensors' density on accuracy.

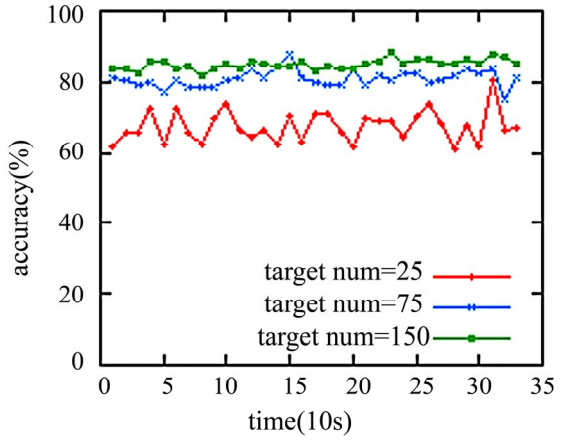

Figure 3(d). Number of targets and tracking accuracy.

In Figure 3(b), accuracies of group target tracking algorithm under different sensor deployment densities are presented. This figure indicates that, when the sensors' density is fixed, the performance gets better as the sensing radius decreases. But this situation changes when R_{sense} =10. At this time, the tracking results are unstable. The reason is that, as sensing radius is decreasing, the degree of network coverage also reduces, which affect the tracking accuracy of our algorithm.

The impact of sensors' density on accuracy is showed in Figure 3(c). To be reasonable, no matter how many sensors are there, the coverage degree should be guaranteed. The simulation results show that the more sensors are deployed, the better performance is. By varying the number of targets in the group target，its impact on accuracy is examined. Figure 3(d) shows that, when the number of targets increases, the accuracy of tracking algorithm improves accordingly.

Figure 4 illustrates a segment of trajectory in a complete tracking process. From this figure, we can find that

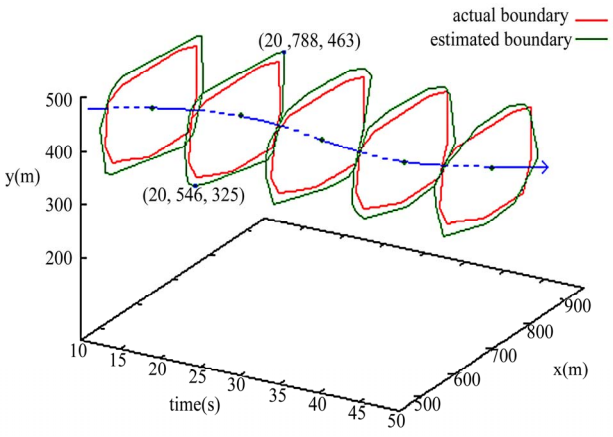

Figure 4. Comparison between actual boundary and estimated one.

the estimated boundary is closed to actual boundary as the group target moving.

5. Conclusions

The problem of group target tracking in wireless sensor networks is fully investigated in this paper. To achieve the group target tracking goal, a flexible group target boundary detecting model is proposed. Moreover, utilizing a computing geometry conception—convex hull, a novel divide-merge group target tracking algorithm is proposed to solve this challenging problem. By analyzing the experimental results from simulations, the group target tracking performance is evaluated. The results show that the proposed algorithm is effective and efficient.

6. References

[1] W. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," IEEE Transactions on Mobile Computing, Vol. 3, No. 3, pp. 258–271, 2004.

[2] W. Zhang and G. Cao. "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks," IEEE Transactions on Wireless Communications, Vol. 3, No. 5, pp. 1689–1701, 2004.

[3] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," ACM Sensys'03, Los Angeles, California, USA, pp. 150–161, 2003.

[4] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley, "Tracking multiple targets using binary proximity sensors," In Proceedings of the 6th International Conference on Information Processing In Sensor Networks, April 2007.

[5] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri. "Target tracking with binary proximity sensors: Fundamental limits, minimal descirptions, and algorithms," In Proceedings of ACM SenSys, 2006.

[6] X. Zhu, R. Sarkar, J. Gao, and J. S. B. Mitchell, "Light-weight contour tracking in wireless sensor networks," In Proceedings of IEEE INFOCOM, pp. 1849–1857, 2008.

[7] K. P. Shih, S. S. Wang, P. H. Yang, and C. C. Chang, "CollECT: Collaborative event detection and tracking in wireless heterogeneous sensor networks," In Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06).

[8] X. Ji, H. Zha, John J. Metzner, and G. Kesidis, "Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks," IEEE International Conference on Communications, Paris, FRANCE, June 20–24, 2004.

[9] D. Cao, B. Jin, and J. Cao, "On group target tracking with binary sensor networks," In Proceedings of the 5th IEEE International Conference on Mobile Ad Hoc and Sensor System (MASS), pp. 334–339, 2008.

[10] O' Rourke and Joseph, "Computational geometry in C," China Machine Press, 2005.

[11] T. H. Cormen, C. E. Leiseron, and R. L. Rivest. "Introduction to algorithms," The MIT Press, pp. 947–956, 2002.

[12] R. C. T. Lee, S. S. Tseng, and R. C. Chang, "Introduction to the design and analysis of algorithms," Addison-Wesley, Chapter 12.6, 2002.

Appendix

I. Dynamic approximation convex hull algorithm:

Input: A sequence of points with operators $(p_1, operator_1), (p_2, operator_2), \cdots$, and the number m of pairs of parallel lines being used.

Output: A sequence of approximate convex hulls $a_1, a_2, \cdots$, where a_i is covering the reserved points $p_1, p_2, \cdots, p_l$, where $l \le i$.

Initialization: Construct m pairs of parallel lines with slopes 0, $\tan(\pi/m), \tan(2\pi/m), \cdots, \tan((m-1)\pi/m)$ respectively and locate all lines so that they all intersect at the first input point p_1. Set i=1, i.e. the current input point with operator is $(p_1, insert)$.

Step 1. **IF** (p_i is between each of the m pairs of parallel lines)

IF (operator=delete**)** delete the point P_i, $l = l - 1$, Stop;

ELSE insert the point P_i, $l = l + 1$, Stop;

ELSE

IF (operator=delete**)** delete the point P_i, $l = l - 1$;

ELSE insert the point P_i, $l = l + 1$, $p' = p_i$;

Step 2. Move the line nearest to point p', without changing its slope, to touch p' and associate p' with this line.

Step 3. Construct an approximate convex hull by connecting the 2m points with respect to each line in the clockwise fashion and denote this approximate convex hull as a_j.

Step 4. If no other point will be input, then stop; otherwise, set i=i+1 and receive the next input point as P_i. Go to Step 1.

II. Graham-Scan(Q) // Algorithm which will be run on the sink

Step 1.let p_0 be the point in Q with the minimum y-coordinate, or the leftmost such point in case of a tie;

Step 2. let $\langle p_1, p_2, \cdots, p_m \rangle$ be the remaining points in Q, sorted by polar angle in counterclockwise order around p_0 (if more than one point has the same angle, remove all but the one that is farthest from p_0);

Step 3. PUSH(p_0 ,S);

Step 4. PUSH(p_1 ,S);

Step 5. PUSH(p_2 ,S**)**;

Step 6. for(i=3, i<=m, i++)

Step 7. {while(the angle formed by points NEXT-TO-TOP(S), TOP(S), and P_i makes a nonleft turn)

Step 8. POS(S);

Step 9. PUSH(P_i, S); }

Step 10. return S;

