ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **15**, Number 4, 2012, 319–334

Maintaining a Minimum Spanning Tree for Kinetic Autonomous Robots in 2D-Euclidean Plane

Chintan MANDAL, Anil Kumar SAHU, Suneeta AGARWAL

Department of Computer Science and Engineering,
Motilal Nehru National Institute of Technology, Allahabad, India
E-mail: chintanmandal@gmail.com, anilkumar0505@gmail.com, suneeta@mnnit.ac.in

Abstract. This paper presents a procedure for maintaining a 2D-Euclidean Minimum Spanning Tree for a set of n autonomous kinetic robots having no central supervision. The proposed procedure is based on the Kinetic data structure framework and the well known fact that the edges of the minimum spanning tree for a given set of points in the 2D-Euclidean domain are contained in the edges of its Delaunay triangulation. The kinetic data structure framework has a centralized data structure along with a priority queue on which a proposed algorithm works to maintain a combinatorial geometrical structure for a set of geometric objects. In this work, we propose an approach where the computation of the geometrical structure is done through the geometrical objects, in our case, the computations being done by the robots themselves. This is unlike that of maintaining a centralized data structure in the kinetic data structure framework. The kinetic autonomous robots, each being a computing object do all computations by themselves and update the spanning tree as the tree changes. In terms of kinetic data structure metrics, our data structure is local and compact.

Key words: Delaunay triangulation, Kinetic data structures, Euclidean minimum spanning trees.

1. Introduction

A team of autonomous, homogeneous and synchronized robots are moving independently in the 2D-Euclidean plane. They all have the same communication range,

battery life, processing capability and memory. They act both as "sensors" and "routers" for communicating among themselves. Such robots can be used to perform tasks such as space exploration, cleaning floors, and search and rescue operations in unknown and hazardous areas. An important criterion for performing a successful job is communication among each other. Communication can be done by broadcasting or routing messages through other robots. An Euclidean minimum spanning tree (EMST) can serve the purpose for communication among the robots for routing. For a connected and undirected graph *G(V,E)* with positive edge weights, a minimum spanning tree (MST) is an acyclic sub graph of G which connects all the vertices / static robots in G such that the total edge weight of the tree is minimum. The EMST is a special case of MST where the Euclidean distance between the vertices are taken as edge weight and the vertices are coordinates in the Euclidean domain. However, as the robots are in constant motion, the EMST changes. In this work, we give a methodology to maintain and update the EMST at discrete times for a given set of kinetic autonomous robots. In this paper, we use the terms robots and points interchangeably.

Very few literatures are available for kinetic EMST in the 2D-Euclidean domain. Fu and Lee [6] proposed an algorithm which requires $O(kn^4 \log\ n)$ preprocessing time for n moving points and k is the maximum degree of the motion function of the points. The algorithm requires *O(m)* space, where m is the maximum number of combinatorial changes of the EMST from time $t = 0$ to $t = \infty$. After preprocessing, the EMST can be calculated at any given time t in *O(n)* time. Agarwal et. al. [1] also proposed an algorithm for calculating Kinetic EMST, in which the edge weights changes with linear function of time. Their algorithm runs in $O(n^{2/3}\log^{4/3}n)$ per combinatorial change and also supports insertion and deletion of edges. Basch *et al.*[4], Rahmati and Zarei [12] individually proposed data structures to maintain a kinetic EMST based on the kinetic data structures (KDS) [2].

All the above algorithms can be envisioned as the KDS having a centrally controlling unit, which maintains a single data structure for maintaining the EMST and a priority queue. Changes in the combinatorial structure are reflected in the single data structure. This is unsuitable for autonomous robots maintaining an EMST. For any change in a data structure in a single robot, it has to be communicated to all the robots to avoid inconsistency. Thus, one can think of decentralizing the maintenance of the EMST through each robot. To the best of our knowledge, no algorithm exists for maintaining an EMST among autonomous robots, all computations done by the autonomous robots themselves. We propose a methodology based on the KDS framework which was proposed by Basch and Guibas [3, 2] to maintain the EMST for the moving autonomous robots, where the changes in the EMST are computed by the robots themselves.

The rest of the paper is organized as follows: section 2 gives definitions and properties of Delaunay triangulation along with a short on KDS; section 3 discusses the proposed approach and methodology for maintaining the EMST among the robots; section 4 gives an analytical study of the proposed algorithm with an experimental result of the EMST obtained due to the kinetic robots and the exact EMST obtained. We conclude by with stating some challenges in the problem in section 5.

2. Related Theory

Our algorithm is based on the well known fact that the EMST is contained in the edges of the Delaunay triangulation (DT) for a set of static points in the 2D-Euclidean plane [9]. In this section, we give the definition of a DT along with its important properties. Later in the section, a short introduction to the KDS is also given.

Definition 1 (Delaunay triangulation [9]). A collection of triangles, $\triangle$, on a set of points $Pt = \{p_1, \ldots, p_n\}$ is *Delaunay*(*DT(Pt)*) if and only if no point of Pt is interior to any circum-circle of a triangle in $\triangle$.

Let $\triangle(G) = (Pt, E)$ be a planar straight line graph representing a triangulation of the set of points $Pt = \{p_1, \ldots, p_n\}$ and E be the set of edges. $\triangle_e \in \triangle(G)$ is a triangle defined by the two end points of the edge, $e \in E$ and the third point opposite it. In Fig. 1a, triangulation *triangulation(ABCDE)* is not a DT as points B and C are contained in *circum-circle*($\triangle$ ADE) violating the circum-circle criteria for DT. However, Fig. 1b shows a triangulation of the same set of points, *triangulation(ABCDE)*, which is a DT, where circum-circle of the triangles of the triangulation does not contain any other points of the triangulation other than the three points on its circumference. A non-DT (Fig. 1a) can be transformed to a DT (Fig. 1b) by deleting $e \in E$, the diagonal of the quadrilateral formed by the end vertices of the edge, e and the vertices inside the circum-circle(s) formed by $\triangle_e$ and inserting a new edge joining the vertices inside the circum-circle. In Fig. 1a, AD is deleted in the quadrilateral $ABDE$ and a new edge, BE is inserted (Fig. 1b). This procedure of deleting an edge and inserting another edge in case of failure of the circum-circle criteria of DT will be referred as a "*flip*". It is to be noted that the outer boundary of the triangulation forms the convex hull of the set of points Pt.

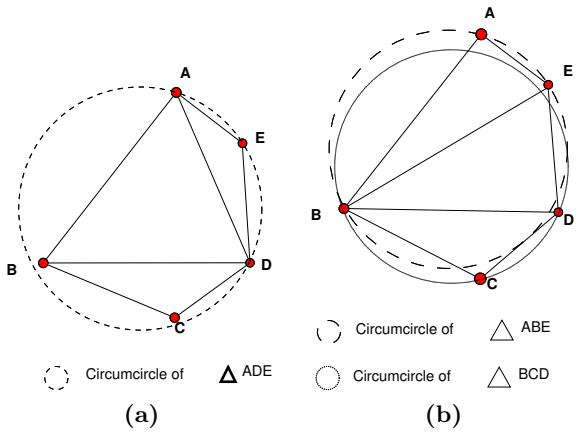


Fig. 1. (a) Triangulation $\triangle(ACBDE)$ is not a DT as circum-circle$\triangle ADE$ contains B and C; (b) Triangulation $\triangle(ACBDE)$ is a DT as portion of the circum-circle$\triangle ABE$ or circum-circle$\triangle BDE$ each does not contain any other points of the triangulation.

We give a short note on KDS [2], which we will use to maintain the combinatorial structure of the DT and the EMST among the points in motion in the 2D-Euclidean space. The KDS is an event based framework which uses temporal coherence among the involved points defined as $P = (P_{x_t},\ P_{y_t})$, each moving with a motion function. It uses data structure(s) along with a priority queue for maintaining an attribute, e.g. the convex hull [7], DT [8] for some geometric objects like points, lines in the Euclidean domain. For maintaining an attribute, the geometric relations between the objects should be valid. These collections of the relations for an object contribute to their individual *certificates*. A failure of a *certificate* results in an *event* [2].

The *certificates* necessary for maintaining a DT are the convex hull of the kinetic robots and the other for maintaining the property that no other robot lies inside a circum-circle of the triangle formed by three robots in the triangulation. The latter certificate is the relation between the three robots of a triangle in the triangulation and the opposite robot of each edge of the triangle (Fig. 1a) defined as:

$$InCircleCertificate(\triangle\ A_tB_tC_t, D_t) = \begin{vmatrix} A_{x_t} & A_{y_t} & A_{x_t}^2 + A_{y_t}^2 & 1 \\ B_{x_t} & B_{y_t} & B_{x_t}^2 + B_{y_t}^2 & 1 \\ C_{x_t} & C_{y_t} & C_{x_t}^2 + C_{y_t}^2 & 1 \\ D_{x_t} & D_{y_t} & D_{x_t}^2 + D_{y_t}^2 & 1 \end{vmatrix}. \qquad (1)$$

In equation 1, $\triangle\ A_tB_tC_t$ is the triangle formed by A_t, B_t, C_t at time t and D_t being a point opposite to one of the edges of the triangle $\triangle\ A_tB_tC_t$. *InCircleCertificate(*$\triangle\ A_tB_tC_t,\ D_t$*)* fails if the value of the determinant is greater than 0. In Fig. 1a, the *InCircleCertificate()*-s are *InCircleCertificate(*$\triangle\ ADE$*,B)*, *InCircleCertificate(*$\triangle\ BCD,\ A$*)*, *InCircleCertificate(*$\triangle\ ABD,\ C$*)* and *InCircleCertificate(*$\triangle\ ABD,\ E$*)*.

The positive roots of the solution of the equation:

$$InCircleCertificate(\triangle\ A_tB_tC_t, D_t) = 0$$

give the failure time of the *certificate* and its corresponding event time for the failure of the Delaunay criteria. The failure event generates a flip of the edge AD with BE. New certificates are calculated with the new *InCircleCertificate(*$\triangle\ ABE,\ D$*)* and *InCircleCertificate(*$\triangle\ BDE,\ E$*)* for the new edge BE. The positive roots of the solution of the *InCircleCertificate()*-s gives the time at which the certificates will fail for BE. All the events are pushed inside a priority queue known as the *event queue*. At each failure time, new *certificates* are recalculated for the new geometric bodies related with the event and the priority queue is updated.

3. The Proposed Approach

Our approach is based on the well known result that the edges for the EMST are the subset of edges in the DT for static points. The DT for a given set of static points can be obtained in $O(n \log n)$ time while the EMST can be obtained in $O(n \log n)$ time by applying Kruskal's [10] or Prim's [11] algorithm over the DT. Based on this well known result, we give a methodology for maintaining a kinetic EMST while

maintaining the DT for a given set of moving robots. All EMST edges of the DT will be referred to as *branches* and the other edges as *chords*. To maintain the DT we use the KDS to track the changes in DT *i.e.*, kinetic Delaunay triangulation (KDT) proposed by Guibas *et al.* [8].

Let $Pt = \{p_1,\ p_2,\ \ldots,\ p_n\}$ be the set of n robots moving in a 2D-Euclidean plane having no obstructions and collisions among themselves. A collision free environment is necessary for the KDT according to the assumptions of KDS. We assume the robots to move in a rectangular boundary bounded by four static dummy robots placed at infinity. In real life, the four static robots can be considered to be placed at large distances, such that all the dynamic robots are contained in the rectangle defined by the four static robots. Thus, the convex hull of the robots will always be this rectangular boundary. The communication cost between the robots is assumed to be constant irrespective of the distance between them and is less than the consecutive difference of failure time of any event. Each robot $p_i \in Pt$ is represented by its motion equation:

$$p_i = (x_i\ +\ v_x t,\ y_i\ +\ v_y t), \tag{2}$$

where (x_i, y_i) be the coordinates of the robots at time t_i (the position when a certificate fails) and (v_x, v_y) gives the velocity along `X` and `Y` axis. We want to maintain a EMST for the kinetic robots in set Pt while maintaining the DT. At t_0 (when the robots were static at the beginning), we form a DT with the given robot set Pt, $DT(Pt) \equiv G(Pt, E)$, where *G(Pt,E)* is the DT graph having Delaunay edges E. The EMST is obtained using the Kruskal's / Prim's algorithm on *G(Pt,E)*. With the continuous change of time, the topology of the DT has to be maintained. The DT gets modified when the circum-circle criteria of a triangle of the triangulation fails. We state here the situations when the EMST or the topology of the DT changes.

The topology of the DT changes when there is a flip of an edge. If the flipped edge is a *chord*, then it will be referred to as a *Normal Flip Event*. This results in no change of the EMST but only a change in the topology of the DT. If the flipped edge is a *branch*, then the current tree is broken into a forest of two trees. This will be referred to as the *Effective Flip Event*. To reconnect the forest, due to minimal communication ranges and avoiding broadcasting of the messages, a shorter edge is chosen among the *chords* of the **quadrilateral(ABCD)**, *i.e.* the quadrilateral in which the edge has been flipped. Thus, an *Effective Flip Event* changes the topology of the DT and the EMST as well.

It is possible that the topology of the DT does not change for a period of time due to no flips. However, due to the motion of the robots, there will be a change in the length of the *branches* and *chords*. To minimize the difference between the EMST due to the kinetic robots and the exact EMST, we introduce a parameter, the *Stretch factor*, λ $(0 < \lambda < 1)$, which form the *Stretch certificates*. The failure of a *Stretch certificate* indicates the event time when the length of an edge becomes $(1 + \lambda)$ times the length of the edge at t_i:

$$length(E_i)_{t_{curr}} \leq (1 + \lambda) length(E_i)_{t_j} \texttt{ at } t_{curr};\ E_i \in E, \tag{3}$$

where $length(E_i)_{t_j}$ and $length(E_i)_{t_{curr}}$ are the squared edge length of E_i at the previous failure time, t_j and the current time, t_{curr} respectively. It checks if the

length of the *branch* is within a $(1+\lambda)$ factor of its length at t_{curr}. If it exceeds the said length, the state of the erring *branch* is changed to a *chord*. This results in the EMST to be broken. A new *branch* is chosen among the *chord(s)* of the quadrilateral in which the erring edge was a *branch*. The change of the states of the edges will be referred as a *switch*. The *Stretch event* does not change the topology of the DT.

In both *Effective Flip Event* and the *Stretch Event*, the present EMST gets broken into a forest and new a *chord* is chosen to rejoin the forest. A *chord* is chosen in the quadrilateral in which the *branch* has been flipped or the state of the *branch* is changed. The *chord* is chosen such that a cycle is not formed in the DT to join the forest of the two trees. The validity of the *chord* chosen to be a connecting *branch* is done by checking for a formation of a cycle due to it being added in the tree by communicating through the robots in a breadth first way along the robots in the *branches*. This requires a $O(\log n)$ time for checking cycles.

3.1. Data Structure for communication

In this section we discuss how the robots keep information about the DT and edges of the DT such that they can communicate. All robots keep information about their adjacent robots, if an edge exists between them in the DT and the status of the edges. The status of the edges of the DT can be a ***branch* (1)** or a ***chord* (0)**. They also store the failure time of each certificate, *i.e.* (i) failure time for the Delaunay edges that are incident to it (calculated from the *InCircleCertificate()*) and (ii) failure time for its stretchable limit (λ) for every *branch* that is incident to it (calculated from the *StretchCertificate()* of the *branches*). All the failure times of the events are kept in each robot in a sorted order according to the time which will occur first.

Table 1. Initial information at each robot **A, B, C and D** at t_0

Robot	List	Event Queue Q
A	$(\{B,\{C,D\};1)$, $(C,\{B,G\};1)$, $(D,\{B,H\};0)$, $(H,\{D,G\};1)$, $(G,\{C,H\};1)$	$\{t_1[E],B\}$, $\{t_2[E],C\}$, $\{t_3[F],D\}$, $\{t_4[S],B\}$, $\{t_5[E],G\}$, $\{t_6[E],H\}$, $\{t_7[S],C\}$, $\{t_8[E],H\}$, $\{t_9[S],G\}$
B	$(A,\{C,D\};1)$, $(D,\{A,E\};1)$, $(C,\{A,F\};0)$, $(E,\{D,F\};1)$ $(F,\{C,E\};0)$	$\{t_1[E],A\}$, $\{t_{10}[E],D\}$, $\{t_{11}[S],D\}$, $\{t_{12}[F],C\}$, $\{t_{13}[S],A\}$, $\{t_{14}[S],E\}$, $\{t_{15}[E],E\}$, $\{t_{16}[F],F\}$
C	$(A,\{B,G\};1)$, $(B,\{A,F\};0)$ $(G,\{A,\infty\};0)$, $(F,\{B,\infty\};0)$	$\{t_2[E],A\}$, $\{t_{12}[F],B\}$ $\{t_7[S],A\}$, $\{t_{17}[F],F\}$ $\{t_{18}[F],G\}$
D	$(A,\{B,H\};0)$, $(B,\{A,E\};1)$ $(H,\{A,\infty\};0)$, $(E,\{B,\infty\};0)$	$\{t_{19}[F],H\}$, $\{t_{10}[E],B\}$, $\{t_3[F],A\}$, $\{t_{11}[S],B\}$, $\{t_{20}[F],E\}$

At the time of initialization, a DT is constructed and an EMST is obtained by a central system using Kruskal's [10] / Prim's [11] algorithm with the given robots.

The central system distributes the failure time calculated from the certificates of the respective edges to its end robots. Figure 2 shows an initial DT for eight robots. After the robots have been updated, they are left on their own with their own motion. As the robots go into motion, communication is done between themselves.

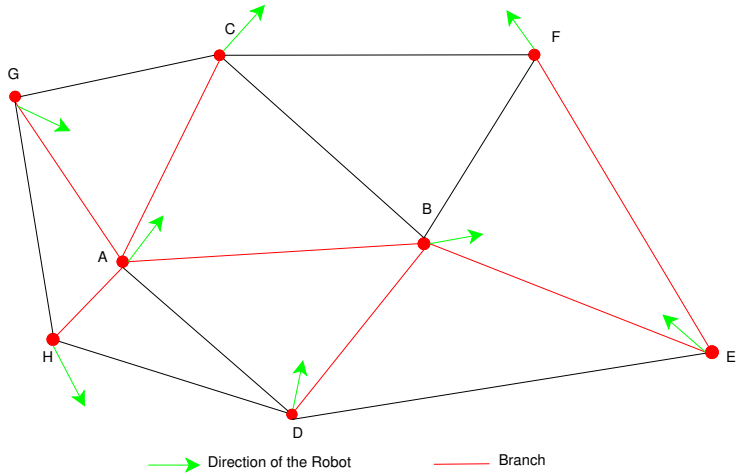


Fig. 2. Initial Delaunay triangulation at t_0

We give below the details of the data structure in each robot to maintain communication between each other:

1. A *List* of nodes indexed on the basis of label of the adjacent robots and each node contains
   (a) Label of the adjacent robots if they are connected to it by edges of the DT.
   (b) A set of labels of the two end robots (represented within {}) which may form an edge as a result of flipping joining the robot and an adjacent robot.
   (c) Edge type - 1 (*branch*) or 0 (*chord*) in the DT joining the robot and the adjacent robot.
2. An *Event Queue*: each entry in queue contains
   (a) Failure time (t_i) of each certificate, related to the edge incident on this robot, sorted from the earliest to the farthest.
   (b) Type of the event (Effective Flip [E] / Normal Flip [F] / Stretch [S]) associated with each t_i.
   (c) All certificates are related with the edges of the DT. An edge is composed of two robots on either side. The index to the other robot is the other information kept in the robot for each failure time of a certificate.

The *List* in each robot contains the details of edges incident on each robot along the Delaunay edges. The *Event Queue (Q)* is a prioritized according to the earliest

time in which an edge incident to it will fail. This helps is identifying the earliest event which will occur for a robot to handle.

Figure 2 shows the DT for a set of robots- {**A, B, C, D, E, F, G, H**} at t_0, each having a direction of its own. All future failure times from certificates are calculated by both the end robots of an edge represented in the *List*. Table 1 gives the *List* and *Event queue (**Q**)* stored in the robots **A, B, C** and **D** at t_0. A *List* entry of **A** as $(\{B,\ \{C, D\};\ 1)$ indicates that **A** is connected to **B** with a *branch* (1) and the other two robots on either side of the *branch* are **C** and **D**. An *Event queue* entry represented as $\{t_1[E], B\}$ means that at t_1, the certificate to be changed is related with the edge connecting **B** and the edge flip will be *Normal Flip*. The *InCircleCertificate()* for edge **AB** will be calculated by both robots **A** and **B**.

3.2. Handling Events

An event occurs when a certificate fails, *i.e.* the time at which an event occurs. Events are classified as (a) *internal* when the combinatorial structure of the main attribute do not change but its certificates need to be changed (*e.g.* the EMST obtained from a DT do not change but the DT itself changes) and (b) *external* when the combinatorial structure and its certificates also change. An *event* can be a flip of an edge (a *branch* or a *chord*) in the DT or even a *switch* of a *branch* when its stretch limit occurs. Each event is processed by deleting the event from the queue and the list that is maintained by the adjacent robots.

There are three events — *Normal Flip Event*(F), *Effective Flip Event*(E), and *Stretch Event*(S). The *Effective Flip Event* is an extension of the *Normal Flip Event*, where a potential *branch* is searched for joining the forest before the flip event takes place. After processing each event, new failure times from certificates for the new objects are calculated.

3.2.1.Normal Flip Event

A *Normal Flip Event* takes place when the *InCircleCertificate()* fails for a triangle and a *chord* is flipped. This results in a change of topology of the DT but not the EMST. This is an *internal* event, as there is no combinatorial change in the EMST, but only changes in the certificates of the DT.

The following operations are performed to handle this event

1. The node, which is pointed by the failed event, is deleted from the list of both the adjacent robots.

2. Two new triangles are created as a result of a flip. The entries in the set are updated by changing the label in the node of the end robots of the edge that is to be flipped.

3. One node is added to the list of end robots of the flipped edge.

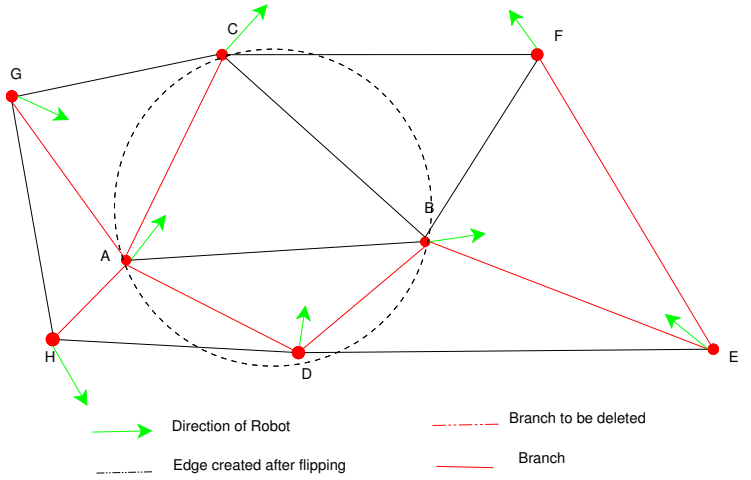
At t_1, in *triangulation(ABCDEFG)* (Fig. 3a), the certificate of edge AB fails as an adjacent robot (**D**) is contained in the circum-circle of the triangle $\triangle$**ABC**. Seeking

in robot **B**, the only entry related with **A** in its *List* is $(A, \{C, D\};\ 1)$. Thus when the *Normal Flip* event (Fig. 3b) happens at t_1, the following happens:

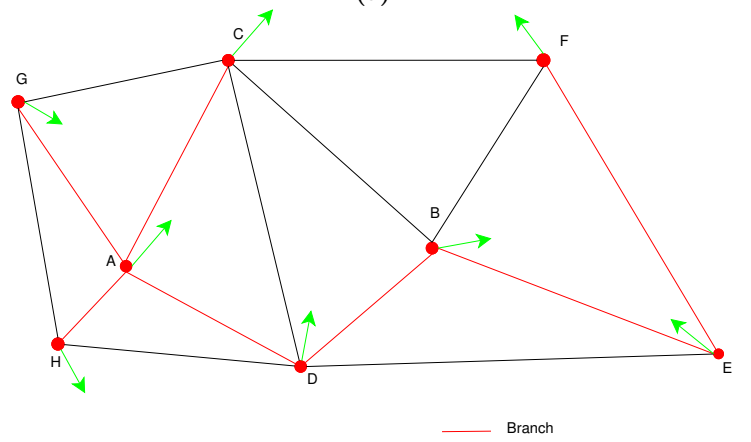
(a) The nodes whose index is stored in the event having fail time t_1, is deleted from the list of both the end robots.

(b) The entries **{B,G}**, **{B,H}** and **{A,E}**, **{A,F}** for future edge in the nodes, which are indexed by the **C** and **D** in the list of **A** and **B** are changed to **{D,G}**, **{C,H}** and **{C,E}**, **{D,F}** in **A** and **B** respectively.

(c) One node is added to the list **C** and **D** which represent the edge **CD** and contains the labels of the future edge *i.e.* **AB**.



(a)



(b)

Fig. 3. *Normal Flip Event*: (a) Failure of Delaunay criteria; (b) Edge **AB** flipped with **CD**.

Table 2 shows the changes in the *List* and *Event queue* after t_1 of the different robots, when there has been a *Normal Flip* of **AB** to **CD**.

Table 2. Information at each robot **A, B, C and D** at t_1 after update of the *List* and *EventQueue*

Robot	List	Event Queue Q
A	$(\{C,\{D,G\};1)$, $(D,\{C,H\};0)$, $(H,\{D,G\};1)$, $(G,\{C,H\};1)$, $(G,\{C,H\};1)$	$\{t_{21}[E],C\}$, $\{t_{22}[F],D\}$, $\{t_5[F],G\}$, $\{t_6[E],H\}$, $\{t_7[S],C\}$, $\{t_8[E],H\}$, $\{t_9[S],G\}$
B	$(D,\{C,E\};1)$, $(C,\{D,F\};0)$, $(E,\{D,F\};1)$, $(F,\{C,E\};0)$	$\{t_{23}[E],D\}$, $\{t_{11}[S],D\}$, $\{t_{24}[F],C\}$, $\{t_{14}[S],E\}$, $\{t_{15}[E],E\}$, $\{t_{16}[F],F\}$
C	$(A,\{B,G\};1)$, $(B,\{A,F\};0)$ $(G,\{A,\infty\};0)$, $(F,\{B,\infty\};0)$	$\{t_2[E],A\}$, $\{t_{12}[F],B\}$ $\{t_7[S],A\}$, $\{t_{17}[F],F\}$ $\{t_{18}[F],G\}$
D	$(A,\{B,H\};0)$, $(B,\{A,E\};1)$ $(H,\{A,\infty\};0)$, $(E,\{B,\infty\};0)$	$\{t_{19}[F],H\}$, $\{t_{10}[E],B\}$, $\{t_3[F],A\}$, $\{t_{11}[S],B\}$, $\{t_{19}[F],E\}$

3.2.2. Effective Flip Event

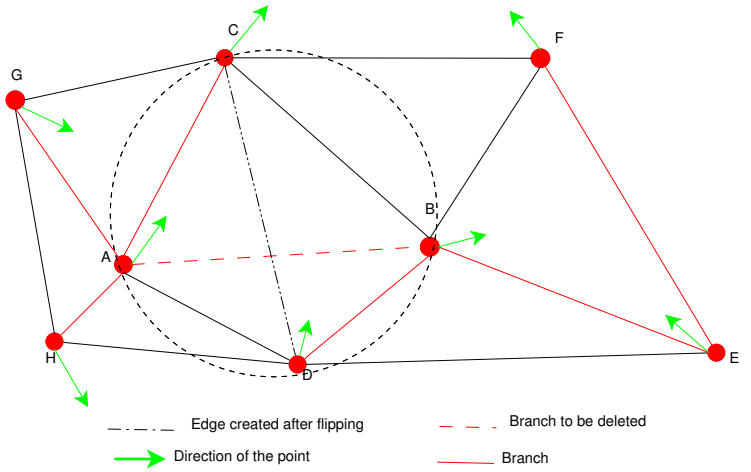
An *Effective Flip Event* occurs when a *branch* is flipped. This event requires an update in the EMST. The *Effective Flip Event* is an extension of the *Normal Flip event*. Before flipping a *branch*, a potential *branch* is searched among the *chords* of the quadrilateral in which the flip is to take place, such that the forest of the disconnected trees can be joined and the EMST is updated. To find the potential *chord*, both robots checks the distance to the other two robots present in the same quadrilateral. Then, the robot that is at minimum distance and will not create a cycle is marked as the new *branch* and its entry is updated to 1. A cycle is detected by following a breadth first search through the present EMST using the potential *branch*. The checking for the cycle requires $O(\log n)$ time for n robots. This is an *external* event, as there is a combinatorial change in the EMST, but only changes in the certificates.

We elaborate *Effective Flip Event* with the following example:

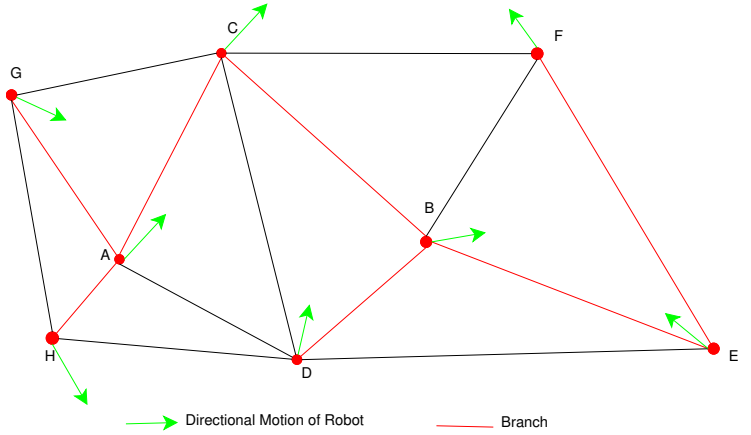
(a) In Fig. 4a, when the branch **AB** is flipped with **CD**, the EMST is broken into a forest. Robots **A** and **B** check distances from **C** and **D** and a least cost edge is chosen which does not make a cycle.

(b) **AC** and **BD** are already *branches*. Thus, a choice exists between *chords* **{BC,AD}** and if `distance`(**BC**) $<$ `distance`(**AD**), **B** updates its node representing edge **BC** as branch (Fig. 4b) *i.e.* changes the node value in the List from 0 to 1.

(c) A new *Stretch Certificate* for edge **BC** is calculated. A cycle is detected by following a breadth first search through the present EMST present.



(a)



(b)

Fig 4. *Effective Flip Event*: (a) Failure of Delaunay criteria; (b) Flipping of **AB** with **CD** and updation of EMST with **BC**.

3.2.3. Stretch Event

We refer failure of the *Stretch Certificate*(*StretchCertificate()*) as a *Stretch Event*. This event changes the EMST. This event is handled if the stretch factor of the branch exceeds the length at the time of the last failure by $1 + \lambda$. In this case, the status of the branch is changed to a *chord*. This change reflects on the EMST to break it into a forest of two trees. To find the potential *chord* to join the tree, the incident edges on the robots of the edge are checked to join the forest. This is again checked similarly as done in *Effective flip event*. This is also an *external* event.

We elaborate the *Stretch Event* with the following example:

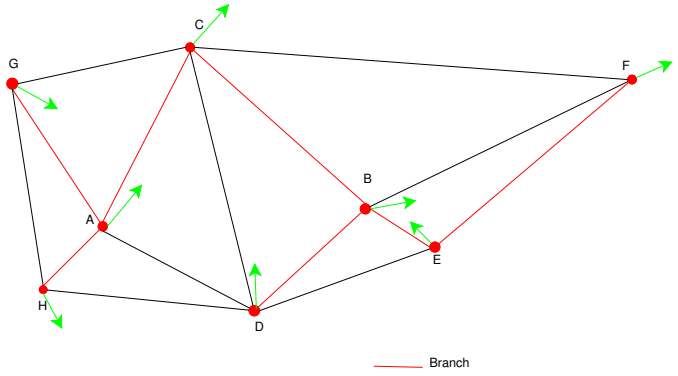
(a) In Fig. 5a, suppose at some time t_i, stretch certificate of edge **BC** fails. The

status of the edge is changed from **1** to **0**. This makes the tree disconnected to a forest of two trees.

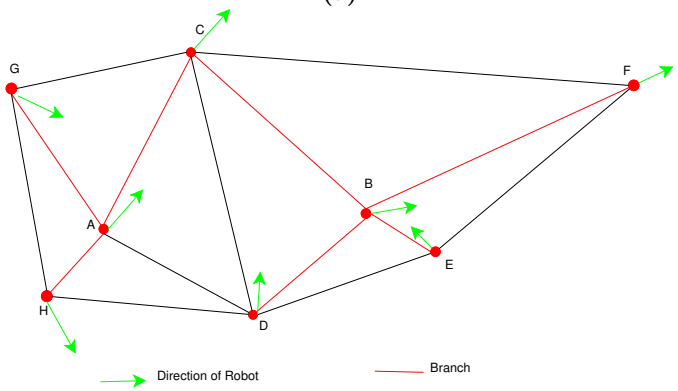
(b) Robots **B** and **C** check the distances to all the other robots joined by *chords* to each of them (**C** and **F** for robot **B**; **D, F, B** and **G** for robot **C**). The length of the chords are communicated to one of the robots (in this case the lower indexed robot, **B**), which checks for existence of cycles with the chords. The status of the shortest chord not forming a cycle and joining the forest is changed from **0** to **1** (Figure 5b).

(c) The cycle is detected by doing breadth first search along the branches of the trees, which requires $O(\log n)$ time.

(d) The *Stretch Certificate* is again calculated for the new branch according to the length of the at t_{curr} [Equation 3]. In our example, as no other edges are shorter than **BC**, the status of **BC** is again changed from **0** to **1**.



(a)



(b)

Fig. 5. *Stretch Event*: (a) Failure of stretch factor of **EF**; (b) switch of **EF** with **BF**.

4. Analysis of the Methodology

The initial EMST from DT can be constructed in $O(n \log n)$. Thus the pre-processing time requires $O(n \log n)$ time. All robots can calculate the certificates (*InCircleCertificate()* and *StretchCertificate()*) related to it $O(1)$ time.

In accordance to KDS metrics, we give the analysis as:

1. The total number of edges in DT is in $O(n)$. So, the total number of certificates is also in *O(n)* and each point participates in average, i.e. $O(1)$ number of certificates so, it is local and compact.

2. To process *Normal Flip Event* and an *Effective Flip Event* it takes *O(1)*, but it takes a *O(n)* time required to update the *Event Queue* and *List* in all the robots. However, an average number of six robots are required to be updated by each robot along an edge for any *event* [5].

3. A cycle is validated by adding a chord as a branch and searching for the cycle along the robots of the existing branches in the DT using a breadth first search taking $O(\log n)$ time.

4. It is not possible to give an upper bound about the number of *Stretch Events* that turns out to be external i.e. requires update in EMST.

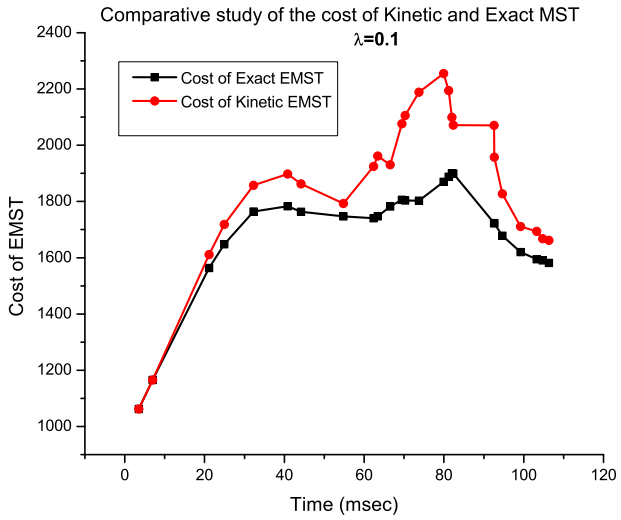
Our EMST is not exact, as the chord chosen for reconnecting the forest is among the chords of the quadrilateral of which a branch has been flipped or from chords of adjacent triangles of the edge which has failed the *Stretch Certificate*. The chord chosen will not be the global minimum chord joining the forests.

We have done simulations for numerous points moving with linear velocity for various stretch factors. A comparative study has been done between the EMST obtained due to the changes in the DT for the kinetic robots and the EMST obtained due to the same robots in the static position from the DT. The exact EMST is calculated using the Kruskal's or Prim's algorithm in the present DT. We give here the comparative graphs for the exact EMST and EMST obtained by updating due to the kinetic robots for $\lambda = 0.1$ (Fig. 6a), 0.5 (Fig. 6b) and 0.9 (Fig. 6c) for time $t = 0$ `to` $t = 115$ `secs` (Fig. 6).

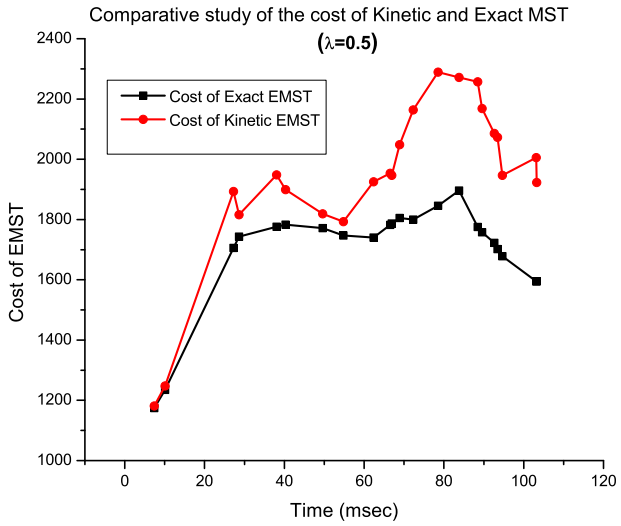
5. Conclusion

We have proposed a methodology based on the KDS to maintain an EMST for autonomous robots in motion in the 2D-Euclidean plane. These could be used in hazardous areas, where a centralized system will be unable to be maintained. Unlike KDS, which uses a central data structure for maintaining the geometrical attributes and a queue for the certificate failure time, we do not maintain it as such. The certificate failure times are distributed among the robots, enabling the EMST to be maintained by the robots themselves. This distribution of the certificate failure times and the updation of the KDS *List* and *Event Queue* among the robots make the

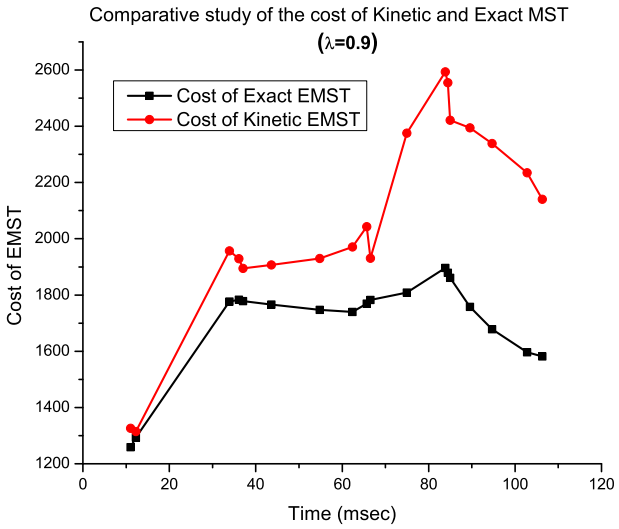
KDS to be not responsive. A KDS is responsive if it is a *O(polylog)* algorithm, i.e. it is a polynomial function of logarithm in n. A longer communication time between robots than the time difference between the consecutive failure times of the certificates ensures that the updation of the robots do not be inconsistent.



(a)



(b)



(c)

Fig. 6. Comparison of kinetic EMST and exact EMST for stretch factors (λ): (a) $\lambda = 0.1$; (b) $\lambda = 0.5$; (c) $\lambda = 0.9$.

References

[1] Agarwal P.K., Eppstein D., Guibas L.J., Henzinger M.R., *Parametric and kinetic minimum spanning trees*, pp. 596–605, Nov 1998.

[2] Basch J., *Kinetic data structures*, 1999.

[3] Basch J., Guibas L.J., Hershberger J., *Data structures for mobile data*, pp. 747–756, 1997.

[4] Basch J., Guibas L.J., Zhang L., *Proximity problems on moving points*, pp. 344–351, 1997.

[5] de Berg M., van Kreveld M., Overmars M., Schwarzkopf O., *Computational Geometry: Algorithms and Applications*, Springer-Verlag, second edition, 2000.

[6] Fu J.-J., Lee R.C.T., *Minimum spanning trees of moving points in the plane*, IEEE Transactions on Computers, **40**(1), pp. 113–118, Jan. 1991.

[7] Guibas L.J., *Kinetic data structures: a state of the art report*, *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics: the algorithmic perspective*, WAFR '98, pp. 191–209, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[8] Guibas L.J., Mitchell J.S.B., Roos T., *Voronoi diagrams of moving points in the plane*, in Gunther Schmidt and Rudolf Berghammer, editors, *Graph-Theoretic Concepts in Computer Science*, volume 570 of *Lecture Notes in Computer Science*, pp. 113–125. Springer Berlin Heidelberg, 1992.

[9] Hjelle Ø., Dæhlen M., *Triangulations and Applications (Mathematics and Visualization)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] Kruskal J.B., *On the shortest spanning subtree of a graph and the traveling salesman problem*, *Proceedings of the AMS*, Vol. **7**, pp. 48–50, 1956.

[11] Prim R.C., *Shortest connection networks and some generalizations*, Bell Systems Technical Journal, pp. 1389–1401, Nov. 1957.

[12] Rahmati Z., Zarei A., *Kinetic euclidean minimum spanning tree in the plane*, *Proceedings of the 22nd international conference on Combinatorial Algorithms*, IWOCA'11, pp. 261–274, Berlin, Heidelberg, 2011, Springer-Verlag.