Energy-Balanced Dispatch of Mobile Sensors in Hybrid Wireless Sensor Network with Obstacles

Shwetha G. K. 2Mrs. Sagarika Behera, 3Dr. Jithendranath Mungara
1(M.Tech, CSE) 2(Assistant Professor, Dept of CSE) 3(Professor Dean, CSE/ISE)
CMRIT, Bangalore

Abstract: We consider a hybrid wireless sensor network with static and mobile nodes. Static sensors monitor the environment and report events occurring in the sensing field. Mobile sensors are then dispatched to visit these event locations to conduct more advanced analysis. The sensing field may contain obstacles of any shape and size. A big challenge is how to dispatch the mobile sensor to the event location without colliding with any obstacles and in a shortest path. Therefore the objective of the paper is to dispatch the mobile sensor to the event location in a hybrid wireless sensor network in the presence of obstacles. Our solution proposes a simple way to dispatch the mobile sensor to the event location in the presence of obstacle. This paper contributes in defining a more general and easiest dispatch solution in the presence of obstacle.

Index terms: Collision-Free Path, Mobile Sensor, Hybrid WSN, Global Positioning System, Dispatch, Static WSN.

I. Introduction

Wireless Sensor Networks (WSNs) are based on physically small-sized sensor nodes exchanging mainly environment-related information with each other [1], [2]. Sensors typically have very limited power, memory and processing resources. Therefore interactions between sensors are limited to short distances and low data-rates. Sensor node energy efficiency and sensor network data-transfer reliability are the primary design parameters.

A WSN is usually deployed with static sensor nodes to perform monitoring missions in the region of interest. However due to the dynamic changes of events, a pure static WSN will face more severe problems [3]. By introducing mobility to some or all nodes i.e., by deploying mobile sensor nodes in a WSN, its capability can be enhanced.

Hybrid Sensor Networks with static and mobile nodes open a new frontier of research in Wireless Sensor Networks (WSNs). Static sensors support environmental sensing and network communication. They serve as the backbone to identify where suspicious events may appear and report such events to mobile sensors. These Mobile sensors are more resource-rich in sensing [5] and computing capabilities and can move to particular locations to conduct more complicated missions such as providing in-depth analysis, repairing the network etc [3]. Once static sensors collect the sensed information about the event, mobile sensors are then dispatched to visit these event locations to conduct more in depth analysis about the events. Applications of wireless sensor networks have been studied in [4], [5].

Mobile sensors have less moving energy and all the event locations should be visited, not even single location should not be left un-visited, because that may cause harmful effect to the sensor network. Therefore balancing energy consumption is very important in case of mobile sensors, so that we can serve all the event locations. Dispatching mobile sensors to the event locations in an energy balanced way is discussed in [6].

The sensing field may contain obstacles but the mobile sensor should be dispatched without colliding with any obstacle, and it should reach event location with minimum distance or shortest path.

In this paper we focus on the problem of dispatching mobile sensor to the event location in shortest collision-free path. A mobile sensor of radius r (non-negative integer) has a collision-free motion among obstacles if its center always keeps at a distance of r and preventing the mobile sensor from moving into these expanded areas. This expanded area is treated as configuration-space (C-space). The shortest path of the center consists of a set of circular arcs of circles called vertex circles (v-circles) of radius r centered at obstacles, and common tangents of the arcs. In this the tangent points and v-circles are determined by the value of the radius r.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 proposes a method to compute shortest collision free path for a mobile sensor. Conclusions and future research topics are drawn in Section 4.

www.iosrjournals.org 47 | Page
II. Related Work

Mobile sensors have been intensively researched to improve a WSN’s topology. In [7], static sensors detecting events will ask mobile sensors to move to their locations to conduct more in-depth analysis. The mobile sensor that has a shorter moving distance and more energy, and whose leaving will generate a smaller uncovered hole, is invited.

The work in [6] addresses how to dispatch mobile sensors to the event locations in an energy balanced way, where dispatch problem is considered for a single round. In that we mainly considered centralized dispatch algorithm, in that they discussed two cases based on the values of number of event locations and the number of mobile sensors. One mobile sensor will be dispatched to one event location (when, mobile sensors >= event locations), one mobile sensor will be dispatched to one cluster of event locations (when, mobile sensors < event locations) But they made an assumption that the sensing field does not contain any obstacles. In our paper, we are trying to relax the assumption that is made in the previous paper. We are treating the event location or cluster of event locations as target.

The studies [8], [9] also address the sensor dispatch problem, but they do not consider energy balancing, dispatching mobile sensor in the presence of obstacles and only optimize energy consumption in one round.

There are several works treating the shortest path of a disc. These approaches use the concept of configuration space (C-space). So that a disc can be processed as a point. Chew [10] extended the idea of the visibility graph to a path graph (called tangent graph in [11], [12]), which registers circular arcs on the boundary of the configuration obstacles (C-obstacle) and common tangents of the circular arcs. Hershberger and Guibas [13] further developed this idea to a nonrotating convex body, and pruned the path graph so that the logarithmic factor is removed. Storer and Reif also showed the usefulness of their algorithm for a disc.

The Voronoi diagram [14] is obviously one answer because a point on the Voronoi graph is collision-free for a disc if its shortest distance to the obstacles is larger than the radius of the disc. However, it does not take care of the issue of shortest paths.

III. Computing The Collision Free Path

Sensors are aware of their own locations, which can be achieved by global positioning system (GPS). In Hybrid Wireless Sensor Network, once static sensor identifies the event, Mobile sensor will be dispatched to the event location.

The work in [6] gives the solution to the dispatch problem. The authors constructed a weighted complete bipartite graph \( G = (SUL, SXL) \). Each mobile sensor and event location is converted into a vertex. Edges only connect vertices between S and L. For each \( s_i \) belong to S and each \( l_j \) belongs to L, its weight is defined as \( w(s_i, l_j) = e_{\text{move}} \cdot d(s_i, l_j) \). Where \( e_{\text{move}} \) is the energy required to move unit distance and \( d(s_i, l_j) \) is the distance between \( s_i \) and \( l_j \).

Algorithm to find M (matching between mobile sensor to event location) is as follows:
1. For each location \( l_j \), associate with it a preference list \( P_j \), which contains all mobile sensors ranked by their weights in correspondence with \( l_j \) in an ascending order. In case of tie, sensors IDs are used to break the tie.
2. Construct a queue \( Q \) containing all locations in L.
3. Create a bound \( B_j \) for each location \( l_j \) belongs to L to restrict the mobile sensors that \( l_j \) can match with. Initially, set \( B_j = w(s_i, l_j) \) such that \( s_i \) is the \( \beta \)-th element in \( l_j \)'s preference list \( P_j \), where \( \beta \) is a system parameter.
4. Dequeue an event location, say, \( l_j \) from Q.

www.iosrjournals.org 48 | Page
5. Select the first candidate mobile sensor, say, \( s_i \) from \( P_j \), and try to match \( s_i \) with \( l_j \). If \( s_i \) is also unmatched, we add the match \((s_i, l_j)\) into \( M \) and remove \( s_i \) from \( P_j \). Otherwise, \( s_i \) must have matched with another location, say, \( l_o \). Then, \( l_i \) and \( l_o \) will compete by their bounds \( B_i \) and \( B_o \). Location \( l_i \) wins the competition if one of these conditions is true:

a. \( B_i > B_o \) Since \( l_i \) has raised to a higher bound, we match \( s_i \) with \( l_i \).

b. \( B_i = B_o \) and \( w(s_i, l_i) < w(s_o, l_o) \) since moving \( s_i \) to \( l_i \) is more energy efficient, match \( s_i \) with \( l_i \).

c. \( B_i = B_o \), \( s_i \) is the only candidate of \( l_i \), and \( l_o \) has more than one candidate: In this case, if \( s_i \) is not matched with \( l_i \), \( l_i \) has to increase its bound \( B_i \). However, \( l_i \) may not increase its bound \( B_i \) if \( s_i \) is not matched with \( l_o \). Thus, we match \( s_i \) with \( l_i \). If \( l_i \) wins the competition, we place the pair \((s_i, l_i)\) in \( M \) by \((s_i, l_i)\), remove \( s_i \) from \( P_j \), enqueue \( l_i \) into \( Q \), and go to step 7. Otherwise, remove \( s_i \) from \( P_j \) (since \( l_i \) will not consider \( s_i \) any more) and go to step 6.

6. If \( l_i \) still has candidate in \( P_j \) (under bound \( B_i \)), go to step 5 directly. Otherwise, increase \( l_i \)’s bound to \( B_i = w(s_i, l_i) \) such that \( s_i \) is the \( \beta^{th} \) element in the current \( P_j \) and then go to step 5. (Note that since \( P_j \) is sorted in an ascending order and the first mobile sensor \( s_i \) is always removed from \( P_j \) after step 5, obtain a new larger bound \( B_i = w(s_i, l_i) > w(s_i, l_i) \))

7. If \( Q \) is empty, the algorithm terminates; otherwise, go to step 4.

In [6], authors have calculated the distance between the mobile sensor and the event location without the presence of obstacles. In this paper we are showing the method to calculate the distance in the presence of obstacles and reaching the target in shortest path. We make use the same (above) algorithm to dispatch the mobile sensor.

### IV. Proposed Schema

Once the static sensor identifies the event location, mobile sensor should be dispatched to the event location. While moving mobile sensor towards the event location, it should not collide with any obstacles in the network. Sensor nodes are battery oriented, energy minimization is very important in WSN and mobile sensors have less moving energy it should reach the event location in a minimum distance. The path that allows mobile sensor to move without colliding with any obstacle is called as collision free path.

Our goal is to find the shortest collision-free path from Mobile Sensor \( s_i \)’s current position to event location’s position \((x_j, y_j)\) which is treated as the destination or target, considering the existence of obstacles. Specifically, the movement of \( s_i \) should not collide with any obstacle. Several studies have addressed this issue [15], [16], [17]. Here, we propose a modified approach of that in [16].

Considering its physical size, \( s_i \) is modeled as a circle with a radius \( r \). Intuitively, \( s_i \) has a collision-free motion if its center always keeps at a distance of \( r \) or larger away from every obstacle. This can be done by expanding the perimeters of all obstacles outwardly by a distance of \( r \). This expanded space is called as Configuration space (C-space) [18], [19]. The configuration obstacles (C-obstacle) are computed by expanding original obstacles by a circle of radius \( r \).

After the expansion, an obstacle vertex becomes a circular arc of a circle called the vertex circle (v-circle) of radius \( r \) centered at the vertex. Fig.1 shows v-circle, C-space and C-obstacles for two obstacles. After finding C-space for all the obstacles we should prevent \( s_i \) from moving into this space. The shortest path consists of a set of circular arcs of the v-circles and common tangents of the circular arcs. The “tangent-arc-tangent” and “arc-arc-tangent” are the basic connection patterns on the shortest path.

Then, the problem can be translated to one of finding the shortest path from \( s_i \) to \((x_j, y_j)\), in a weighted graph \( H = (s_i U (x_j, y_j), U V, E) \), where \( V \) contains all vertices \( v \) of the polygons representing the expanded areas of obstacles such that \( v \) is not inside other expanded areas, and \( E \) contains all edges \((u, v)\) such that \( u, v \in \{s_i U \)
Energy-Balanced Dispatch Of Mobile Sensors In Hybrid Wireless Sensor Network With Obstacles

(x, y) U V} and (u, v) does not pass any expanded area of obstacles. The weight of (u, v) € E is the length of uv (weight of the edge considered as the length). Fig.2 gives an example, where the double circles are vertices of H. Nodes e and f are not vertices because they are inside obstacles 2’s and 3’s expanded areas, respectively. Edges (a,b), (a,g), (h,b) and (h,g) € E, but (h, c) and (h, d) does not belongs to E because they pass the expanded area of obstacle 2. After constructing such graph H, we can use the Dijkstra’s algorithm [20] to calculate the shortest path from s, to (x, y).

Algorithm to find shortest path between source and target is as below:

function shortestpath(points, source, target):
    for each point v in points:
        dist[v] := infinity ;
        previous[v] := undefined ;
    end for ;
    dist[source] := 0 ;
    Q := the set of all point in points ;
    while Q is not empty:
        u := point in Q with smallest distance in dist[] ;
        if dist[u] = infinity:
            break ;
        end if
        remove u from Q ;
        for each neighbor v of u:
            alt := dist[u] + dist_between(u, v) ;
            if alt < dist[v]:
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;
            end if
        end for
    end while
    S := empty sequence
    u := target
    while previous[u] is defined:
        insert u at the beginning of S
        u := previous[u]
    end while
    return S;
end shortestpath

This distance d is a metric on the point set of any connected points G, that is,
(1) d(u, v) ≥ 0 for all points u and v of G;
(2) d(u, v) = 0 if and only if u = v ;
(3) d(u, v) = d(v, u) for all points u and v of G; and
(4) d(u, v) + d(v, w) ≥ d(u, w) for all points, u,v and w of G.

Distance Formula: Given the two points (x1, y1) and (x2, y2), the distance between these points is given by the formula:
\[ d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \]

V. Conclusion And Future Work

In this paper, we presented a method to find shortest and collision-free path for a mobile sensor to reach to the event location that is treated as target. This method is the easiest, simple and efficient way to find the path for a mobile sensor.

In our work we have treated mobile sensors to be in circular shape. But they may also be in different shapes. As a future research, we extend our work to find shortest, collision-free path for a mobile sensor which is in any shape.


