An Incremental Deployment Algorithm for Mobile Sensors

Mohammad Emtiyaz Khan
Department of Computer Science
University of British Columbia

***Abstract*— In this report, we propose an incremental deployment algorithm for mobile sensors. The algorithm gives a deployment scheme maximizing coverage, which is an important issue for a variety of applications. We consider a simple one-dimensional problem and formulate a quadratic program with several constraints. The constraints make sure that the solutions are relevant to coverage maximization, and the algorithm keeps making progress at every iteration. The proposed method is tested with simulations where the algorithm shows good performance. An extension to two dimensional case is also discussed.**

I. Introduction

We propose an incremental deployment algorithm for mobile sensors. There are plenty of applications in which mobile sensors are very useful. For example surveillance, in which the task is to prevent intruders from entering protected sites for instance those containing vital assets from attacks and infiltration. A network of autonomous, mobile sensors with limited individual capabilities provide an economical means to do such surveillance.

In these applications an important problem is to maximize the coverage. Usually it's not possible to position the sensors accurately. Hence the sensors are scattered in the area of interest and they may have overlapping sensing regions. A deployment algorithm enable the sensors to position themselves while minimizing overlap (Fig. 1).

Several deployment algorithms do exist in literature and most of them are based on a path planning algorithm. In this approach, positions which result in minimum overlap, are computed first and then a path planning algorithm is employed to guide the sensors to those position. However in a dynamically changing environment, these algorithms present serious computational and communication issues. For example, a moving obstacle may change the path of the sensors. In such situation the optimization needs to be run again. Also the positions of all sensors needs to be communicated to a central server, which increases the communication overhead. An algorithm based on greedy approach is proposed in [1] and several other approaches are reviewed.

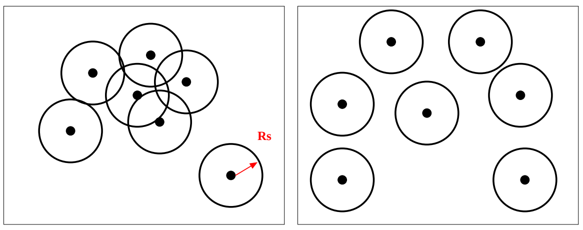

Fig. 1. Deployment algorithm: mobile sensors move to minimize the overlap between the sensors.

We present an incremental algorithm for mobile sensors which is based on similar ideas as [1]. The central idea in this report is to formulate the coverage problem as a continuous optimization problem, and solve it at each time step to get the new positions. Although the idea is similar to [1], we are more concerned about the formulation as a optimization problem and solving it using traditional methods.

II. One Dimensional Deployment Problem

We first solve a simple one dimensional problem where the field of action is a line of length L. Consider N mobile sensors deployed along the line, each with a sensing range of $2R_s$. The sensors are deployed initially (time 0) at arbitrary (known) positions which are denoted by $\{\mathbf{x}_i(0)\}_{i=1}^N$. Each sensor can move along the line and their motion is given by the following uniform speed model:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)T_s \tag{1}$$

Here $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t)$ denote position and speed at time t, and T_s denote the sampling time. We restrict the model by putting bounds on the speed, and hence each $-\bar{\mathbf{v}} \leq \mathbf{v}_i(t) \leq \bar{\mathbf{v}}$.

Our aim is to maximize the area covered by these sensors at every time step. In other words, we want to compute the speed with which sensors should move in order to maximize the coverage (and hence equivalently to minimize the overlap), i.e. to compute $\mathbf{v}_i(t), \forall i, \forall t$

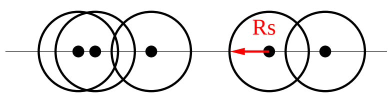

Fig. 2. A one-dimensional deployment problem with multiple sensors. Sensing range is in one dimension (circles are just to show the sensing range clearly).

such that some measure of *overlap* between the individual sensor's sensing range is minimized. We now present the approach.

III. Optimization Problem formulation

Intuitively, minimizing *overlap* is similar to maximizing distance between the sensors, for e.g one possibility is to maximize sum of (a measure of) distance between the sensors. In this report, we used a 2-norm distance function, which gives us the following cost-function:

$$f_2\Big(\mathbf{v}_1(t),\dots,\mathbf{v}_N(t)\Big) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\Big[\mathbf{x}_i(t+1)-\mathbf{x}_j(t+1)\Big]^2 \tag{2}$$

One may also think of using 1-norm distance function (absolute value). However we found that 1-norm doesn't give a tractable cost function. This is shown in Appendix A. We now show that 2-norm doesn't have such problems and it can be formulated as a quadratic program (QP).

For notational convenience, we drop all the time dependence (so that $\mathbf{v}_i(t)$ and $\mathbf{x}_i(t)$ are denoted as $\mathbf{v}_i$ and $\mathbf{v}_i$ respectively). We denote the vector of speed of all the sensors by $\mathbf{v} = [\mathbf{v}_1^T \dots \mathbf{v}_N^T]^T$. We now prove the following proposition:

Proposition 3.1: Maximization of cost-function f_2 is equivalent to the maximization of the quadratic function

$$\bar{f}_2(\mathbf{v}) = \mathbf{v}^T P\mathbf{v} + 2\mathbf{p}^T\mathbf{v} \tag{3}$$

$$\text{where } P = \begin{bmatrix} (N-1) & -1 & \dots & -1 \\ -1 & (N-1) & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & (N-1) \end{bmatrix}$$

and vector $\mathbf{p}$ such that each element is given by $\mathbf{p}_i = N(\mathbf{x}_i - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}} = \sum_{k=1}^{N}\mathbf{x}_k/N$. This proposition is proved in Appendix B.

Although 2-norm gives a nice quadratic cost function, an important thing to note here is that the maxima of the function f_2 be unbounded. For e.g. consider a simple case with only two sensors. The function will be $f_2 = (\mathbf{x}_1 - \mathbf{x}_2)^2$. Fig. 6 shows that the maxima is at the boundary. This will still be true when the number of sensors is increased, as the maximum of

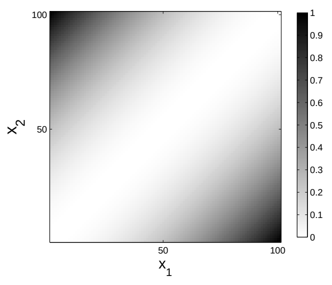

Fig. 3. f_2 when there are only two sensors. Note that the maximum value occur at the boundary.

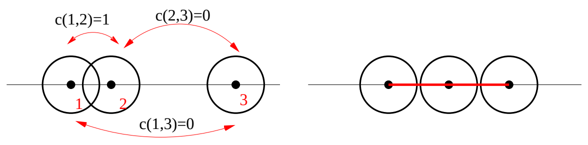

Fig. 4. (Right) Sensor 1 and 2 have distance less than $2R_s$, and hence have overlap. But sensor 3 is far enough from both the sensors . Care function for pair (1,2) is 1, and for pair (1,3) and (2,3), it is 0. (Left) The range for which care function is defined to be 1 for middle sensor.

sum for positive functions will still be the sum of the maximums (although not vice-versa). Hence in general, the maximum will be found at the boundary. We will see that this will always result in the situation wherein sensors are moving either at maximum or at minimum speed. This may not be desirable always. However we still formulate this problem as a continuous optimization and hope that this problem can be fixed later using some regularization.

We now constrain the problem to remove the solutions which are not relevant to overlap minimization.

A. Constraint I : Care function

The function f_2 in Eq. (2) treats all sensor pairs equally important. However this is not we require always. Consider the example shown in Fig. 4 (left), where sensor 1 and 2 are close enough, however sensor 3 is at a distance more than $2R_s$ from both sensor 1 and 2. By maximizing distance of 3 with other sensors we don't gain any decrease in overlap (as it doesn't have any overlap with any sensor). We would like to maximize only distance between 1 and 2 as there is an overlap.

Hence if two sensors are already at a distance more than $2R_s$, then that distance should not be included in the function. For this purpose, we define a care function $c_{i,j}$ as follows:

$$c_{i,j} = \begin{cases} 1 & \text{if } \mathbf{x}_i - 2R_s \le \mathbf{x}_j \le \mathbf{x}_i + 2R_s \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

From this definition the range for which the sensor *cares* about is shown in Fig. 4(right). We can now modify the cost-function as follows:

$$f_2^c(\mathbf{v}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{i,j} \Big[\mathbf{x}_i(t+1) - \mathbf{x}_j(t+1)\Big]^2 \qquad (5)$$

and following the procedure similar to Proposition 3.1 we can obtain a quadratic objective function:

$$\bar{f}_2^c(\mathbf{v}) = \mathbf{v}^T \bar{P} \mathbf{v} + 2\bar{\mathbf{p}}^T \mathbf{v}$$

The matrix $\bar{P}$ and $\bar{\mathbf{p}}$ will have similar form as P and $\mathbf{p}$ except that the contribution from the sensors with $c_{ij} = 0$ will be zero. For e.g. for the situation in Fig. 4 where $c_{1,2} = 1, c_{2,3} = c_{1,3} = 0$,we will have,

$$\bar{P} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ instead of} P = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Now (intuitively) if a sensor is far enough from all the other sensors (and hence doesn't have the overlap with any of them), then the sensor need not move at all. Hence for those sensors, the speed should be set to zero. This gives us an equality constraint.

Equality Constraint: For all Sensor i such that $\bar{P}_{i,k} = 0, \forall k \neq i$, we have,

$$\mathbf{e}_i^T \mathbf{v} = 0 \qquad (6)$$

Stacking these constraints into a matrix, we get equality constraints:

$$A\mathbf{v} = 0 \qquad (7)$$

where rows of A consists of $\mathbf{e}_i$ for all sensors i which satisfy the condition.

B. Constraint II: Incremental improvement

One important requirement is that the algorithm improvement the situation at every (time) iteration, that means that the separations between sensors should not decrease. We capture this in terms of an inequality constraint. It is shown in the Fig. 5 that for all i, j with $c_{i,j} = 1$ if $\mathbf{x}_i(t) > \mathbf{x}_j(t)$, then $\mathbf{v}_i(t) \geq \mathbf{v}_j(t)$ to ensure that $|\mathbf{x}_i(t+1) - \mathbf{x}_j(t+1)| \leq |\mathbf{x}_i(t) - \mathbf{x}_i(t)|$. Hence the inequality constraint:

$$-\text{sign}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{v}_i - \mathbf{v}_j) \leq 0 \quad \forall c_{i,j} = 1 \qquad (8)$$

This can be simplified and stacked to get the following inequality constraints:

$$B\mathbf{v} \leq 0 \qquad (9)$$

where rows of B consists of constrains given by Eq. (8).

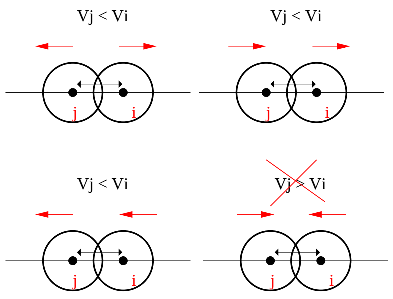

Fig. 5. Proof for the claim for incremental improvement

C. Constraint III: Bound Constraints

Finally we have the bounds on the speed and the boundary condition. Speed for every agent is constrained to a range $[-\bar{\mathbf{v}}, \bar{\mathbf{v}}]$, and every agent is supposed to remain within the field of action $[0, L]$. This is captured in the following constraints:

$$\max(-\mathbf{x}_i, -\bar{\mathbf{v}}) \leq \mathbf{v}_i \leq \min(\bar{\mathbf{v}}, L - \mathbf{x}_i) \quad \forall i \qquad (10)$$

Stacking these in a vector, we get:

$$\mathbf{b}_l \leq \mathbf{v} \leq \mathbf{b}_u \qquad (11)$$

with $\mathbf{b}_l, \mathbf{b}_u$ are vectors of bounds for each sensor.

D. Final Quadratic Program

The resulting quadratic program is given below:

$$\begin{array}{ll} \max\limits_{\mathbf{v}} & \frac{1}{2}\mathbf{v}^T \bar{P} \mathbf{v} + \bar{\mathbf{p}}^T \mathbf{v} \\ \text{sub.to} & A\mathbf{v} = 0 \quad B\mathbf{v} < 0 \\ & -\bar{\mathbf{v}} \leq \mathbf{v} \leq \bar{\mathbf{v}} \end{array} \qquad (12)$$

To solve the above problem we used the package `quadprog` available in MATLAB optimization toolbox. The algorithm used is Active set method (medium-scale optimization). A detailed description can be found in [2]. The algorithm finds a feasible point during an initial phase, and then search for a solution along the edges and faces of the feasible set by solving a sequence of equality-constrained quadratic programming problems. For Equality-constrained QP it uses Null-space/range space methods depending on the matrix P.

The incremental deployment algorithm is given below:

```
INCREMENTAL DEPLOYMENT ALGORITHM
1   $t \leftarrow 0$
2   while $|\max \bar{f}_2^{(t)} - \max \bar{f}_2^{(t-1)}| < \tau$
3       do
4           Find find $c_{i,j}$ for all pairs of sensors
5           Solve QP given by Eq. (12) to get $\mathbf{v}^*$
6           Update positions $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i^* T_s$
7           $t \leftarrow t + 1$
```

We next present the results for the above algorithm.

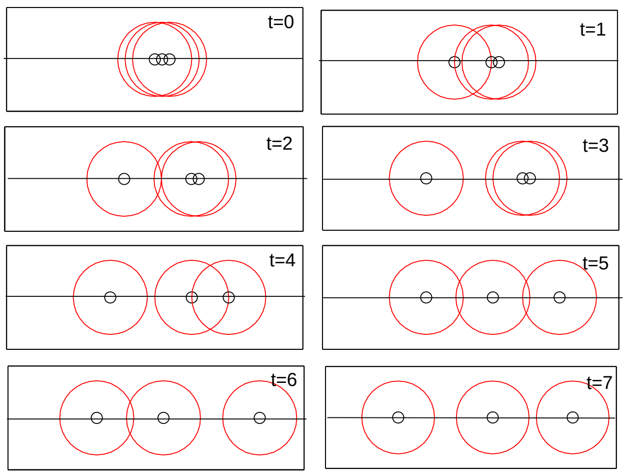

Fig. 6. Results for a simple case: 3 sensors with range of 5m and maximum speed 2m/sec.

IV. Results for one -D case

Results for a simple case with $N = 3$, $R_s = 5$ and $\bar{\mathbf{v}} = 2$ is shown in the Fig. 6. We can see that in seven time step the overlap is zero. At each time step the QP chooses the speed which is maximum, minimum or zero (as explained in Section III, it was expected with 2-norm distance). The algorithm was also tested for several different number of sensors, and in most of the cases the algorithm minimizes the overlap significantly. However one problem occurs when the sensors are too much crowded, the algorithm doesn't settle fast. In such cases setting the threshold properly helps.

V. Conclusions

We formulated one-dimensional coverage problem as a quadratic program with cost function as sum of the distance function. We added few constraints to eliminate the solutions irrelevant to coverage maximization, and to make sure that the algorithm makes improvement at every iteration. We found that the approach shows significant results, and can potentially be extended to 2-D case (Appendix C).

Appendix

A. Problem with 1-norm

We may consider taking a 1-norm distance function and maximize the following function:

$$f(\mathbf{v}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |\Delta\mathbf{x}_{ij} + \mathbf{v}_i - \mathbf{v}_j| \quad (13)$$

$$\leq \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |\Delta\mathbf{x}_{ij}| + |\mathbf{v}_i - \mathbf{v}_j| \quad (14)$$

So one alternative is to maximize the upper bound:

$$\bar{f}(\mathbf{v}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |\mathbf{v}_i - \mathbf{v}_j| \quad (15)$$

$$= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |\mathbf{e}_i - \mathbf{e}_j|^t |\mathbf{v}| \quad (16)$$

$$= (n-1)\sum_{i=1}^{n} |\mathbf{v}_i| \quad (17)$$

as $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |\mathbf{e}_i - \mathbf{e}_j|^t = (n-1)\mathbf{1}$. It's easy to see that this maximize says that all the speed term should be chosen at the boundary (Similar to the 2-norm but it doesn't matter what sign they take, which is intuitively wrong. Consider a case with two sensors, these sensors should not move towards each other as it will decrease the distance and increase the overlap. But the above maximization may consider that solution also (This is also illustrated in Fig. 5).

B. Proof of Proposition 3.1

Using the motion model Eq. (1), we can simplify each term in Eq. (2),

$$\mathbf{x}_i(t+1) - \mathbf{x}_j(t+1) = \Delta\mathbf{x}_{ij}(t) + \left[\mathbf{v}_i(t) - \mathbf{v}_j(t)\right]$$

where $\Delta\mathbf{x}_{ij}(t) \equiv \mathbf{x}_i(t) - \mathbf{x}_j(t)$. We can expand Eq. (2) to get,

$$f_2(\mathbf{v}) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \left[(\mathbf{v}_i - \mathbf{v}_j)^2 + 2\Delta\mathbf{x}_{ij}(\mathbf{v}_i - \mathbf{v}_j) + \Delta\mathbf{x}_{ij}^2\right]$$

Ignoring the lost term and using the fact $\mathbf{v}_i = \mathbf{e}_i\mathbf{v}$ ($\mathbf{e}_i$ is the standard basis vector), we get:

$$\bar{f}_2(\mathbf{v}) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \mathbf{v}^T(\mathbf{e}_i - \mathbf{e}_j)^T(\mathbf{e}_i - \mathbf{e}_j)\mathbf{v} + 2\Delta\mathbf{x}_{ij}^T(\mathbf{e}_i - \mathbf{e}_j)\mathbf{v} \quad (18)$$

$$= \mathbf{v}^T\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} (\mathbf{e}_i - \mathbf{e}_j)^T(\mathbf{e}_i - \mathbf{e}_j)\right]\mathbf{v} + 2\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \Delta\mathbf{x}_{ij}^T(\mathbf{e}_i - \mathbf{e}_j)\right]\mathbf{v}$$

$$= \mathbf{v}^T P\mathbf{v} + 2\mathbf{p}^T\mathbf{v} \quad (19)$$

It can be verified that the P and $\mathbf{p}$ will have the form given in the proposition 3.1

C. Extension to two dimension deployment problem

In two dimensional case, each $\mathbf{x}_i$ and $\mathbf{v}_i$ will be two dimensional vector (elements corresponding to x and y coordinates). Following the approach similar to 1-D case we can maximize the following function.

$$f(\mathbf{v}_1(t), \ldots, \mathbf{v}_N(t)) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} ||\mathbf{x}_i(t+1) - \mathbf{x}_j(t+1)||_2^2 \tag{20}$$

Define P_i such that $P_i\mathbf{v} = \mathbf{v}_i$. P_i will have the following form:

$$P_i = \left[\begin{array}{ccc} \mathbf{0}_{2\times 2} \ldots \mathbf{0}_{2\times 2} & I_{2\times 2} & \mathbf{0}_{2\times 2} \ldots \mathbf{0}_{2\times 2} \end{array} \right] \tag{21}$$

with identity at the column $(2i-1)$. Proceeding similar to the one-D case we can obtain the following quadratic function:

$$\bar{f}(\mathbf{v}) = \mathbf{v}^T P \mathbf{v} + 2\mathbf{p}^T \mathbf{v} \tag{22}$$

where $P = \begin{bmatrix} (N-1)I & -I & \ldots & -I \\ -I & (N-1)I & \ldots & -I \\ \vdots & \vdots & \ddots & \vdots \\ -I & -I & \ldots & (N-1)I \end{bmatrix}$ with each identity matrix of size 2×2. Also the vector $\mathbf{p}$ will be with the elements $2i-1$ and $2i$ will be given as follows: $\mathbf{p}_{2i-1:2i} = N(\mathbf{x}_i - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}} = \sum_{k=1}^{N} \mathbf{x}_k / N$.

A similar QP can be written with the constraints. The care function can be extended directly to two dimensional and so hence also the Equality constraint: $A\mathbf{v} = 0$. The inequality constraint for incremental improvement needs to be explored further as it is not easy to extend using the same approach as one-dimension problem.

References

[1] Andrew Howard, Maja J Mataric and Gaurav S Sukhatme *An Incremental Self-Deployment Algorithm for Mobile Sensor Networks*, Autonomous Robots, Special Issue on Intelligent Embedded Systems, 13(2), Sept 2002, pp. 113–126.

[2] Gill, P. E. and W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, London, UK, 1981.