2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Worst-Case Performance of a Mobile Sensor Network Under Individual Sensor Failure

Hyongju Park and Seth Hutchinson

Abstract— **In this paper, we consider the problem of worst-case performance by a mobile sensor network (MSN) when some of the nodes in the network fail. We formulate the problem as a game in which some subset of the nodes act in an adversarial manner, choosing their motion strategies to maximally degrade overall performance of the network as a whole. We restrict our attention in the present paper to a target detection problem in which the goal is to minimize the probability of missed detection. We use a partitioned cost function that is minimized when each sensor executes a motion strategy given by Lloyd's algorithm (i.e., each agent moves toward the centroid of its Voronoi partition at each time instant), and when the probability of missed detection for each functioning sensor increases with the distance between sensor and target for correctly functioning sensors; adversarial nodes in the network are unable to detect the target, and move to maximally increase the probability of missed detection by the properly functioning sensors. We pose the problem as a multi-stage decision process, and use forward dynamic programming over a finite horizon to numerically compute optimal strategies for the adversaries. We compare the resulting strategies to a greedy algorithm, providing both system trajectories and evolution of the probability of missed detection during execution.**

I. Introduction

Mobile sensor networks (MSNs) have been a popular research area over the past decade [1], [2], [3], [4], [5], [6], [7], [8], [9]. For many applications, MSNs require relatively low power, and have low system and maintenance costs. MSNs have been used, for instance, for ocean sampling, odor source detection and localization, and contamination source detection [2]. However, as with any large, distributed system, there is a high likelihood that one or more nodes in the network will fail at some point in time. Failures include malfunctioning sensors, in which case sensor measurements will be corrupted, and malfunctioning actuators, in which the motion of sensor nodes will be suboptimal.

In this paper, we consider the worst-case performance of an MSN under such failures. Specifically, we consider the case for which a malfunctioning sensor always gives an erroneous value of the quantity to be sensed, and a malfunctioning sensor node moves along the trajectory that maximally degrades overall detection performance.

Our emphasis on worst-case analysis also applies to situations in which certain nodes in the network deliberately act maliciously (e.g., this could occur in military applications if an enemy were to gain control of a subset of the deployed sensors). This leads us to formulate the problem as a game in which a set of adversarial agents act to maximally degrade the performance of the overall network. Throughout the paper, we will therefore consider two sets of agents: a set of *cooperative agents*, who function properly and perform under the assumption that all other nodes in the MSN are functioning properly, and the set of *adversarial agents*. Note that we assume throughout this paper that cooperative agents are not aware of the identities (or existence) of the adversarial agents. In many situations this is a reasonable assumption, but relaxing this assumption is still one of our long-term goals.

To formalize our problem, we consider a specific target detection problem. An MSN comprising N mobile sensor nodes has the task of detecting a target whose location is specified only by a probability distribution ϕ. We adopt the model proposed in [6], [9], and assume that a sensor is able to detect the target only if it is the closest sensor to the target (this is a so-called *partitioned* cost model, in which sensors are able to detect only those targets that lie within their own Voronoi regions [10]). Furthermore, the probability that a functioning sensor will detect a target decreases monotonically with the distance to the target. It has been shown that the optimal deployment of such an MSN (i.e., the configuration of sensors that minimizes the probability of missed detection) is achieved when each sensor behaves according to the Lloyd's algorithm [11], [12], i.e., each sensor moves toward the centroid of its own Voronoi region at each time step. Therefore, here we assume that all cooperative agents faithfully execute Lloyd's algorithm.

For this problem, we can now precisely state the optimality criterion for the adversarial agents. Let p_i denote the configuration of the i^{th} node in the MSN. For a deployment of sensors $\mathcal{P} = \{p_1, \ldots, p_N\}$, let $\mathcal{L}(\mathcal{P})$ denote the probability of missed detection by the network (as will be shown below, $\mathcal{L}(\mathcal{P})$ takes into account the behavior of both cooperative and adversarial agents). If there are m adversarial agents, whose configurations are $p_1, \ldots, p_m$, the optimal configuration of the adversaries satisfies

$$\{p_1^*, \ldots, p_m^*\} = \underset{p_1,\ldots,p_m}{\arg\max}\, \mathcal{L}(\mathcal{P}) \tag{1}$$

under the constraint that p_j is located at the centroid of the Voronoi region for agent j, for $m+1 \leq j \leq N$. In other words, the adversaries attempt to maneuver so

H. Park is in the Department of Mechanical and Science Engineering at the University of Illinois. park334@illinois.edu

S. Hutchinson is a professor of Electrical and Computer Engineering at the University of Illinois. seth@illinois.edu

This material is based in part upon work supported by AFOSR (award AF FA9550-12-1-0193) and the National Science Foundation (award CNS 0931871).

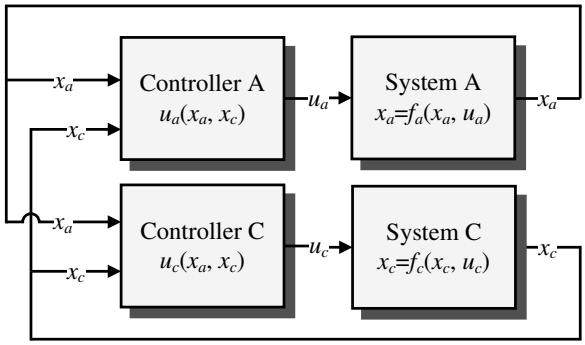


Fig. 1: Block diagram of our control system.

that the cooperative agents converge to the worst possible configurations when they execute Lloyd's algorithm.

The usefulness of the solution to (1) is twofold. First, it provides an answer to the question *How badly can the detection performance of an MSN be degraded by deploying adversarial agents?* This can be interpreted as a measure of robustness of MSN to the adversarial agents. Second, the solution can be used by actual adversaries to maximally disrupt the performance of an MSN.

This paper is organized as follows. Section II formally defines our problem. In section III, we define the control policy for cooperative agents. At this point, we have not yet derived closed form expressions for the optimal adversary strategies. Rather, we present numerical results that have been obtained using finite-horizon dynamic programming. The approach is described in Section IV, and we demonstrate the simulation results in Section V. Finally, Section VI concludes the paper with future works.

II. System Model

A. Our Network Environment

We consider N mobile sensors with position $\mathcal{P} = \{p_1, \ldots, p_N\}$ deployed in a bounded, convex workspace Q in $\mathbb{R}^2$. Suppose that m out of N agents are adversarial agents denoted with $\mathcal{A} = \{p_1, \ldots, p_m\}$, while rest of the agents are cooperative agents denoted with $\mathcal{C} = \{p_{m+1}, \ldots, p_N\}$.

B. Two Subsystems

Our system is composed of two subsystems as shown in Fig. 1. One is *System C* which is a system for set of cooperative agents, and the other is *System A* which is a system for adversarial agent. First, given the previous states from both sub-systems, control inputs are generated from controller A, and C, then applied to System A, and C respectively. As you can see from Fig. 1, sub-system blocks are placed in paralleled which implies that System A, and C evolve in synchronous manner. Control law for system A, and system C must be different because cooperative agents tries to minimize the cost function $\mathcal{L}(\mathcal{P})$ in (1), while adversarial agent try to maximize it.

C. State Space, and State Transition

Our system is a tuple $(X_c, X_a, U_c, U_a, X_{c0}, X_{a0}, f_c, f_a)$, where X_c is the state space for cooperative system, X_a is the state space for adversarial system, U_c, and U_a are input spaces for system C, and A respectively, $X_{c0} \subset X_c$ is the set of initial state space for cooperative system, $X_{a0} \subset X_a$ is the set of initial state space for adversarial system, $f_c : X_c \times U_c \rightarrow X_c$ is the evolution map of System C, and $f_a : X_a \times U_a \rightarrow X_a$ is the evolution map for System A. Let us denote by $x_c \in X_c$, the state of System C $-$ the set of position of cooperative agents, and $x_a \in X_a$, the state of System A $-$ the set of position of adversarial agents. First the state for System A is

$$x_a^k = \{p_1^k, \ldots, p_m^k\} \tag{2}$$

The state transition equation for System A is

$$x_a^{k+1} = f_a(x_a^k, u_a^k) \tag{3}$$

For simplicity, in our study we use the following equation which is the discrete version of integrator dynamics.

$$x_a^{k+1} = x_a^k + u_a^k, \tag{4}$$

In similar manner, the state for System C is given by

$$x_c^k = \{p_{m+1}^k, p_{m+2}^k, \ldots, p_N^k\} \tag{5}$$

The state transition equation for System C is

$$x_c^{k+1} = f_c(x_c^k, u_c^k) \tag{6}$$

where the superscripts denote the time indices. State of System C evolves with Lloyd's algorithm which will be reviewed shortly in the next section. Now using the state transition equation (4) of System A, let us define the *reachable state space* for system A at stage k and denote it as $X_{\text{rss}}^k \subset X_a$ given initial configuration $x_a^0 \in X_a$. It can be obtained in recursive way as follows.

$$\begin{aligned} X_{\text{rss}}^0 &= \{x_a^0\} \\ X_{\text{rss}}^k &= \{x_a^k \in X_a \mid x_a^k = x_a^{k-1} + u_a^{k-1}\} \\ & u_a^{k-1} \in U_a,\ x_a^{k-1} \in X_{\text{rss}}^{k-1},\ k = 1, \ldots, K \end{aligned} \tag{7}$$

The reachable state space was defined to account for the physical constraint of each adversarial agent $-$ it can only move to its neighborhood during one stage. It will also be used to provide condition for the stopping criteria in the later section.

III. Control Policy of Cooperative Agents

A. Voronoi Partitions

Given a bounded, convex workspace Q, and configuration of agents $\mathcal{P}$, the Voronoi partition of i^{th} agent is defined as follows.

$$V_i = \{w \in Q \mid \text{dist}(w, p_i) \leq \text{dist}(w, p_j), \forall i \neq j\}$$

where $i, j \in \{1, \ldots, K\}$, and $\text{dist}(\cdot, \cdot)$ is a Euclidean distance between two points. We denote by C_{V_i} *the Centroid of the Voronoi Partition* V_i given Q, and $\mathcal{P}$.

B. Discrete Lloyd's Algorithm

In this paper, we deal with the special case when at stage k, each cooperative agent moves toward the centroid of its Voronoi partition while adversarial agents follow their own policy. Considering these two heterogeneous sub-systems, a slightly modified version of Lloyd's algorithm from [6], [12] is given below. For simplicity, the superscripts k which denote the number of stage were suppressed at the moment.

0) Initially, for each stage there are N number of agents with position $\{p_1, \ldots, p_N\}$ deployed in our workspace Q. Let us define a new set of variables $\{\hat{p}_1, \ldots, \hat{p}_N\}$ which will be used to store the *set of desired positions for cooperative agents*.

$$\hat{p}_i \leftarrow p_i, \text{ for } i = 1, \ldots, N$$

1) For set of desired positions of agents $\{\hat{p}_1, \ldots, \hat{p}_N\}$, construct a set of Voronoi partition $\{V_1, \ldots, V_N\}$.
2) For set of desired positions of agents $\{\hat{p}_1, \ldots, \hat{p}_N\}$, compute its centroid of its Voronoi partition $\{C_{V_1}, \ldots, C_{V_N}\}$.
3) If the following condition is satisfied, terminate the algorithm.

$$\text{dist}(\hat{p}_i, C_{V_i}) < \epsilon, \quad i \in \{m+1, \ldots, N\}$$

where ϵ is a small positive real number. It is the maximum allowable error between $\hat{p}_i$ and C_{V_i} whose value depends on your platform. Otherwise update the set of desired positions of cooperative agents as follows.

$$\hat{p}_i \leftarrow C_{V_i}, \quad i \in \{m+1, \ldots, N\}$$

After the update, return to 1).

Note that during the execution of the algorithm, the adversarial agents' positions were not altered. This makes sense because adversarial agents do not follow the Lloyd's algorithm.

C. Synchronous Control

First assume that all the agents in system C has a same system clock. In other words, every cooperative agent moves in synchronous manner. Furthermore, every agent in system C has identical hardware specifications such that given the control $u_c = \{v_{m+1}, \ldots, v_N\}$, magnitude of every vector in u_c is bounded by some positive real number v_{max} which denotes the *maximum displacement of a cooperative agent* during a single stage. i.e., $\|v_i\| \leq v_{max}$ for $i \in \{m+1, \ldots, N\}$. The state transition equation in (6) is simplified as

$$x_c^{k+1} = x_c^k + u_c^k, \quad k \in \{0, 1, \ldots, K-1\}$$

where the control u_c is a saturation function.

$$u_c = \begin{cases} \hat{p}_i - p_i & \text{if } \text{dist}(\hat{p}_i, p_i) \leq v_{max} \\ v_{max} \cdot \frac{\hat{p}_i - p_i}{\text{dist}(\hat{p}_i, p_i)} & \text{if } \text{dist}(\hat{p}_i, p_i) \geq v_{max} \end{cases} \tag{8}$$

IV. Problem Formulation

A. Probability of Missed Detection as Reward

Our reward was measured with the following function given the configuration of agents $\mathcal{P}$ in Q, which you can also find in [9].

$$\mathcal{L}(\mathcal{P}) = \sum_{j=1}^{N} \int_{V_j} \left(\prod_{i=1}^{N} h(\|q - p_i\|) \right) \phi(q)\, dq \tag{9}$$

$$h(\|w - p_k\|) = 1, \quad \forall w \in Q,\ k \in \{1, \ldots, m\} \tag{10}$$

where, our reward function (9) is a special case of the cost function define in [9] where the dominance region W_j from [9] is Voronoi partition V_j for each $j = 1, \ldots, N$. The function $h : \mathbb{R}^+ \to \mathbb{R}^+$ is defined by $h(x) = \eta\, x^2$ where η is a positive real value which depends on the diameter of our workspace Q such that, $(\eta\, \text{diam}(Q))^2 \leq 1$. The function value of $h(\|q - p_k\|)$ is the probability of an event that an agent located at p_k fails to detect a target given that the target is located at $q \in Q$ *i.e. probability of missed detection*. The equation (10) implies that the adversarial agent always fails to detect any target in Q. Now let us consider the *partitioned reward* $-$ the reward when each cooperative agent can only detect targets in its Voronoi partition $-$ that is analogous to *partitioned cost* in [6], [9]. To obtain the partitioned reward, the following constraint need to be added in addition to equation (9-10).

$$h(\|w - p_i\|) = \begin{cases} 1 & w \in Q \setminus V_i \\ \eta\, \|w - p_i\|^2 & \text{otherwise} \end{cases}, \quad i \in \{m+1, \ldots, N\} \tag{11}$$

The partitioned reward will be used throughout this paper, because the each cooperative agent moves towards the centroid of its Voronoi partition which is the direction of gradient assent for partitioned reward. In other words, the best reward for the adversarial agent would be to maximize the partitioned reward function which cooperative agents try to minimize.

Now, it is necessary to define the current reward function $g : X_c \times X_a \to \mathbb{R}^+$ for each stage which will be used for dynamic programming on next section. Using equation (9-11), the *current reward incurred at stage* $k \in \{1, \ldots, K-1\}$ is given as follows,

$$g^k(x_c^k, x_a^k) = \mathcal{L}(\mathcal{P}^k)$$

subject to,

$$\begin{aligned} &h(\|w - p_j^k\|) = 1,\ \forall w \in Q,\ j \in \{1, \ldots, m\} \\ &h(\|w - p_i^k\|) = 1,\ \forall w \in Q \setminus V_i^k, \\ &i \in \{m+1, \ldots, N\} \end{aligned} \tag{12}$$

where superscript k for each term implies its value at stage.

B. Policy for Adversarial Agents

Let us define a map $\mu : X_c \times X_a \to U_a$. That is given $x_c^k \in X_c$, $x_a^k \in X_a$, we consider a control law for stage k

which depends on state from both subsystems.

$$u_a^k = \mu^k(x_c^k, x_a^k), \quad k \in \{0, 1, \ldots, K-1\} \tag{13}$$

We define a policy π to be the sequence of the above map.

$$\pi = \{\mu^0, \mu^1, \ldots, \mu^{K-1}\} \tag{14}$$

C. Dynamic Programming

The Lloyd's algorithm is a deterministic algorithm; however state transition derived from the algorithm cannot be represented with an analytic function, nor a differentiable function which make this problem not suited for calculus of variations approach. Rather this problem can be posed as a multi-stage decision process as we did in the previous section, so that it can be solved with dynamic programming (DP). The indexing rule in our DP formula is consistent with that you can find in [13]. Since in our problem the final state is unknown, we cannot apply the backwards dynamic programming algorithm; however we have a choice to use forwards DP instead. Before deriving forwards DP, we first formulate the backwards DP problem.

$$H^K(x_c^K, x_a^K) = 0, \tag{15}$$

and

$$\begin{aligned} H^k(x_c^k, x_a^k) \quad &= \quad \max_{u_a^k \in U_a} \Big\{ g^k(x_c^k, x_a^k) \\ &+ \quad H^{k+1}\left(f_c(x_c^k, u_c^k), f_a(x_a^k, u_a^k)\right) \Big\}, \\ &\quad k = 0, 1, \ldots, K-1, \end{aligned} \tag{16}$$

where equation (15) shows that the terminal reward equals to zero, and equation (16) denotes the *optimal reward-to-go* at stage k. The total reward over K stages is

$$H^0(x_c^0, x_a^0) = \max_{u_a^0, \ldots, u_a^{K-1}} \sum_{i=0}^{K-1} g^k(x_c^k, x_a^k) \tag{17}$$

The optimal policy $\pi_d^* = \{\mu_d^{0*}, \mu_d^{1*}, \ldots, \mu_d^{K-1*}\}$ is the set of control which solves the RHS of equation (17). The subscript 'd' implies that DP algorithm was used to solve the problem. For use in Forward DP, let us first define the backwards state transition equation similar to (3), and (6) respectively.

$$x_a^{k-1} = \tilde{f}_a(x_a^k, u_a^{k-1}), \quad k \in \{1, \ldots, K\} \tag{18}$$

$$x_c^{k-1} = \tilde{f}_c(x_c^k, u_c^{k-1}), \quad k \in \{1, \ldots, K\} \tag{19}$$

Now we state our forward DP problem which is analogous to equation (15-16).

$$\tilde{H}^K(x_c^1, x_a^1) = g^0(x_c^0, x_a^0), \tag{20}$$

and

$$\begin{aligned} &\tilde{H}^k(x_c^{K-k+1}, x_a^{K-k+1}) \\ = \quad &\max_{u_a^{K-k} \in U_a} \Big\{ g^{K-k}(x_c^{K-k}, x_a^{K-k}) \\ + \quad &\tilde{H}^{k+1}\left(\tilde{f}_c(x_c^{K-k+1}, u_c^{K-k}), \tilde{f}_a(x_a^{K-k+1}, u_a^{K-k})\right) \Big\}, \\ &k = 1, \ldots, K-1 \end{aligned} \tag{21}$$

where $\tilde{H}^k(x_c^{K-k+1}, x_a^{K-k+1})$ is the *optimal reward-to-come* to stage $K-k+1$ from initial stage. The total reward over K stages is

$$\begin{aligned} \tilde{H}^1(x_c^K, x_a^K) \quad &= \quad \max_{u_a^0, \ldots, u_a^{K-1}} \sum_{k=0}^{K-1} g^k(x_c^k, x_a^k) \\ &\quad k = 0, 1, \ldots, K-1 \end{aligned} \tag{22}$$

Since RHS of (17), and that of (22) is identical, the optimal policy π_d^* obtained from backwards DP solves equation (22), too.

D. Stopping Criteria for our DP

Recall that the final stage K is an unknown, which makes the forward DP algorithm the only feasible approach. There needs to be an criterion to terminate the DP algorithm to prevent it from running forever which is impractical. Our forward DP algorithm has K number of stages which is obtained from

$$K = \arg\min_k \left\{ k \in \mathbb{N} \mid X_{\text{rss}}^k = Q,\, u_a^{k*} = 0,\, u_c^k = 0 \right\} \tag{23}$$

Using the equation, it is our interest to find the stage with minimum index which satisfies two arguments inside the bracket on the RHS. The intuition for the first argument is to consider the size of the workspace, and to avoid local maximums. In other words, we run the forward DP algorithm until the reachable state space of system A at stage K, X_{rss}^K includes workspace Q. In addition, we rely on heuristics that if $u_a^{K*} = 0$, then $u_a^{i*} = 0$ for all $i > K$. $u_c^k = 0$ means that cooperative agents must be placed in their centroid of their Voronoi partition at stage k.

E. Greedy Method

In greedy method, one is only interested in maximizing the current reward incurred at each stage. For each stage the optimal control is obtained from

$$u_a^{i*} = x_a^{i+1*} - x_a^{i*}, \tag{24}$$

where

$$x_a^{i*} = \arg\max_i \, g^i(x_c^i, x_a^i), \quad i = 0, 1, \ldots, K-1 \tag{25}$$

The set of optimal control constitutes an optimal policy $\pi_g = \{\mu_g^{0*}, \ldots, \mu_g^{K-1*}\}$, where subscript '$g$' implies that greedy method is used to solve the problem. The greedy algorithm terminates at stage K when $u_a^{K*} = 0$, and $u_c^K = 0$ for some $K \in \mathbb{N}$.

V. Simulation and Results

In our simulation, we chose our workspace to be $Q = [0, 1]^2$. As you can see from Fig. 2, we have 4 cooperative agents, and 1 adversarial agent initially deployed at centroids of their Voronoi partitions in the workspace Q such that $p_i = C_{V_i}$ for each $i = 1, \ldots, 4$. In DP algorithm, if the state space gets too large, the calculation becomes implausible. For this reason, we discretized the workspace into $n \times n$ grid, and assumed that the adversarial agent can only move along

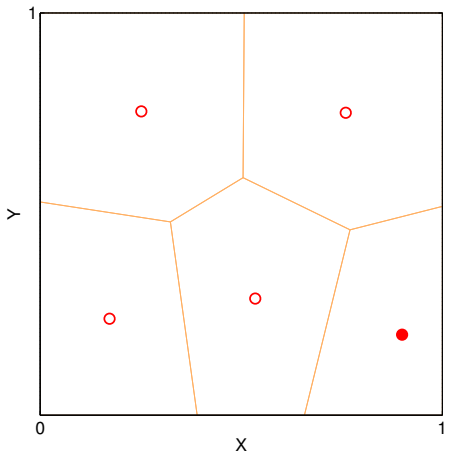


Fig. 2: Initial configuration of agents. (•: adversarial agent, ○: cooperative agents)

the grid. 8-neighbors rule was used, which makes cardinality of input space to be 8. Also, we imposed a constraint $v_{max} = grid\,size \times \sqrt{2}$ − the maximum displacement allowed during one stage − which applies to all agents in MSN. Fig. 3 shows the trace of agents during our multi-stage process under optimal policy obtained from forwards DP algorithm. Fig. 4 shows $g^k(x_c^k, x_a^k)$ − the current reward incurred at each stage k from both forwards DP, and greedy method. Fig. 3, and 4 each contains three sub-figures respectively which are results obtained with different n − the number of grids per column. As one may expect, as n increases, the motion of adversarial agents becomes more realistic. Furthermore If $n \to \infty$, adversarial agents can move over continuous workspace which is the most realistic approach; however at the cost of enormous amount of calculation. In Fig. 3, it could be verified that the case $n = 10$ is a good approximation of the case $n = 30$ in terms of the shape of optimal path for the adversarial agent. In Fig. 4, you can compare the current cost values for each stage. If you take a look at the current cost value for the last stage, given $n = 10,\ 20,\ 30$, the cost value is $0.716,\ 0.717,\ 0.719$ respectively. Also, the case $n = 10$ is a good approximation in terms of cost incurred at the final stage − about 99.58% of that of $n = 30$. As you can see from Fig. 3-(a), and Fig. 4-(a), the results from forwards DP, and greedy method yielded a same solution. This is a special case in which the local optimal policy is the global optimal policy under 4-stage process. In Fig. 4. (a)-(c), final stages indices for $n = 10,\ 20,\ 30$ are $4,\ 9,\ 12$ respectively which does not agree with the final stage index obtained from (23) that is $9,\ 18,\ 27$. This result shows that our heuristic stopping criteria can be inefficient.

VI. Future works

There are many interesting points to be considered in the next stage of our research. First, we plan to increase the number of adversarial agents, or try different adversarial-to-cooperative agents ratio in MSN. Given a fixed total number of agents, increase in the number of adversaries means exponential increase in the size of both state space X_a, and reachable state space X_{rss}. In this case, approximate DP method can be used to find approximate solution with less computation. Ultimately, we would like to derive closed-form formulas for the optimal adversarial agents' strategies. Under such strategies, if the final stage is K, both u_a^{i*}, and u_c^i must remain zero for subsequent stages $i > K$. Furthermore, we plan to consider different control policy for cooperative agents e.g., each agent follows the gradient descent flow minimizing the "total cost" instead of "partitioned cost" defined in [9].

References

[1] A. Howard, M. Mataric, and G. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*. Citeseer, 2002, pp. 299–308.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, 2003.

[4] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM, 2004, pp. 45–57.

[5] P. Ogren, E. Fiorelli, and N. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *Automatic Control, IEEE Transactions on*, vol. 49, no. 8, pp. 1292–1302, 2004.

[6] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 2, p. 243255, 2004.

[7] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, electronically available at http://coordinationbook.info.

[8] M. Schwager, D. Rus, and J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 371–383, 2011.

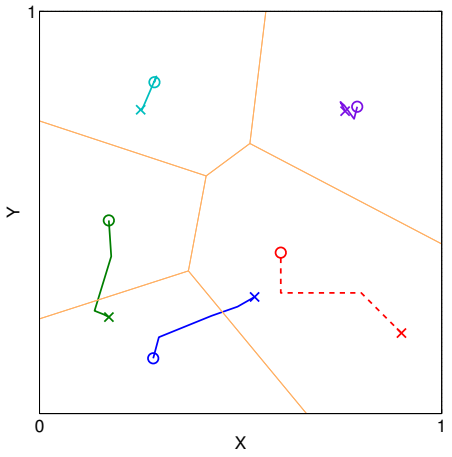
[9] S. Hutchinson and T. Bretl, "Robust optimal deployment of mobile sensor networks," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, p. 671676.

[10] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. Wiley, 2009, vol. 501.

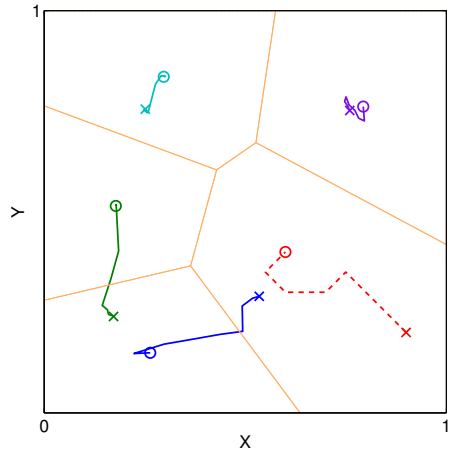
[11] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[12] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

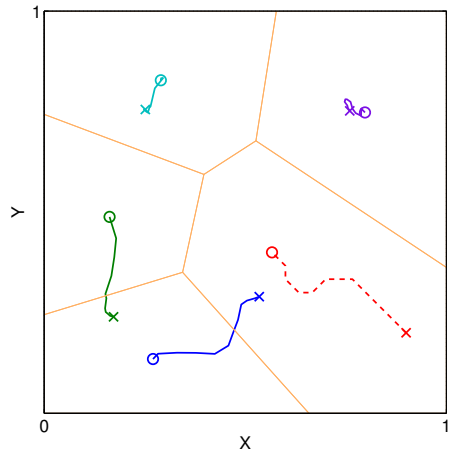
[13] D. Bertsekas, "Dynamic programming and optimal control 3rd edition, volume i," 2011.



(a) $n = 10$

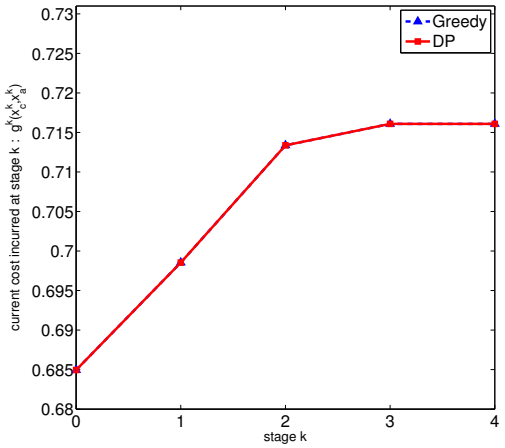


(b) $n = 20$

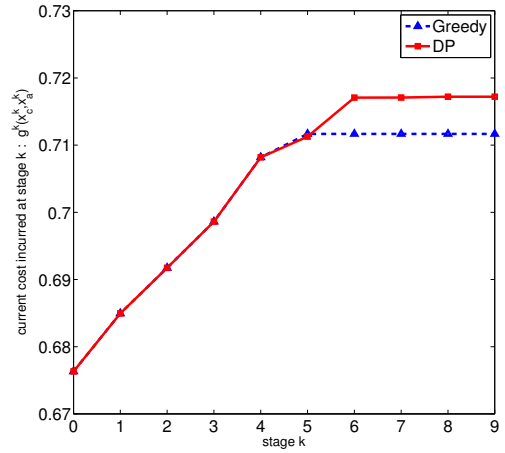


(c) $n = 30$

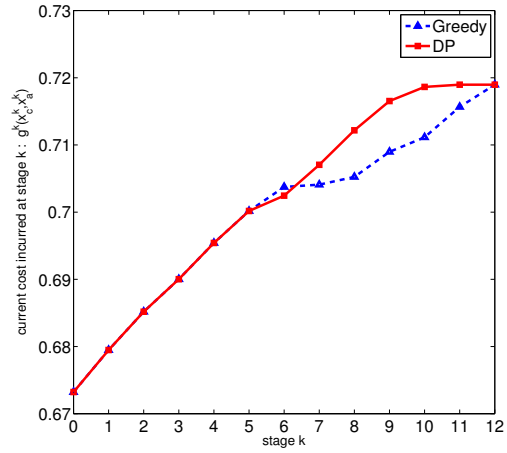
Fig. 3: Trace of agents during the multi-stage process under optimal policy. where X_a was discretized into $n \times n$ grid on Q. ($\times$: initial position of agents, $\circ$: final position of agents, dashed line: trace of adversarial agent, solid line: trace of cooperative agents).



(a) $n = 10$



(b) $n = 20$



(c) $n = 30$

Fig. 4: Current cost incurred at each stage for $n \times n$ grid on Q.