ARS ADVANCED ROBOTIC SYSTEMS INTERNATIONAL

Multi-Robot Searching using Game-Theory Based Approach

Yan Meng
Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA
yan.meng@stevens.edu

Abstract: *This paper proposes a game-theory based approach in a multi–target searching using a multi-robot system in a dynamic environment. It is assumed that a rough priori probability map of the targets' distribution within the environment is given. To consider the interaction between the robots, a dynamic-programming equation is proposed to estimate the utility function for each robot. Based on this utility function, a cooperative nonzero-sum game is generated, where both pure Nash Equilibrium and mixed-strategy Equilibrium solutions are presented to achieve an optimal overall robot behaviors. A special consideration has been taken to improve the real-time performance of the game-theory based approach. Several mechanisms, such as event-driven discretization, one-step dynamic programming, and decision buffer, have been proposed to reduce the computational complexity. The main advantage of the algorithm lies in its real-time capabilities whilst being efficient and robust to dynamic environments.*
Keywords: *multi-robot systems, game-theory, dynamic programming equation, multi-target searching.*

1. Introduction

It is an important and challenging field of research to design and deploy a multi-robot system to perform complex coordinate tasks, such as exploration, search and rescue, and map-building. One of the key issues for a multi-robot system is how to coordinate the behaviors of individual robots so that the overall global performance can be optimized.

To be more efficient for a searching task, it is reasonable to assume that some partial information is available either through distributed sensors installed in the searching area, or based on some heuristics from human beings under the emergency situations. One natural way to capture available information is to represent it as a probability map, which is the likelihood of the target presence in the search space.

Generally, an approach that proved to be more efficient for searching tasks consists of discretizing time and partitioning the continuous space into a finite collection of cells. The search problem is then reduced to deciding which cell to visit at each time interval. In the UAV search scenarios, Bertuccelli and How [Bertuccelli and How, 2005] introduced a statistical framework for an UAV searching for static targets, where the probability associated with each cell was described by a Beta distribution that was updated based on whether a target was detected or not by the sensor. This approach was extended later in [Bertuccelli and How, 2006] for an UAV search with uncertain probability maps to the case of dynamic targets, where the Beta distributions were updated using a recursive method that includes both propagation and measurement update steps.

Eagle [Eagle, 1984] noted that a discrete search can be formulated as an optimization on a partially observable Markov decision process (POMDP) and proposed a dynamic programming solution to it. However, since the optimization of POMDPs is extremely computationally extensive, this approach is often not practical. Instead, Eagle and Yee [Eagle and Yee, 1990], and Stewart [Stewart, 1979] formulated the discrete search as nonlinear integer programming problem and proposed branch-and-bound procedures to solve it, which in the case of [Eagle and Yee, 1990] are optimal.

However, updating the state of each cell at each time interval would be computational/memory expensive and impractical in real-world scenarios. Therefore, other discretization methods have been proposed with probability maps. In Urban Search and Rescue (USAR) scenarios, Murphy [Murphy, 2000] proposed a biomimetic search strategy where a robot partitions the search space based on the semantic understanding of the expected distribution of survivors, then systematically searches each of the volumes in ranked order. In the case of finding targets in free space, such as boats lost at sea, Bourgault et al. [Bourgault et al., 2003] employed a Bayesian approach where the target probability density function (PDF) is used as a priori information. As rescue air vehicles cover the ocean, the target PDF is updated using the model of the sensor and expected target motion. Optimal trajectories for the search are those that maximize the cumulative probability of detection over a limited time horizon. Lau et al. [Lau et al., 2005] divided a searching space into different regions where available information about the targets was expressed by the expected proportion of targets present in each region. A search strategy that minimizes the expected average time for detection of each target was proposed, which deals with an environment where multiple targets may be

present and allows travel and search costs of individual regions to be arbitrarily specified.

However, all of these searching strategies only deal with a single robot. As we know, a multi-robot system is more desirable in some scenarios, such as exploration, search and rescue, due to the robustness, stability, adaptability, and scalability. Meanwhile, the mutual interactions between individual robots sharing a common workspace could be much more complex in general cases.

Some research has been conducted on the multi-robot searching. Kok et. al. [Kok et al., 2003] described a framework to coordinate multiple robots using coordinate graphs (CG). First a discretization of the state by assigning roles to the agents was conducted. Then a CG-based method was applied to the derived set of roles. The capability of dynamic update of the topology of the CG and the role-based coordination rules made this approach to be practical for large scale teams. In [L'opez-Ortizl & Schuierer, 2002], an optimal searching strategy for a target on multiple concurrent rays in parallel using p robots was presented based on a solution of *cow path problem* that can be treated as a special condition with p = 1 and has been solved in [Kao et al., 1993][Baeza-Yates et al., 1993]. The interaction between the robots is relatively simple in this case due to the special configurations. In [Jin et al., 2003], several different targets were considered in a heterogeneous team of UAVs, where some target locations are suspected a priori with a certain probability, while the others are initially unknown.

For those more complex dynamic environments, another methodology of solving coordination problem utilizes a game theory, which is a convenient tool for modeling and solving multi-robot interaction problems. Skrzypczyk [Skrzypczyk, 2004] proposed an architecture of a control system for a real time collision-free movement coordination in a multi-robot environment, performing their navigational tasks by using the normal form games. In [LaValle & Hutchinson, 1994], an approach for analyzing and selecting time-optimal coordination strategies for multiple robots whose configurations were constrained to lie on C-space road map was proposed. The maximal Nash Equilibrium concept was used to find the optimal strategies for each robot. A pursuit-evasion problem as a Markov game was described in [Hespanha et al., 2000], which was the generalization of a Markov decision process to the case when the system evolution is governed by a transition probability function depending on two or more players' actions. This probabilistic setting made it possible to model the uncertainty affecting the player's motion. Combining exploration and pursuit in a single problem was then translated into learning the transition probability function while playing the game. A dynamic programming solution to a Stackelberg equilibrium of a partial-informaiton Markov process was proposed. However, this approach required that the pursuit-evasion policies be learned for each new obstacle configuration. To reduce the complexity of a large-scale team, the architecture model needs to be dynamically simplified. Vidal et al. [Vidal et al., 2002] extended and improved this approach to a probabilistic game theoretical framework to control a team of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) to pursue a second team of evaders while concurrently building a map in an unknown environment in. To reduce the computational complexity, two computationally feasible greedy pursuit policies: local-max and global-max, were proposed.

The main limitation of game-theory based approaches is that the methods are extremely computational expensive, which significantly hamper the practical utilization in real-world scenarios.

In this paper, we aim at developing a robust, real-time game-theory based approach for multi-target searching in a dynamic environment using a multi-robot system. It is assumed that a rough priori probability map of the targets in the environment is given. To cut down the request of system computational time and memory, the environment is partitioned into regions instead of cells. Although a priori probability of targets (i.e., probability map) in each region is assumed to be provided, however, these probabilities may be changed under dynamic environment. Therefore, dynamic updating of the probability map is necessary based on the current sensor information. A dynamic programming is applied to formalize the utility function for each robot, where not only the probability map and travel costs are considered, but also the interaction between the robots. Based on this utility function, a cooperative nonzero-sum game is generated, where both pure Nash Equilibrium and mixed-strategy Equilibrium solutions are provided to guide the robot behaviors to achieve an optimal overall performance.

To reduce the computational complexity of the proposed game-theory based approach, the following mechanisms have been proposed: (1) event-driven discretization, where probability map is only updated when new event occurs instead of at fixed time intervals; (2) one-step dynamic programming, where only current states will contribute to next time step prediction; and (3) a decision buffer method, where future game decisions can be stored in a buffer so that decision making and searching can be conducted in parallel.

The searching time of the proposed game-theory based algorithm can be reduced significantly compared to other heuristic algorithms. Furthermore, due to the mutual understanding of robots, the game theory based algorithm can achieve higher robustness and fault-tolerance under dynamic environment.

The paper is organized as followings. Section 2 describes the problem statement. Section 3 explains how the probability map is updated, and an event-triggered system discretization is proposed to reduce the computation complexity. Dynamic programming based utility function is described in Section 4. Game-theory based approach is proposed in Section 5. Simulation and experimental results are given in Section 6. Section 7 concludes the paper and discusses the future work.

2. Problem statement

The searching environment is discretized into *J* regions. There are *N* homogeneous mobile robots and *G* stationary targets, which may be distributed within one region or at different regions. A rough priori *probability map*, which contains the priori probabilities of target existence in each region, is assumed to be provided. When a searching task starts, this probability map will be updated dynamically based on the new perception information of the robots. In terms of capability of each physical robot, the following assumptions are made:

- Each robot can localize itself within the map using an on-board laser range finder and odometry measurements.
- Each robot can detect the targets using an on-board camera system.
- The robots can communicate with each other through wireless communication within the map.
- Each robot can avoid obstacles and plan an optimal global path to the destination.
- The searching task stops when all targets are found.

The searching problem can be stated as follows: given a rough priori probability map, develop an efficient and robust strategy for multiple robots searching for multiple targets in a dynamic environment so that the expected average searching time can be minimized or near-minimized.

3. Probability Map Update and Event-Driven System Descritization

A rough priori probability map (i.e. likelihood distribution) is installed in the robots. When the searching task starts, this probability map is updated dynamically based on the current searching results.

The vector of priori probability of targets within each region can be represented as:

$$\widetilde{\mathbf{P}}(k \mid k) = [\widetilde{P}_1(k \mid k), \widetilde{P}_2(k \mid k), ..., \widetilde{P}_J(k \mid k)]^T ,$$

where , $\widetilde{P}(k \mid k) \in [0,1]$

where *M* is the maximum region number. The prediction of the probability can be estimated based on its previous probability (at time step *k*) and current measurement observed by robot at time step *k+1*, which is represented as follows:

$$\hat{\mathbf{P}}(k+1 \mid k) = f(\hat{\mathbf{P}}(k \mid k), \mathbf{z}(k+1))$$

$$\hat{\mathbf{P}}(k \mid k) = \widetilde{\mathbf{P}}(k \mid k) \tag{1}$$

where $\hat{\mathbf{P}}_i(k+1 \mid k)$ represents the vector of probabilities $[\hat{P}_1(k+1 \mid k), \hat{P}_2(k+1 \mid k), \ldots, \hat{P}_J(k+1 \mid k)]^T$ predicted at time step *k*+1, *f* represents the update function, $\hat{\mathbf{P}}(k \mid k)$ denotes the vector of probabilities at time step *k* which are estimated based on previous probabilities. $\mathbf{z}(k)$ represents the vector of robot observations $[z_1(k), z_2(k), ..., z_N(k)]^T$ that register target detection or target absence at time step *k*, where $z_i(k)$ denotes the observation by robot *i* at time step *k*, and *N* is the number of robots.

Let $P_j^n(k \mid k)$ denotes the probability of robot *n* choosing region *j* at time step *k*, we have

$$\sum_{j=1}^{J} P_j^i(k \mid k) = 1, \quad i = 1,2,...,N. \tag{2}$$

To update the probability map, Bayesian measurement update is one straight-forward approach, which computes the new probability belief state as a function of robot observation inputs and its former probability belief state. However, this procedure can only be applied to each individual region with new observations from the robot that is searching in that region. If we want to propagate this update to other regions, it would be computational intensive. Therefore, a simple update approach is applied here to meet the real-time performance.

Whenever a robot finishes searching region *i* at time step *k*, where a target may be detected or not, the current probability in this region will be evenly distributed in unsearched regions at next time step *k*+1, and the probability of the searched region at next time step *k*+1 will be set to zero. This procedure would keep the sum of probabilities of unsearched regions always equals to 1. This procedure can be represented as followings:

$$\hat{P}_i(k+1 \mid k) = 0;$$

$$\hat{P}_j(k+1 \mid k) = \frac{\hat{P}_j(k \mid k)}{1 - \hat{P}_i(k \mid k)}, \; j \neq i, \; j=1,2,\ldots,J. \tag{3}$$

where $\hat{P}_i(k \mid k)$ represents the estimated probability of a target in region *i* at time step *k*, $\hat{P}_i(k+1 \mid k)$ denotes the predicted probability of a target in region *i* at time step *k*+1, and *M* is the maximum region number within the map.

If the system is descretized at a predefined time interval, the probability map needs to be updated at each time step. However, in a large-scale searching area with low density of targets, the region-based probability map may stay the same most of the time. To reduce the computational complexity, an event-triggered system descritization is applied in this paper, where the time step and probability map are only updated as necessary instead of at every predefined time step.

Once the searching task starts, the finite state machine of a robot consists of two states: busy and free. The robot state is defined as busy when it is searching inside a region. Otherwise, its state is defined as free. Initially all robots are set as free, and the robots' states are updated dynamically during searching. A new event happens when a robot enters a region or finishes searching its current region, which would trigger the update of the robot states.

The event-trigger is defined as a series of discrete time when each robot changes its state, which can be represented as follows:

$$evt\_trigger = [trig_1, trig_2, ..., trig_N], \quad (4)$$

where N is the number of robots. Next event will be triggered at $\min(trig_1, trig_2, ..., trig_N)$, which will be accumulated with the total searching time and be subtracted from $\max(trig_1, trig_2, ..., trig_N)$. With this event-triggered discretization, the probability map is only updated at each event. Since the robots only communicate with each other upon new event, the communication overhead can be obviated significantly compared with the fixed-time-interval discretization method.

4. Dynamic Programming Equation based Utility Function

Generally, a utility is defined as a payoff value of a robot selecting a region to search at next discrete time. For a multi-robot system, to improve the collective searching efficiency, the utility of each robot does not only depend on its own payoff value, but also on other robots' decisions.

Obviously, the utility associated with each robot depends on the probability of the target at each region. The higher the probability, the higher the utility value should be. However, this probability-only based approach may try to achieve the most valuable goal (i.e. highest probability of target detection) irrespective of the difficulty of that goal. For example, the agent may skip the nearest region which has lower probability, and go for a very far region which has higher probability but may take much longer time to reach. To build more sensible agents, in this paper, we combine travel cost and utility calculations for each action and then calculate the expected utility.

The set of decisions made by robots from 1 to N at time step k is denoted by $\mathbf{D}(k) = [d_1(k), d_2(k), ..., d_N(k)]^T$. The set of probabilities of the target from region 1 to J at time step k is denoted by $\hat{\mathbf{P}}(k \mid k) = \left[\hat{P}_1(k \mid k), \hat{P}_2(k \mid k), ..., \hat{P}_J(k \mid k)\right]^T$. The travel cost can be divided into two time-based vectors, $\mathbf{T}^i(k)$ and $\mathbf{T}_c$, where $\mathbf{T}^i(k) = \left[t_1^i(k), t_2^i(k) ... t_J^i(k)\right]^T$ represents the time vector that robot i takes to navigate from its current position at time step k to different regions from 1 to M, and $\mathbf{T}_c = [t_1, t_2 ..., t_J]$ denotes the set of time required for a robot to cover each region, where t_i denotes the time required to cover the region i for each robot. The utility function of robot i can be defined as a function of $(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i(k), \mathbf{T}_c)$.

First, let us define the payoff value of robot i searching region n as follows:

$$g(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i, \mathbf{T}_c) = \frac{\hat{P}_n(k \mid k)}{k_1 t_n^i(k) + k_2 t_{cn}}, \quad (5)$$

where $t_n^i(k)$ and t_{cn} represent the time required for robot i to navigate from its current position at time step k to region n, and the time required for robot i to cover region n, respectively. k_1 and k_2 are scale factors which can be adjusted based on different environmental structures. For example, robots may move faster in empty corridors and move slower in the rooms with higher density of obstacles. In this case, k_1 is set greater than k_2.

To achieve an optimal overall performance, each robot has to take the current decisions of other team members into consideration when it makes its own decision. To resolve this interaction issue, a Dynamic Programming Equation (DPE) is applied to define the utility function for robot i as follows.

$$U_i(d_1(k), d_2(k), ..., d_N(k)) = f_i(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i(k), \mathbf{T}_c)$$

$$= \begin{cases} 0, & \text{if } P_n(k \mid k) = 0 \\ g(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i(k), \mathbf{T}_c) & \text{if } P_n(k \mid k) = 1 \\ h(\mathbf{D})[g(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i(k), \mathbf{T}_c) + & \\ (1 - P_n(k \mid k)) \max_n \left\{ f(\hat{\mathbf{D}}(k), \hat{\mathbf{P}}(k \mid k), \hat{\mathbf{T}}^i(k), \mathbf{T}_c) \right\}] & otherwise \end{cases} \quad (6)$$

where U_i represents the utility function of robot i, $\hat{P}_n(k \mid k)$ represents the estimated probability of target detection in region n at time step k. $g(\mathbf{D}(k), \hat{\mathbf{P}}(k \mid k), \mathbf{T}^i, \mathbf{T}_c)$ is defined in (5).

$$(1 - p_n(k \mid k)) \max_{\hat{n}} \left\{ f(\hat{\mathbf{D}}(k), \hat{\mathbf{P}}(k \mid k), \hat{\mathbf{T}}^i(k), \mathbf{T}_c) \right\}$$

represents the maximum expected utility of robot i by selecting different $\hat{n}$ for the rest of the unsearched regions after finishing the region n. $\hat{\mathbf{D}}(k), \hat{\mathbf{P}}(k \mid k)$, and $\hat{\mathbf{T}}^i(k)$ represent the estimated values after robot i finishes its current searching at region n at time step k, and are defined as follows respectively:

$$\hat{\mathbf{D}}(k) = \left[d_1(k), ..., \hat{d}_i(k), ..., d_N(k)\right]^T$$

$$\hat{\mathbf{P}}(k \mid k) = \left[\frac{\hat{P}_1(k-1 \mid k)}{1 - \hat{P}_n(k-1 \mid k)}, ..., \frac{\hat{P}_{n-1}(k-1 \mid k)}{1 - \hat{P}_n(k-1 \mid k)}, 0, \frac{\hat{P}_{n+1}(k-1 \mid k)}{1 - \hat{P}_n(k-1 \mid k)} ... \frac{\hat{P}_J(k-1 \mid k)}{1 - \hat{P}_n(k-1 \mid k)}\right]^T,$$

and

$$\hat{\mathbf{T}}^i(k) = \left[\hat{T}_1^i, \hat{T}_2^i, .., T_n^i, ., \hat{T}_J^i\right]^T, \quad (7)$$

The definition of $\hat{\mathbf{D}}(k)$ assumes that when robot i is making decision through this recursive procedure, other robots keep their current decision.

Considering the situation where several robots may choose same region simultaneously based on their own utility functions, which will definitely decrease the overall searching performance. We define a factor $h(\mathbf{D})$ as follows:

$$h(\mathbf{D}) = \begin{cases} k_3 * t_n^i(k) / \sum_{j \neq i, j=1}^{N} t_n^j(k), j \neq i \\ 1 \qquad otherwise \end{cases} \quad (8)$$

where $t_n^i(k)$ is the travel time that robot *i* takes from its current position at time step *k* to region *n*, $\Sigma t_n^j(k)$ is the total travel time that other robots (except *i*) take from their current positions to region *n*, and k_3 is a scale factor to be adjusted based on different environmental structures.

The definition of $h(\mathbf{D})$ actually embeds the coordination between the multiple robots. In other words, cutting down the utility value helps to prevent multiple robots picking up the same region simultaneously, which will eventually improve the overall searching efficiency.

The coefficients $k_1, k_2,$ and k_3 are gains to adjust the influence of individual factors of the utility function. Various searching behaviors can be achieved by tuning these parameters, which will be discussed in simulation section.

The dynamic programming, however, is somewhat intractable for large-scale region numbers. It is possible to approximate the dynamic programming without iterating all possible states. To reduce the computational complexity, we propose one-step dynamic programming solution for Equation (6). Basically, the average expected contribution from team members is calculated as the contribution that the team members would make from their current positions. This approximation is reasonable when each step is relatively small.

In general, the utility is zero if the probability of target detection in region *n* is zero. If the probability of the target detection in region *n* is 1, it means that there is at least one target left and this is the last region need to be searched. In this case, the utility function is only related to the payoff value by searching region *n*. Otherwise, the utility is a recursive function defined in (6).

5. Game-Theory based Searching Strategy

Game theory is the mathematical study of strategies of interaction between different parties. It was emerged during the beginning of the 20th century. It originally mostly concerned with economics, but later applied to politics, evolution, biology etc. Basically, it is assumed that players are rational, in other words, given the information they have, they choose the strategy with the highest expected payoff.

A multi-robot searching task can be modeled as a multi-player cooperative nonzero-sum game. The players choose their strategies simultaneously at the beginning of the game. Although the overall process of the searching is dynamic, we can treat it as a sequence of static games at each discrete time. At each discrete time, the players must solve a static game that is nonzero-sum because the probability in question is conditioned to the distinct observations that the corresponding team is available at that time.

The probabilities of all the regions are updated whenever a region has been searched. With the updated probability map, all free robots recalculate their utilities. The game stops when all targets are found.

According to the current positions of robots and current probability map, the utility matrix $\mathbf{U}_n$ can be calculated. Utility matrix $\mathbf{U}_n$ is a *N*-dimensional matrix for *N* robots, where there are *J* (region number) units at each dimension, and each cell of the matrix consists of *N* utility values for each robot at the corresponding position.

Based on the calculated utility matrix, the Nash equilibrium solution is adopted for this nonzero-sum game. On the one hand, playing at a Nash equilibrium ensures a minimum performance level for each team. On the other hand, no player can gain from a unilateral deviation with respect to the Nash equilibrium policy. According to Nash equilibrium theory, some pure strategy equilibriums of this multi-player game exist if the following inequalities are satisfied:

$$\begin{aligned} &U_1(d_1(k), d_2(k), ..., d_N(k)) \le U_1(d_1^*(k), d_2(k), ..., d_N(k)) \\ &U_2(d_1(k), d_2(k), ..., d_N(k)) \le U_2(d_1(k), d_2^*(k), ..., d_N(k)) \\ &\vdots \\ &U_N(d_1(k), d_2(k), ..., d_N(k)) \le U_N(d_1(k), d_2(k), ..., d_N^*(k)) \end{aligned} \quad (9)$$

In the cases where there is no pure strategic Nash equilibrium, a max-min method is applied to calculate the mixed-strategy equilibrium. Let $p_n(j)$ denotes the probability of robot *n* choosing region *j*, and $\mathbf{P}_n = [p_n(1) \quad p_n(2) \quad \dots \quad p_n(J)]^T$, we have

$$\sum_{j=1}^{J} p_n(j) = 1, \qquad n = 1,2,...,N$$

The max-min method can be defined as follows:

$$\begin{aligned} &d_1^*(k) = \max_{d_1} \min_{d_2,...,d_N} \left\{ \sum_{j=1}^{J} U_1(1,j) p_1(j), ..., \sum_{j=1}^{J} U_1(J,j) p_1(j) \right\} \\ &\vdots \\ &d_N^*(k) = \max_{d_N} \min_{d_1,...,d_{N-1}} \left\{ \sum_{j=1}^{J} U_N(1,j) p_N(j), ..., \sum_{j=1}^{J} U_N(J,j) p_N(j) \right\} \end{aligned} \quad (10)$$

$\mathbf{P}_n$ satisfies the following linear equation $\mathbf{U}_n^m \times \mathbf{P}_n = [0 \quad 0 \quad 0 \quad 1]^T$. Since the game is a finite strategic-form game, the existence of the mixed-strategy equilibrium can be guaranteed. Multiple equilibria may be obtained by using (9) and (10). The agents select the region with the highest probability in $\mathbf{P}_n$ as their next destination. Generally speaking, how to choose the proper solution from the multiple equilibria is a complex problem [Harsanyi and Selten, 1998]. The criteria are that the solution that provides the maximum payoffs for all the players and "fair" distribution of cost among the players should be chosen.

Based on the calculated utility matrix, the Nash Equilibrium (NE) is applied for this nonzero-sum game. Playing at a NE ensures a minimum performance level for the team. On the other hand, no player can gain from a unilateral deviation with respect to the NE policy.

It is easy for robots to make decision if only one NE point is available. However, when the region number increases, there may exist more than one NE point. In this case, a

max-min method is applied to calculate the mixed-strategy equilibrium.

The pseudo code for game-theory based strategic algorithm is described in Fig. 1. Since the searching procedure can be divided into a sequence of static games, the question is when should the robots start a new static game during searching? To answer this question, let's start with a simple case: two robots. There are three scenarios of robot states at any event-driven time step: (1) both are free; (2) one is free and one is busy; and (3) both are busy. Among these three cases, the robots only need to make decisions on the first two cases.

When both robots are free, the utility matrix is calculated and their searching decisions are computed based on the game strategy. If one robot is busy and the other one is free, should we restart a new game whenever one robot is free? Since the other robot is busy in searching a region, it would make more sense to let it finish its current searching instead of reselecting the searching region due to the new status.

```
WHILE (not all of targets are found)
  IF (all robots are free or at least one robot is free)
    Calculate the utility values of each robot using
    equation (6);
      Build the utility matrix U_n and find the
      maximum point of utilities in the matrix for each
      robot;
      IF (the max point for all the robots is located at
      the same point)
        Pure NE point. Take this NE point as the next
        decision;
      ELSE
        Find the mixed strategy using the max-min
        method to estimate probability P_n by solving
        the equation U_n^m × P_n = [0 0 0 1]^T with
        linear programming algorithm;
        Select the region with the highest probability
        in P_n as next decision;
      END;
    END;
    Estimate the time when the next trigger event will
    happen based on the time traveled to new destination
    and the time searching the current region;
    Move the robots to the destination region;
    Update the searching status;
END;
```

Fig.1. Game-theory based strategic algorithm

There are two options to resolve this problem. First one is we don't start a new game, and the free robot makes its decision only based on its utility value at current time step. The motivation for this simplified procedure is to reduce the computational time. If a new game is restarted whenever one robot is free, which may happen very frequently, especially in a large-scale multi-robot system, the computational time would be increased significantly since the game strategy needs longer time to make decision. However, this approach tends to turn game theory approach into a utility-based approach in a large-scale robot team, where the chance that all of the robots are free at the same time is seldom.

To make the proposed game theory approach to be scalable, a decision-buffer based approach is proposed, where its flowchart is shown in Fig. 2. Basically, each robot creates a decision buffer to store the new decisions whenever a new game starts. The new decisions are stored in the format of region numbers in the order of searching sequence. Once a robot finishes its current region, the searched region number will be erased from the buffer. It will check if there is any new decision available. If yes, it will move to the assigned region of new decision. If not, it will start a new game with the assumption that all robots have finished their current searching and probability map has been updated although some team members may be still busy in reality. Once a new decision is made, all robots will update their decision buffer. For those busy robots, they will continue their current searching until they become free, then they would move to the assigned regions with new game decision without starting a new game. Since this approach avoids lots of redundancies in decision making and parallelizes the searching and decision making procedures, it reduces the computational complexity and is very efficient in terms of decision making. Both approaches have been implemented in this paper. To distinguish them, the first one is called *game theory (GT)*, and second one is called *game theory with buffer (GT_buf)*.

In addition, buildings may collapse, facilities may scatter, and some of the obstacles may block the ways to enter regions under real-world USAR scenarios. Therefore, the real world map may be different from the given map. It is reasonable to expect that some unexpected obstacles are distributed randomly in the environment. If this is the case, a new game has to be restarted if utility value of any robot has been changed to adapt to a new path.

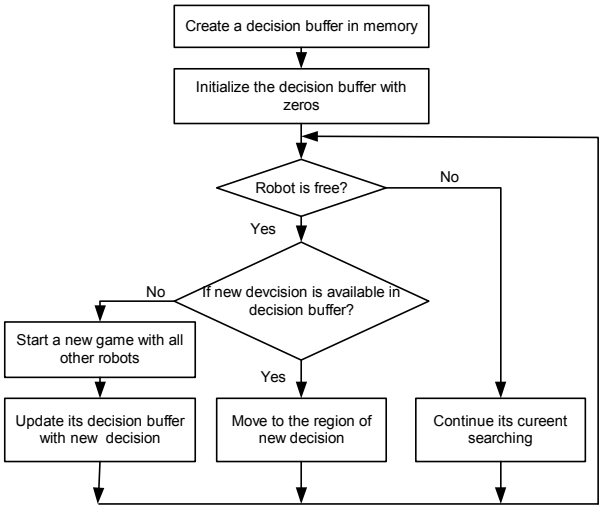


Fig. 2. Flowchart of decision-buffer based approach

For a multi-robot system, the overall outcome depends critically on the choices made by each self-interested robot. Each robot simply computes the best utility outcome and knows that the other robots they are working with will do the same. Since robots can use game theory to predict what others will do, which obviates the need for explicit communication – coordination arises because of the assumption of mutual rationality.

6. Simulation Results and Discussion

6.1. Other Heuristic Searching Strategies

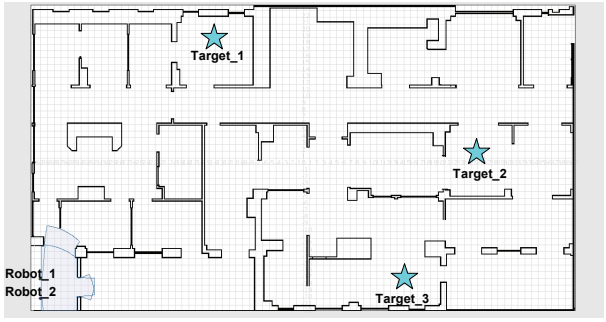
In order to evaluate the searching performance of the proposed game-theory based approaches (i.e., *game theory without buffer (GT)* and *game theory with buffer* (*GT_buf*)), comparison with other heuristic searching strategies is necessary.

Several searching strategies are proposed here. First one is called *randomly selection (RS)* approach, where each robot randomly selects the next region to search from the list of unsearched regions without taking into account of prior probability of the target distribution. Second strategy is called *probability-based (PB)* approach, where each robot only picks the region with the highest probability as its next objective region. If more than one region has the same highest probability, the robot randomly picks one from them. Third one is called *utility greedy (UG)* approach. Basically, each robot with utility greedy strategy defines its own utility function based on (6), and picks next searching region with the highest utility value. If more than one region has the same highest utility values, the robot randomly picks one from them.

6.2. Simulation Environment and Setups

To evaluate the performance of above approaches, RS, PB, UG, GT, and GT_buf, simulation experiments are carried out on a *Player/Stage* simulator with a network simulator *ns-2*. *Player/Stage* simulator is used to simulate the control and sensing of the robots, and *sns*-2 simulator is used for wireless communication between the robots.

We carried out our search experiment with a team of two homogeneous Pioneer 3DX mobile robots. Each robot is equipped with a pan-tilt-zoom CCD camera, a laser range finder and one front sonar ring and one back sonar ring. The camera sensor is utilized for target detection, the laser range finder is for the range estimation and robot localization, and sonar is for obstacle avoidance. There are three targets scattered in the environment, which can



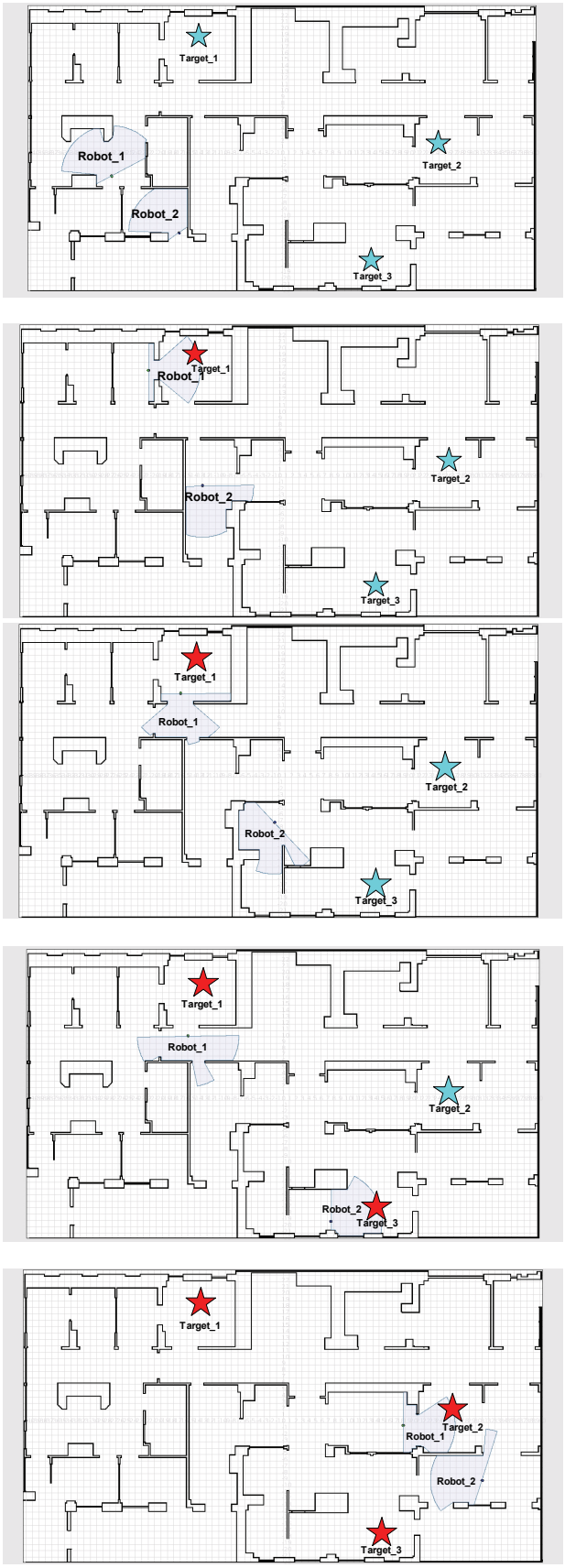


Fig. 3. Snapshots of two robots searching for three targets in a 25-region simulation environment using a Player/Stage robot simulator.

be easily identified from the green color. All robots move at a predefined constant speed.

The Adaptive Monte-Carlo Localization algorithm [Fox et al., 1999] is used for the localization method. Each robot has its own global path planner, which is wavefront based path planner, and local obstacle avoidance algorithms, where the vector filed histogram algorithm is adopted here.

A set of snapshots of one searching experiment using game theory approach in a configuration with 25 regions are shown in Fig. 3, where three stationary targets are distributed in different regions. Initially, two robots start from the left corner. Then robot 1 detected target 1 first based on its own decision at each event-driven time step. When the target was detected, the color of target was changed from green to red. The prior probability of the detection region was distributed to unsearched regions. Target 2 was then detected by robot 2, and eventually both robots move towards Target 3, and robot 1 detected it first. The searching task stops when all three targets are detected.

6.3. Simulation Results

To evaluate the searching performance in a scalable environment, different simulation configurations are applied, where each configuration has different number of regions including 6, 10, 15, 20, 25, 30, and 40. Each searching approach runs 35 times at each configuration. At each configuration, three targets are distributed according to a priori probability map. The simulation results of mean value and the corresponding variance of searching times are shown in Fig. 4.

It is obviously that GT_buf outperforms all other methods. The searching time of UG and GT are much less than that of PB. The reason for this is because the travel and searching time was ignored in the latter case. RS has the worst performance since neither priori probability map nor the travel/searching time is considered.

It can be seen that the performance of GT is competitive with UG. This scenario mainly depends on the procedure design of the GT in the simulation. As we discussed in Section 5, for GT, if one robot is busy and the other one is free, the free robot makes its decision only based on the utility value instead of starting a new game. The motivation for this simplified procedure is to reduce the computational time. If this case happens very often, then the overall performance of GT tends to be close to that of UG. The GT_buf overcomes this problem by carrying on the decision making and searching procedures in parallel while taking advantage of better decisions from game theory. Sometimes GT gets even worse performance than UG especially in the case with more regions, this is because GT takes longer time to make decision than that of UG method.

In a real world scenario, it is difficult to obtain an accurate priori probability map especially in USAR situations where the variance may be very large. To explore the system robustness with respect to priori probability map variations, another set of simulations are implemented. 35 runs are conducted for each searching

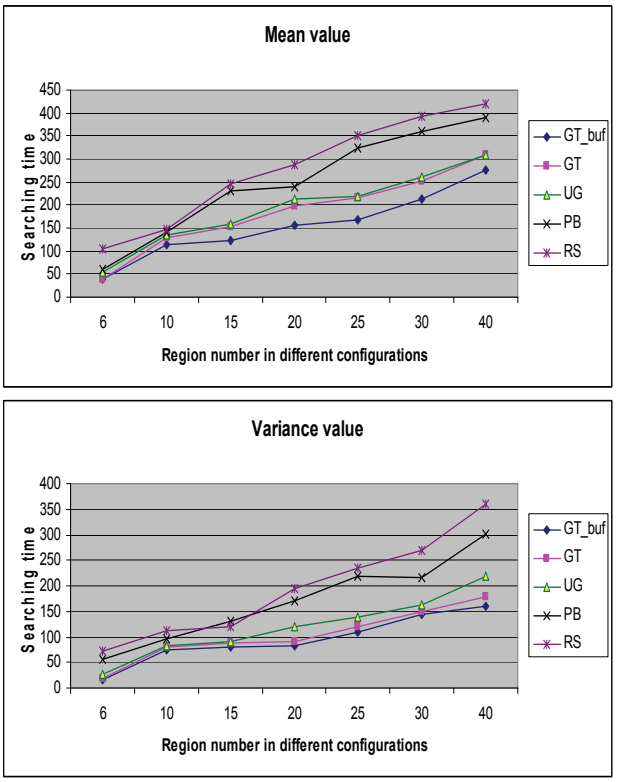


Fig. 4. Searching time (mean and variance) with different configurations

approach, where the target is distributed in the searching regions with the variations of priori probability map ranging from 10% to 50%. Simulation results in the 10-region configuration are shown in Fig. 5.

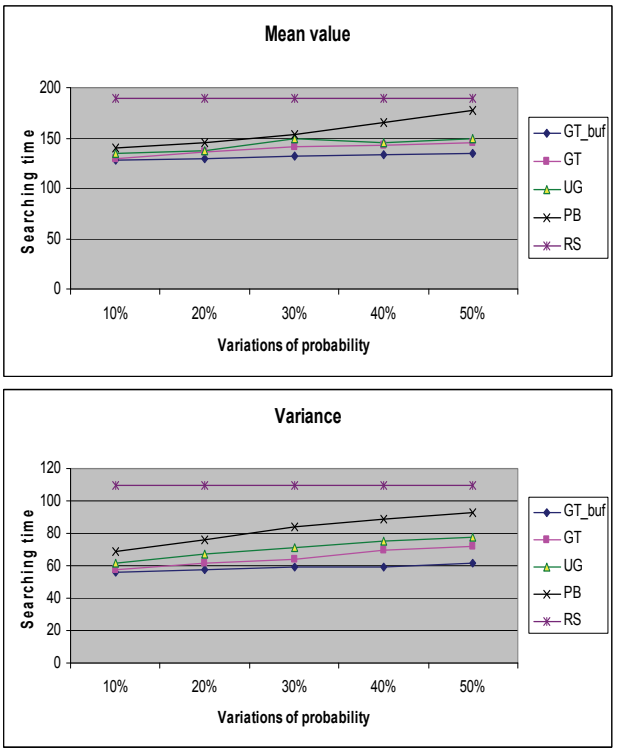


Fig. 5. Searching time (mean and variance) with different variations in probability of target distribution in a 10-region configuration

It can be seen that PB is very sensitive to the probability variation since the probability is the only criteria for the robot to make searching decisions. The probability variation has no effect on the RS at all, which is easily to be understood. The GT and UG are much more robust than PB, where both GT and GT_buf beat UG in both mean searching time and variance. This indicates that the game theory is more robust to handle environmental uncertainty compared to utility greedy approach, although we have to pay the penalty of more computational time for game strategy.

When an obstacle is detected on the way to the robot's destination, different approaches would have different behaviors. (1) For PB, the robot has to dynamically replan the path to the original higher probability region even though the travel cost is extremely high with the new path; (2) For UG, a new decision has to be made considering the travel cost, therefore, different region may be selected; (3) For GT and GT_buf, a new game may be started to adapt to changed utility values.

To evaluate the robustness of the proposed approaches, 35 simulation runs have been conducted in a 25-region environment, where 5 obstacles are randomly distributed within the searching area at each run. The simulation results are shown in Fig. 6. It can be seen that PB is very vulnerable to the environmental change. Although the mean searching time of RS is relatively high, RS is very robust to the environmental change. The GT and UG approaches have similar performance in terms of mean and variance, which have been explained in previous sections. Comparing with GT and UG approaches, GT_buf approach takes longer because it may need to restart new games whenever utility values have been changed due to new obstacles, however, GT_buf is more robust in terms of variance values.

To investigate the effect of different initial positions of robots on the searching performance, eight different position pairs for the two robots are selected. 35 runs for each pair are conducted using GT_buf in a 10-region configuration with accurate prior probabilities of target distribution. The simulation results are shown in Fig. 7. It can be seen that the searching time may differ

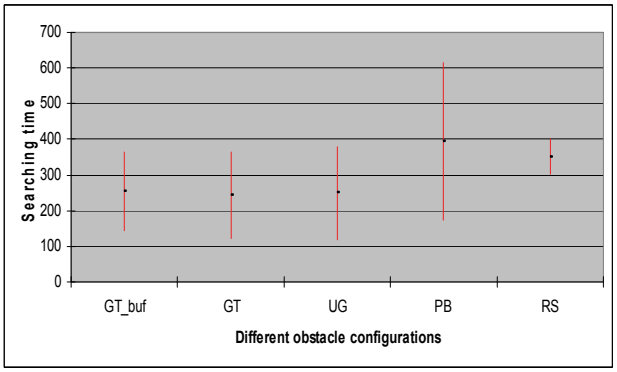


Fig. 6. Searching time with different obstacle distributions in a 25-region configuration, where black dots represent the mean values and red lines denote the variance values.

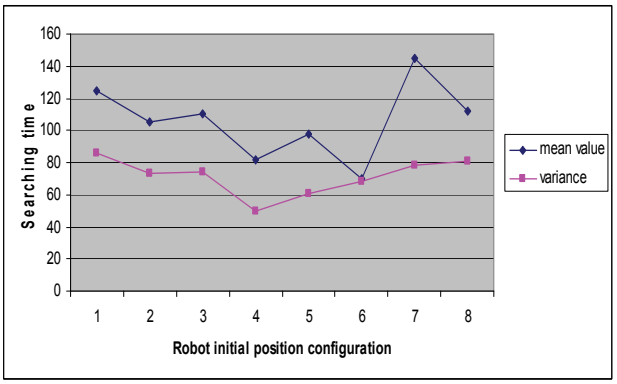


Fig. 7. Searching time (mean and variance) with different initial positions of robots

significantly with different initial position even though the same searching strategy is applied. Since the utility value not only depends on the probability of the target distribution, but also on the travel time to the objective room, if the robots initially happen to be close to the room with higher probability of target, the searching time would be less than other cases.

6.4. Discussion

Although we only use two robots to verify the proposed game theory based method in the simulation, this method can be extended to a large-scale robot system, as shown in the equations in Section 5. To reduce the computation complexity in a large-scale system, three mechanisms are applied: event-driven discretization, one-step dynamic programming, and a decision-buffer method. By using one-step dynamic programming, we can reduce computation complex significantly since only current states will be used to contribute to next time step prediction instead of the all the previous steps.

With the even-driven discretization, new games are only initiated if one robot finishes its current searching area. However, in a large-scale system, such events may happen very often, which may lead to the robots to keep making decisions.

To address this issue, a decision-buffer based method is applied. In this method, during the decision making on a new game, the robots can still focus on their current unfinished tasks, and the future game decisions will be stored in the robots' own buffers. In this way, searching and decision making can be conducted in parallel. When a robot finishes its current task, it would check its buffer for next decision. Only when a robot finishes its current task and there is no further decision in its buffer, a new game calculation has to be initialized. Compared to the dynamic equation programming based method, the computation complexity is increased linearly with the size of the system, which means it can scale well to a large-scale system.

7. Conclusions

A game-theory based strategic searching approach is proposed in this paper to cooperate a multi-robot system in

a searching task. To take the interaction between the robots into consideration, a dynamic programming equation is applied to estimate the utility function, where not only a priori probability map but also the travel costs are considered, as well as other robots' current decisions. Based on this utility function, a utility matrix can be calculated for a N-robot nonzero-sum game, where both pure Nash Equilibrium and mixed-strategy equilibrium can be applied to guide the robots to make their decisions.

To improve the real-time performance of the game-theory based approach, three different mechanisms have been proposed in this paper. First, event-driven discretization method is utilized to reduce the unnecessary probability map update, game restart, and communication cost. Second, instead of using full-length dynamic equation to calculate the utility, one-step dynamic equation is applied to reduce the computation complexity significantly. Third, a decision-buffer mechanism is proposed to cut down unnecessary game decision making procedures. Meanwhile this mechanism makes it possible for robots to conduct searching and decision making in parallel, which naturally leads to better search performance.

Comparing to other heuristic searching strategies, such as random selection, probability-based, and utility greedy approaches, the simulation results demonstrated that the proposed game-theory based approach has better performance and is more robust to handle the environmental uncertainty.

Although homogeneous robots are utilized in our simulation, the proposed algorithm can easily be extended to the heterogeneous robots with different moving speeds and local sensing capabilities by setting up different travel and covering time for each robot.

Our future research includes two levels. First, extend this game-theory based approach to a real-world applications with physical mobile robots. Since each robot has its own on-board processor and can execute in parallel, the computational time of this approach would not be a big issue in a small team of real-world robots. However, as is known, the computational complexity would become intractable with a large-scale robot team using the game-theory based approach. Therefore, at second level, we will investigate the scalability of this approach and develop new models to reduce the computational complexity in our future work.

8. References

Baeza-Yates, R., Culberson, J., and Rawlins. G., Searching in the plane, Informationand Computation, 106:234–252, 1993.

Bertuccelli, L. F., and How J. P., Robust UAV search for environments with imprecise probability maps, IEEE Conference on Decision and Control, 2005.

Bertuccelli, L. F., and How J. P., Search for dynamic targets with uncertain probability maps, Proceedings of the 2006 American Control Conference, Minnesota, USA., 2006, pp. 737-742.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. F., Coordinated decentralized search for a lost target in a Bayesian world, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

Eagle, J. N., Optimal search for a moving target when the search path is constrained, Operations Research, 32(5): 1107-1115, 1984.

Eagle, J. N. and Yee, J. R., An optimal branch and bound procedure for the constrained path, moving target search problem. Operations Research, 38(1):110-114, 1990.

Fox, D, Burgard, W., Dellaert, F., and Thrun, S.. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *AAAI*. 1999.

Harsanyi, J. C. and Selten, R., A general theory of equilibrium selection in games, the MIT Press, Cambridge, Massachusetts, 1998.

Hespanha, J. P., Prandini, M., and Sastry, S., Probabilistic pursuit-evasion games: a one-step Nash approach. In proc. of the 39th conf. on decision and control, vol. 3, Dec. 2000.

Jin, Y., Minai, A., and Polycarpou, M., Cooperative real-time search and task allocation in UAV teams, IEEE Conference on Decision and Control, 2003.

Kao, M. Y., Reif, J. H., and Tate, S. R., Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem, In Proc. 4th ACM-SIAM Sympos. Discrete Algorithms, pages 441–447, 1993.

Kok, J. R., Spaan, M.T.J., and Vlassis, N., Multi-robot decision making using coordination graphs, In Proceedings of the International Conference on Advanced Robotics (ICAR), pp. 1124–1129, Coimbra, Portugal, June 2003.

Lau, H., Huang, S., and Dissanayake, G., Optimal search for multiple targets in a built environment, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05), Edmonton, Alberta, Canada, August 2005.

LaValle, S. and Hutchinson, S., Path selection and coordination for multiple robots via Nash equilibria, Proc. IEEE Int. Conf. Robot and Automation, 1994, pp. 1847-1852.

L'opez-Ortiz1, A., and Schuierer, S. Online parallel heuristics and robot searching under the competitive framework, 8th Scandinavian Workshop on Algorithm Theory, Turku, Finland, July 3-5, 2002.

Murphy, R. R. Biomimetic search for urbane search and rescue, IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2073-2078, 2000.

Skrzypczyk, K., Game theory based target following by a team of robots, Fourth International Workshop on Robot Motion and Control, June 17-20, 2004.

Stewart, T. J., Searching for a moving target when searcher motion is restricted. Computers and Operations Research, 6:129-140, 1979.

Vidal, R, Shakernia, O., Kim, H.J., Shim, D.H., and Sastry, S. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation, IEEE Trans. on Robotics and Automation, Vol. 18, No. 5, Oct. 2002, pp.662-669.