Opportunistically Cooperative Neural Learning in Mobile Agents

Yanli Yang, Marios M. Polycarpou, Ali A. Minai
ECECS Department
University of Cincinnati
Cincinnati, OH 45221-0030

Abstract – Searching a spatially extended environment using autonomous mobile agents is a problem that arises in many applications, e.g., search-and-rescue, search-and-destroy, intelligence gathering, surveillance, disaster response, exploration, etc. Since agents such as UAV's are often energy-limited and operate in a hostile environment, there is a premium on efficient cooperative search without superfluous communication. In this paper, we consider how a group of mobile agents, using only limited messages and incomplete information, can learn to search an environment efficiently. In particular, we consider the issue of centralized vs. decentralized intelligence and the effect of opportunistic sharing of learned information on search performance.

I. INTRODUCTION

Algorithms for searching an extended environment have been studied extensively [6, 5, 1], but recent advances in autonomous vehicle technology, robotics, wireless communication and miniaturization have given the problem a new aspect: Cooperative search by a population of autonomous agents [3, 2, 4]. Typically, the search involves covering a region, locating and identifying potential targets or situations that require a response. This paradigm is applicable to many types of missions, including search and destroy, search and rescue, mine-clearing, autonomous munitions deployment, exploration, prospecting, etc. The basic idea is for a group of N agents to explore the environment while satisfying three requirements:

- **Speed:** The search should be as fast as possible.
- **Coverage:** The entire environment must be covered.
- **Efficiency:** Scarce resources (e.g., energy) must be conserved.

While agents can certainly search the environment without cooperation, the search can be made much more efficient by using cooperation to minimize duplicated effort. However, the control algorithms for achieving such cooperation can be difficult to design.

The problem of cooperative search is essentially one of planning efficient search paths — paths that yield a maximum of new information. If each agent in the system has access to the current coverage status of the environment and the positions and plans of all other agents, it can, in theory, plan the best path for itself. However, in realistic systems, this ideal condition is violated in two ways. First, given the extent of the environment and the limited energy resources of the agents, it is unreasonable to expect an agent to be aware of all others. Typically, it would only have knowledge of those in its close proximity. Second, since all agents make their decisions simultaneously, it is logically impossible for each of them to know the current plans of all others. But the plans still interact. Thus, each agent must make its plan based on incomplete and/or outdated information about the plans of others — a situation sometimes referred to as "bounded rationality" in economics.

We have previously reported on a recursive approach to cooperative search using multi-objective cost functions and q-step path planning [4]. In this paper, we consider a simple version of the search problem using a discretized cellular space with agents that move synchronously with a constant speed. In this framework, agents may represent unmanned air vehicles (UAV's) searching a geographical region for a target. Such UAV's generally have limited communication and turning capabilities. In order to search the environment efficiently, each agent needs to predict the state of its search neighborhood in the near future. Agents do this using feed-forward neural networks trained by a reinforcement learning (RL) algorithm [7]. Such algorithms have been used successfully in robotics and multi-agent systems [8]. The goal of the research reported here is to explore two issues:

- Comparing the performance of a *centralized learning (CL)* approach, where all agents use the same neural network for their predictions, and a *decentralized learning (DL)* approach, where each agent has its own, independently trained network.

- Determining whether opportunistic copying of superior predictors by unsuccessful agents in the DL scenario leads to improvement in performance and speed of learning.

The rationale behind this study is as follows. One would expect that, in a homogeneous environment, a centralized "brain", being trained with information from all agents, would be better than several "mini-brains", each trained on the limited experience of a single agent. However, the CL approach has several drawbacks: 1) Since all agents rely on the same neural network, the time to obtain a prediction increases with the number of agents; 2) All agents are forced to follow the same prediction model, thus precluding the possibility of better models emerging; 3) Constant communication between agents and the central network is needed, thus wasting energy; and 4) The approach is not robust, since error or malfunction in the central network can disrupt the whole system. Also, the CL approach breaks down completely in non-homogeneous environments. The DL approach, on the other hand, implicitly explores the space of possible predictive models as each agent builds its own network. However, the information used in training each agent's network is necessarily limited to the agent's own experience. One possible way around this is to have agents occasionally compare their models, and have less successful agents adopt the models of more successful ones with some *copying probability.* We call this the *opportunistic cooperative learning (OCL)* approach to prediction because agents use the opportunity afforded by proximity to improve performance. This essentially sets up a guided random search in model space — similar to that performed by simulated annealing and genetic algorithms.

In this paper, we consider how the copying probability affects the performance of the system relative to the CL and DL methods. While it is possible to envision several models for the copying probability, for simplicity we focus only on the case when it is constant. Results for more sophisticated models will be reported in the future.

II. PROBLEM DEFINITION

The *environment* consists of a $L_x \times L_y$ grid with periodic boundary conditions. Each grid position is termed a *cell.* Each cell (x, y) has an associated *certainty* value, $z_{x,y} \in [0, 1]$, representing the degree to which it has been searched. There may be locations with a pre-specified certainty of 1, indicating that they are of no interest for the search. The certainties for all locations in the environment are recorded on a continuously updated *certainty map*, $z_{x,y}(t)$, which is observable by all agents. This centralized map is used only for simplicity, since our goal is to compare different learning policies. In future papers, we will consider decentralized methods for the construction and use of certainty maps.

The certainty for each cell is initialized to a value that reflects *a priori* information about that location. A visit by one agent to cell (x, y) at time t increases certainty as

$$\begin{aligned} z_{x,y}(t+1) &= z_{x,y}(t) + 0.5[1 - z_{x,y}(t)] \\ &= 1 - 0.5[1 - z_{x,y}(t)] \end{aligned}$$

i.e., the increment in certainty is one half of the remaining uncertainty. If m agents visit the cell at time t, the certainty is updated as

$$z_{x,y}(t+1) = 1 - 0.5^m[1 - z_{x,y}(t)]$$

Thus, the incremental benefit for each agent at (x, y) diminishes by a factor of 0.5. The change in certainty is incorporated into the certainty map immediately.

The *reward* for entering cell (x, y) at time $t + 1$ is the increment in certainty caused by that entry shared equally by all entering agents. Thus, if m agents enter cell (x, y) at time $t + 1$, each agent, i, gets reward

$$\rho_i(t+1) = [z_{x,y}(t+1) - z_{x,y}(t)]/m$$

A total of N identical agents are involved in a continuous search of the environment. Their goal is to increase the total certainty over the environment as rapidly as possible. To this end, each agent attempts to satisfy two objectives:

- **Search Reward Maximization:** Obtaining as large a search as possible (on average).
- **Cooperation:** Avoiding simultaneous exploration of the same location by more than one agent.

These objectives guide the agents in planning their paths through the environment.

Agent Dynamics and Planning:
At time t, agent i has grid position $(x_i(t), y_i(t))$, and can be in one of eight possible orientations, $o_i(t)$: 0 (north), 1 (northeast), 2 (east), 3 (southeast), 4 (south), 5 (southwest), 6 (west), and 7 (northwest). Each agent plans its path q steps ahead of its current location, adding a new move at each time-step [4]. For this report, we use $q = 2$. Thus, at time-step t, the agent selects its position for $t+2$, the position for $t + 1$ having already been selected at step $t - 1$. At time-step t, the agent executes an *action* comprising the following three steps:

1. It chooses a new orientation, $o_i(t+2) \in \{o_i(t+1) - 1, o_i(t+1), o_i(t+1) + 1\}$ *mod* 8, i.e., the new orientation can change by at most one step. Note that $o_i(t+1)$ is known from step $t - 1$.

2. It then designates the neighbor of $(x_i(t+1), y_i(t+1))$ facing orientation $o_i(t+2)$ as $(x_i(t+2), y_i(t+2))$.
3. Finally, it moves to grid location $(x_i(t+1), y_i(t+1))$ with orientation $o_i(t+1)$.

This essentially means that, at every step, the agent either continues to move in the same direction as before or changes course to left or right by 45°, giving it three possible choices for step $t+2$. We designate these by $l_i = (x_i^l, y_i^l)$ (left), $f_i = (x_i^f, y_i^f)$ (front), and $r_i = (x_i^r, y_i^r)$ (right). Figure 1 shows this graphically for various orientations. Note that l_i, f_i, and r_i depend on time t, but we omit this in the notation for clarity. Limiting the agents' turning capability to 45° reflects the limitations often present with UAV's.

If agent i can move to a cell in n steps, the cell is said to be a *n-target* for i. The set, $C_n(x, y, t)$ of all agents that have a cell (x, y) as a n-target at time t is said to be that cell's *n-competitor set*.

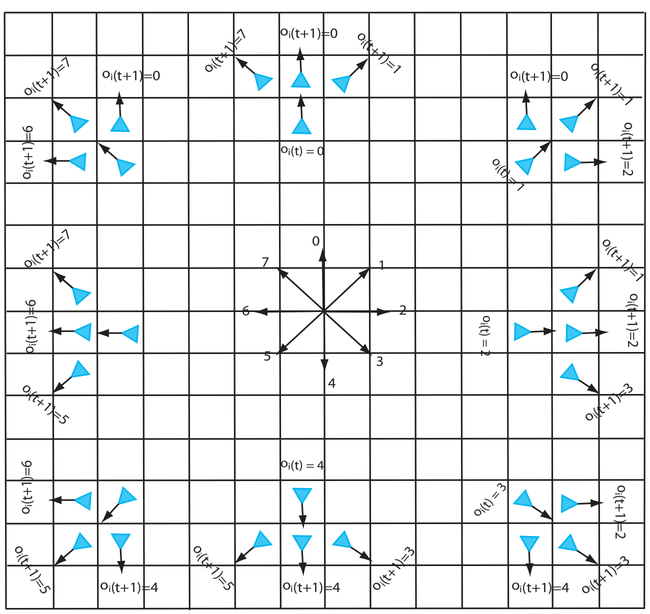

Figure 1: Possible move choices for agents in all 8 orientations. Triangles indicate agents and arrows are the orientations.

III. METHOD

The main task for each agent at time-step t is to choose one of three moves for $t+2$ given the already fixed move for $t+1$. This is done by estimating (predicting) the expected reward for each 2-step target cell and selecting the one with the best payoff. We consider two methods for estimating reward:

Greedy Estimation: In this case, the agent, i, assumes that it will be able to capture the entire reward available at target location k ($k \in \{l_i, f_i, r_i\}$), thus estimating

$$\hat{\rho}_i(t+2) = 0.5[1 - z_k(t)]$$

where $z_k(t)$ is the certainty of cell $k = (x_i^k, y_i^k)$, $k \in \{l_i, f_i, r_i\}$. This naive estimate ignores two things. First, it is possible that other agents may visit k at step $t+1$, thus reducing the reward available to i at $t+2$. Second, other agents may enter k at $t+2$, thus diluting the reward i receives. We use the naive estimate primarily to determine a reasonable reference baseline for system performance.

Cooperative Estimation: This estimate takes into account both the factors ignored by the greedy estimate. It is determined as follows. Suppose $\nu_k(t+1)$ agents occupy cell k at $t+1$ and $\nu_k(t+2)$ (including i) at step $t+2$. Then we get:

$$z_k(t+1) = 1 - 0.5^{\nu_k(t+1)}[1 - z_k(t)]$$

and

$$z_k(t+2) = 1 - 0.5^{\nu_k(t+1)+\nu_k(t+2)}[1 - z_k(t)]$$

The reward obtained by i in this case would be:

$$\begin{aligned} \rho_i(t+2) &= \frac{1}{\nu_k(t+2)}[z_k(t+2) - z_k(t+1)] \\ &= \frac{\zeta}{\nu_k(t+2)}[1 - z_k(t)] \end{aligned} \quad (1)$$

where

$$\zeta = 0.5^{\nu_k(t+1)}[1 - 0.5^{\nu_k(t+2)}]$$

Thus, the reward is determined by three factors: 1) Current certainty of k, $z_k(t)$; 2) Occupancy of k at $t+1$, $\nu_k(t+1)$; and 3) Occupancy of k at $t+2$, $\nu_k(t+2)$. Of course, $z_k(t)$ is known from the certainty map and $\nu_k(t+2)$ cannot possibly be known. For $\nu_k(t+1)$, two cases are possible:

- **Case I, Communicative Agents:** Since all agents have already decided their move for step $t+1$ at time $t-1$, $\nu_k(t+1)$ is, in fact, already determined at step t. If agents in the neighborhood of i communicate their planned moves to i, it can easily calculate $\nu_k(t+1)$ and $z_k(t+1)$ for each of the target cells.
- **Case II, Secretive Agents:** If agents in i's neighborhood do not communicate their plans to it, i must estimate $\nu_k(t+1)$ based on current information.

In this paper, we currently only consider the case of communicative agents. Thus, only $\nu_k(t+2)$ needs to be estimated for $k \in \{l_i, f_i, r_i\}$. This is done based on six items of information:

1. **Occupancy information:** $[\nu_{l_i}(t+1), \nu_{f_i}(t+1), \nu_{r_i}(t+1)]$.
2. **Competition information:** $[c_{l_i}(t+1), c_{f_i}(t+1), c_{r_i}(t+1)]$, with

$$c_k(t+1) = \frac{1}{\beta}|C_1(x_i^k, y_i^k, t+1)|$$

where $|.|$ denotes cardinality, x_i^k and y_i^k are the coordinates of target cell k, and β is a scaling constant (we use $\beta = 8$).

Together, these two sets of values define the *state* for i, $s_i(t) = [\nu_{l_i}(t+1), \nu_{f_i}(t+1), \nu_{r_i}(t+1), c_{l_i}(t+1), c_{f_i}(t+1) c_{r_i}(t+1)]$. Note that all the c and z values for $t+1$ are available to i at time t because the agents are communicative. We use a neural network consisting of three independent sub-networks to estimate $\nu_{l_i}(t+2)$, $\nu_{f_i}(t+2)$, and $\nu_{r_i}(t+2)$ using $s_i(t)$ as the state input (see Section IV). It should be noted that the state information available to each agent is an extremely incomplete view of the system's state even in the agent's neighborhood. More informative state formulations can be envisioned (e.g., certainty values for all neighbors of target cells), but this increases the complexity of the learning problem.

The predicted value of $\nu_k(t+2)$ is used in Eq. (1) along with the known values of $z_k(t)$ and $\nu_k(t+1)$ to obtain an estimate of the 2-step reward, and the agent chooses the cell that promises the greatest reward. All agents then update their positions synchronously, and each agent receives the appropriate reward. Note that the action taken at step t was chosen at $t-1$.

IV. LEARNING ALGORITHM

As described earlier, the agents use tripartite neural networks for predicting $\nu_k(t+2)$, each subnetwork predicting the 2-step occupancy of one of the target cells. This is accomplished with a Q-learning procedure [9], using the true occupancy values, $\nu_k(t+2)$, which become available at time $t+2$. Essentially, the neural networks learn to produce an increasingly accurate estimate of $\nu_k(t+1)$ given $s_i(t)$. The weights of the neural networks are modified using the Levenberg-Marquardt procedure.

As described earlier, we consider three situations:

Centralized Learning (CL): In this case, there is only one tripartite neural network. All agents communicate their observations, $\nu_k(t+2)$, to this network, which calculated the errors for all its corresponding predictions and uses these for learning.

Decentralized Learning (DL): In this case, each agent has its own tripartite network, trained using its own predictions and observations. There is no copying of networks among agents.

Opportunistic Cooperative Learning (OCL): In this case, each agent maintains a running *reward average*

$$\bar{\rho}_i(t+1) = (1-\alpha_i)\bar{\rho}_i(t) + \alpha_i \rho_i(t+1); \quad 0 < \alpha_i \leq 1$$

which shows how well it has done recently. This is used as a measure of performance for its predictor. When two agents find themselves in neighboring cells, they compare their $\bar{\rho}$ values and the agent with the lower value copies the network and the $\bar{\rho}$ value of the other agent with probability π. Note that $\pi = 0$ corresponds to the DL case.

V. SIMULATION RESULTS

While, in practice, the agents would learn as they search, we have used a two-phase approach to evaluate the performance of the various approaches. In the *training phase*, the system is trained over n_{train} steps, with the neural network weights modified at each step. The training is done using a search in an actual environment, but the certainty values of visited cells are continually reset to 1 after they are visited few time steps, thus creating greater opportunity for learning. The training phase is followed by an *evaluation phase*, during which the trained agents search an environment for n_{eval} steps without any further learning. This shows how the uncertainty about the environment is reduced over time with agents trained by different algorithm. The results for the greedy algorithm are also plotted for comparison.

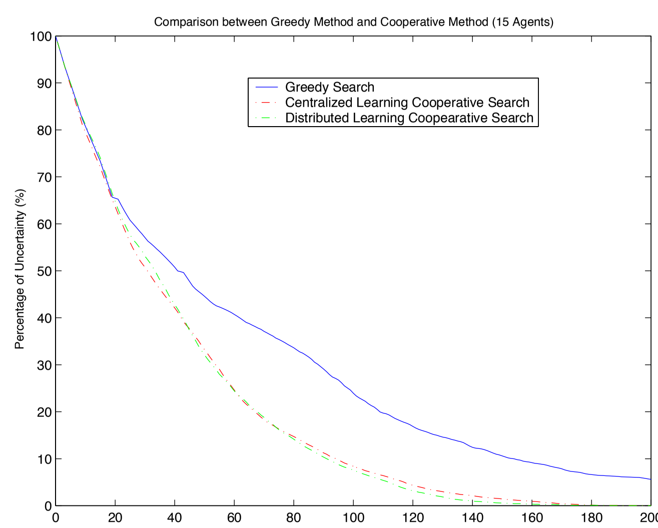

Figure 2: Search performance for 15 agents. The CL and DL algorithms use 100 steps of learning.

Figure 2 shows the time-course of uncertainty reduction when 15 agents trained with various algorithms search a 20×20 environment. Clearly, greedy search performs much more poorly than the cooperative algorithms. This effect is likely to increase with the number of agents, since that also enhances the quality of learning. The figure indicates that both cooperative algorithms perform equally well. However, this is the result of the long training time (100 steps).

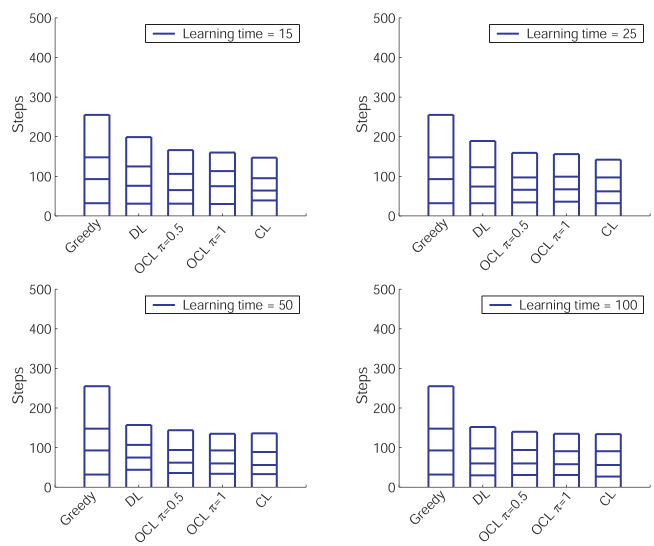

Figure 3: Search efficiency: The increments on each bar indicate the number of search steps needed to reduce uncertainty by 50%, 75%, 90% and 98%, respectively. The system has 15 agents searching a 20×20 environment.

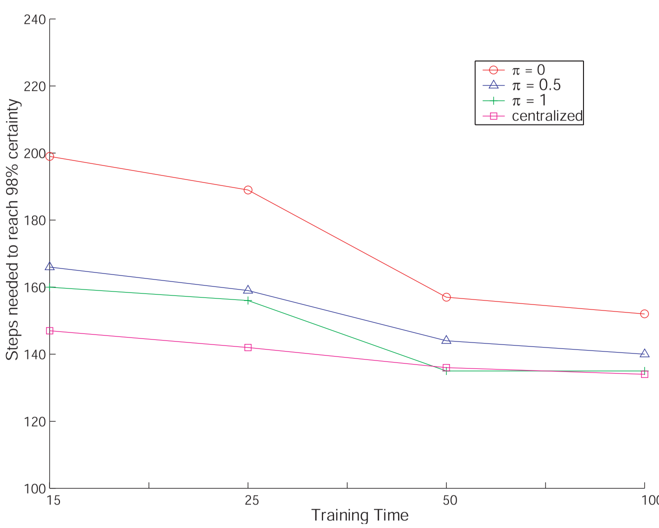

Figure 4: Number of search steps needed to reach 98% certainty as a function of training time.

Figure 3 shows how rapidly uncertainty is reduced by agents trained with different algorithms for 15, 25, 50 and 100 steps. Clearly, the OCL algorithms with $\pi > 0$ learn faster than the DL algorithm and almost as fast as the CL algorithm. This is confimed by Figure 4, which shows the time needed to reach 98% certainty. However, it is apparent that OCL needs about 50 steps of learning before reaching the performance of CL, while DL is consistently worse than both.

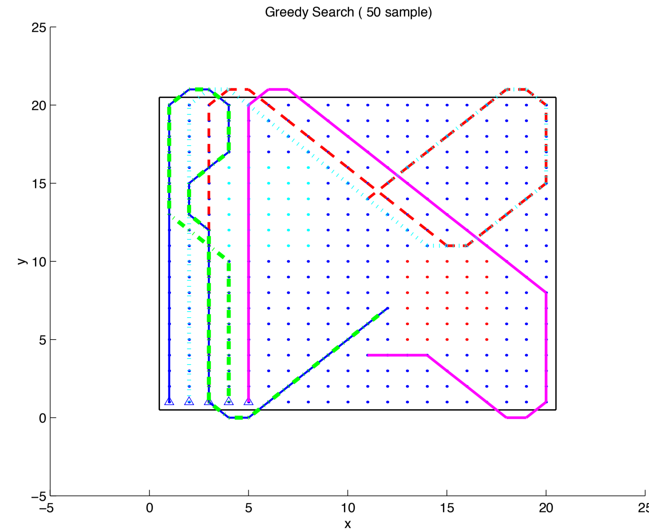

Figure 5: Search paths for agents in a 5 agent system using greedy search. Note that many paths overlap, reducing search efficiency.

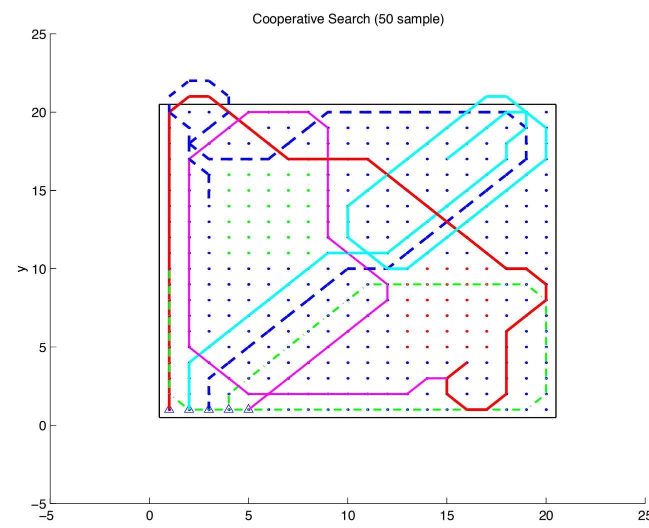

Figure 6: Search paths for agents in a 5 agent system using DL with $\pi = 0.5$.

Finally, Figures 5 and 6 show the actual search paths taken by 5 agents searching a 20×20 environment. It is apparent that greedy agents (Figure 5) tend to follow each other — essentially "picking the crumbs" left by other agents, while cooperative agents (Figure 6) are able to find more diverse search paths.

VI. CONCLUSIONS

In this paper, we have compared the performance of several learning algorithms to train agents conducting cooperative search in an extended environment. The agents learn to predict the behavior of other agents in their neighborhood using neural networks trained via reinforcement learning. We consider the centralized case, where all agents share a neural network, and the decentralized one, where each agents carries its own network. The results indicate that the centralized learning approach — which is problematic from a scalability and efficiency perspective — can be replaced by the decentralized approach if poorly performing agents are allowed to opportunistically acquire the expertise of better performers.

Acknowledgement: This research was supported in part by the DARPA MICA program and a DAGSI/AFRL grant.

References

[1] S.J. Benkoski, M.G. Monticino, and J.R. Weisinger. A survey of the search theory literature. *Naval Research Logistics*, 38:469–494, 1991.

[2] H. Choset and P. Pignon. Coverage path planning: the boustrophedon cellular decomposition. In *International Conference on Field and Service Robotics*, Canberra, Australia, 1997.

[3] S. Goldsmith and R. Robinett. Collective search by mobile robots using alpha-beta coordination. In A. Drogoul, M. Tambe, and T. Fukuda, editors, *Collective Robotics*, pages 136–146. Springer Verlag: Berlin, 1998.

[4] M. Polycarpou, Y. Yang, and K. Passino. A cooperative search framework for distributed agents. In *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*, pages 1–6, 2001.

[5] H.R. Richardson. Search theory. In D.V. Chudnovsky and G.V. Chudnovsky, editors, *Search Theory: Some Recent Developments*, pages 1–12. Marcel Dekker, New York, NY, 1987.

[6] L.D. Stone. *Theory of Optimal Search.* Acadamic Press, New York, 1975.

[7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

[8] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth International Conference on Machine Learning*, pages 330–337, 1993.

[9] C. J. C. H. Watkins. *Learning with Delayed Rewards.* PhD thesis, University of Cambridge, 1989.