PARALLEL STOCHASTIC HILL-CLIMBING WITH SMALL TEAMS

Brian P. Gerkey, Sebastian Thrun

Artificial Intelligence Lab
Stanford University
Stanford, CA 94305, USA
gerkey@ai.stanford.edu, thrun@stanford.edu

Geoff Gordon

Center for Automated Learning and Discovery
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon+@cs.cmu.edu

Abstract   We address the basic problem of coordinating the actions of multiple robots that are working toward a common goal. This kind of problem is NP-hard, because in order to coordinate a system of n robots, it is in principle necessary to generate and evaluate a number of actions or plans that is exponential in n (assuming $P \neq NP$). However, we suggest that many instances of coordination problems, despite the NP-hardness of the overall class of problems, do not in practice require exponential computation in order to arrive at good solutions. In such problems, it is not necessary to consider all possible actions of the n robots; instead an algorithm may restrict its attention to interactions within small teams, and still produce high-quality solutions.

   We use this insight in the development of a novel coordination algorithm that we call *parallel stochastic hill-climbing with small teams*, or *Parish*. This algorithm is designed specifically for use in multi-robot systems: it can run off-line or on-line, is easily distributed across multiple machines, and is efficient with regard to communication. We state and analyze the Parish algorithm present results from the implementation and application of the algorithm for a concrete problem: multi-robot pursuit-evasion. In this demanding domain, a team of robots must coordinate their actions so as to guarantee location of a skilled evader.

1. Introduction

Multi-robot systems have the potential to be far more useful than single robots: multiple robots may perform a given task more efficiently than a single robot, multiple robots may be more robust to failure than a single robot, and multiple robots may be able to achieve tasks that are impossible for a single robot. However, reaching that potential can be extremely difficult, especially in the case where multiple robots make task achievement *possible*, rather than simply *better*. The difficulty arises primarily from the combinatorial possibilities inherent in the problem of coordinating the actions of multiple robots, which is in general NP-hard (Garey and Johnson, 1979). Given a system of n robots and a common goal, it may be necessary to generate and evaluate a number of actions or plans that is exponential in n (assuming that $P \neq NP$).

One common way to attack such a problem is brute-force search in the joint state/action space. That is, treat the multi-robot system as one many-bodied robot and look through the exponentially many possibilities until the right answer is found. Though this approach will produce an optimal solution, it is only viable on simple problems, as the necessary computation quickly becomes intractable as the number of robots and/or the complexity of the problem grows. This fact contradicts the intuition that having more robots available should make a task easier, rather than harder, to solve. Additionally, this approach is undesirable for most robotic applications, because it requires a centralized planner / executive, which precludes local control decisions at the level of an individual robot.

Another, more popular, approach is to treat the multi-robot system as a collection of independent single robots and allow each one to make individual control decisions, irrespective of the other robots' actions. This approach scales very well, as it requires each robot to consider only its own possible actions, the number of which remains constant as the number of robots grows. Unfortunately, this technique will not necessarily produce a good solution. In fact, if the actions of the robots *must* be coordinated in order to achieve a task, then allowing them to simply make individual choices without considering or consulting each other is unlikely to lead to any solution at all.

We believe that between these two extremes lies fertile ground for the development of heuristic multi-robot coordination algorithms that produce good solutions yet scale well with the number of robots. In particular, we suggest that many multi-robot problems can be solved quickly and effectively by allowing the formation of and planning for small teams over short time horizons. That is, rather than considering the possible actions of all n robots or of just 1 robot, consider groups of up to t robots, where $1 \leq t \leq n$, but prefer smaller groups, because they are computationally cheaper to coordinate. In this paper we introduce an algorithm, *parallel stochastic hill-climbing with small teams*, or

Parish, which combines the idea of small teams with the use of heuristics and stochastic action selection. In addition to scaling well and tending to produce good solutions to coordination problems, Parish is easily distributable, and can be executed either on-line or off-line, both of which are desirable properties for multi-robot algorithms.

We have implemented Parish for the problem of *multi-robot pursuit-evasion*, in which a group of robots must work together to search a given environment so as to guarantee location of a skilled mobile evader. This is a difficult problem that clearly requires coordination among the robots (a single robot is only capable of clearing environments that are topologically equivalent to a single hallway). And, unlike more weakly interactive tasks, like foraging, pursuit-evasion occasionally requires very tight coordination between robots in order to make any progress at all. We provide results from tests in simulation of search strategies produced by Parish.

2. Background and related work

The first rigorous formulation of the pursuit-evasion problem is due to Parsons, who restricted his study to the case in which the environment is a discrete graph (Parsons, 1976). Nothing is known about the location or motion of the evader, who is assumed to be able to move arbitrarily fast through the graph. The evader can occupy any edge in the graph; to find the evader, a searcher must walk along the edge occupied by the evader and "touch" the evader. The entire graph is initially *contaminated*, which means that the evader could be anywhere. As the search progresses, an edge is *cleared* when it is no longer possible for the evader to occupy that edge. Should it later happen that the evader could have moved back to a previously clear edge, that edge is said to be *recontaminated*. Using this terminology, the goal of the problem can be restated as follows: find a trajectory for each searcher such that the an initially contaminated graph is cleared.

More recently, a visibility-based version of the pursuit-evasion problem was introduced (Suzuki and Yamashita, 1992), which changed the domain from discrete graphs to continuous polygonal free spaces. Complete algorithms have been described for searchers having either 1 or 2 "flashlights" (Lee et al., 2002), omnidirectional vision (Guibas et al., 1999), and limited field-of-view vision (Gerkey et al., 2004). Randomized pursuit algorithms have also been studied, in both discrete graphs (Adler et al., 2003) and polygonal free spaces (Isler et al., 2003).

3. Algorithm

The *Parish* algorithm coordinates a multi-robot system in a scalable manner by considering the possible actions of not only single robots, but also small

teams of robots. The general form of the algorithm can be summarized as follows:

Algorithm *Parish: Parallel stochastic hill-climbing with small teams*
Input: n robots; multi-robot problem M; maximum team size $t \leq n$; value heuristic $v(q)$;
    probability distribution $P(q)$, $P(q_j) > P(q_i) \Leftrightarrow v(q_j) \geq v(q_i)$
1.   **while** M not done
2.   **do parallel for** each robot s
3.   **do for** $l \leftarrow 1$ **to** t
4.   **do** $Q_l \leftarrow \{q : q$ is a feasible l-searcher plan involving $s\} \bigcup \{\emptyset\}$
5.   Sample $\hat{q}_l$ from Q_l according to $P(q)$
6.   **if** $\hat{q}_l \neq \emptyset$
7.   **then** Execute $\hat{q}_l$
8.   **break**

The *value* heuristic $v(q)$ has two components: a *benefit* heuristic $b(q)$ and a *cost* function $c(q)$. The benefit heuristic b estimates the (possibly negative) marginal benefit (i.e., progress that would be made toward solution) of a given plan. In other words, b estimates the optimal value function, which is unknown (computing the optimal value function is equivalent to solving the original NP-hard problem). If a plan q involves any robots that are currently part of other teams that are engaged in other plans, then $b(q)$ includes an estimate of the (probably negative) benefit that will result from disbanding those teams and halting the execution of those other plans. The function c calculates, in the same units as b, the cost of executing a given plan. This cost can be any salient aspect of the domain that is external to progress, such as distance moved. The *value* of a plan q is then $v(q) = b(q) - c(q)$.

Because the heuristic b is only an *estimate* of the true benefit of a given plan, we cannot always select the highest-valued plan. Such a strategy will, in all but the simplest problems, lead to local maxima of progress from which the system will not escape. Thus we employ a stochastic selection rule: rather than greedily selecting the apparently best plan, we sample a plan $\hat{q}_l$ from the set Q_l of available plans, according to a probability distribution $P(q)$ that prefers higher-valued plans but sometimes selects an apparently worse plan. This technique is commonly used in optimization to escape from local extrema and is in reinforcement learning to balance exploration against exploitation. So robots executing Parish are collectively hill-climbing according to local progress gradients, but stochastically make lateral or downward moves to help the system escape from local maxima.

The exact nature of the selection rule can be adjusted according to the accuracy of the benefit heuristic. If b is known to be a very accurate estimate of the optimal value function, then the highest-valued plan should be selected with accordingly high probability, and vice versa if b is known to be less accurate (of course, if b is very inaccurate, then progress will be slow, and more effort should likely be put toward designing a better heuristic).

Since the robots make plans individually, the computation of the algorithm can easily be distributed across multiple machines, with communication required only to update the state of the problem and to form (or break up) teams. If a good model of the environment is available, then Parish can run off-line, with the robots interacting with this model to produce a plan for later execution. If no good model is available, or if the environment is dynamic, then Parish can run on-line, with the robots interacting with the environment directly. Also, robots will tend select and execute single-robot plans, if good ones can be found, because they do not require breaking up other teams. Thus they will make individual progress as long as possible, until such time as team formation is more beneficial.

3.1 Economic interpretation

As is the case with many multi-agent search algorithms, there is an obvious economic interpretation of Parish. The multi-robot system can be seen as a synthetic economy, in which individual robots can buy the services of other robots. A robot receives (possibly negative) "reward" for making (possibly backward) progress toward the goal. Each robot then selfishly tries to "earn" as much reward as possible. The value, $v = b - c$, that a robot attaches to a plan that it has formulated is the "price" that that robot will "pay" in order to form the team that will help in executing the plan (the robot may offer a price slightly less than v, in order to retain some positive profit). A robot only joins a team when it is offered a sufficiently high price to take it away from its current team, if any. Stochastic plan selection then corresponds to a robot occasionally making a choice that does not maximize its reward, to account for the fact that, because of inaccuracies in prices (i.e., values), strict reward-maximization will not necessarily lead to a solution.

Although this economic interpretation relates our algorithm to previous work in economically-inspired multi-robot coordination approaches (e.g., Gerkey and Matarić, 2002; Dias and Stentz, 2003), we do not find it particularly helpful. Coordination algorithms such as Parish can be understood and clearly stated as instances of distributed search or optimization; economic interpretations can unnecessarily cloud the discussion by introducing misleading analogies between synthetic markets as used by robots and real markets as used by humans.

3.2 Application to multi-robot pursuit-evasion

We now make Parish concrete by explaining how we apply it to the problem of multi-robot pursuit-evasion and stating the resulting algorithm. In the multi-robot pursuit-evasion problem, a team of n robots is required to search an environment (of which a map is provided) so as to guarantee location of a

Figure 1. *The Botrics Obot mobile robot, equipped with a SICK scanning laser range-finder, which has a 180° sensor field.*
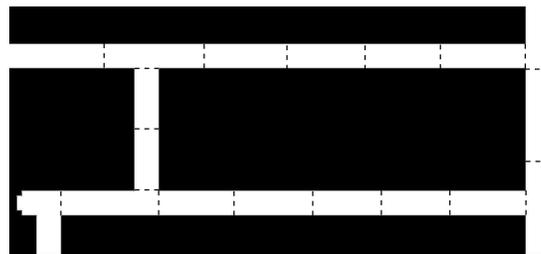
(a)
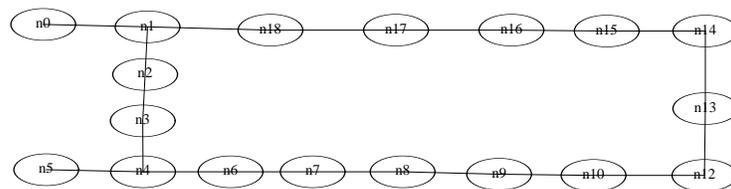
(b)

Figure 2. *An office-like environment, decomposed into convex regions (a) and then transformed into a discrete graph (b).*

skilled mobile evader. The only information available about the evader is its size and maximum speed; no model of its initial position or subsequent trajectory is given. For our purposes, a robot "finds" the evader if the evader is detected within the robot's sensor field. Our robots are each equipped with a scanning laser range-finder that provides a 180° field of view and reliable detection range of approximately 8 meters (Figure 1).

We first transform our problem to an instance of Parsons's discrete graph search problem (Parsons, 1976). This transformation involves decomposing the free space in the given map into finitely many regions such that a single robot can clear a region by standing anywhere on and perpendicular to the region border, while looking into the region. Furthermore, we want to guarantee

that a differential-drive robot with a 180° field of view can move from one border of a region to any other border of the same region and keep the destination border in view along the way. Two necessary and sufficient conditions for the regions are that they each: (i) be convex, and (ii) have no dimension greater than the maximum sensor range (8 meters). For the work presented in this paper, the decomposition was performed manually, but the process could be automated according to visibility constraints (e.g., Guibas et al., 1999; Gerkey et al., 2004). Given such a region decomposition, we construct an undirected graph $G = (V, E)$, where the vertices V are the regions, and the edges E are the borders where adjacent regions meet. An example decomposition and the resulting graph are shown in Figure 2.

We can then apply Parish, stated below, to the graph G, and transform the resulting solution back to the robots' control space, with each move in the graph becoming a move to a region border in the physical environment.

Preliminaries:

- Searcher positions and edge contamination states are stored as labels in the graph.
- The graph, the list of teams, and the list of plans are shared data structures: each searcher has an identical copy of each structure, and a mutual exclusion mechanism is used to ensure consistency when making changes.
- S_i denotes searcher i.
- Given a list L, $L[i]$ denotes the ith element of L.
- A *plan q* specifies a sequence of moves for one or more searchers.
- The *null plan*, denoted $\emptyset$, makes no moves.
- Given a plan q, q.members() returns the set of searchers required to execute q.
- $G' \leftarrow G + q$ denotes the application of plan q to graph G to produce the resulting graph G'.
- Given a team T with n members, to *disband T* is to separate the members of T into n singleton teams, one individual per team.

Algorithm *Parish for multi-robot pursuit-evasion*
Input: Connected, undirected graph G; n searchers placed in G (if initial placement is not given, place them randomly); maximum team size t; value heuristic $v(G, q)$; probability distribution $P(q)$, $P(q_j) > P(q_i) \Leftrightarrow v(q_j) \geq v(q_i)$

```
1.   T ← [ ]  (∗ List of teams ∗)
2.   A ← [ ]  (∗ List of plans ∗)
3.   for i ← 1 to n
4.       do (∗ Start with singleton teams and no plans ∗)
5.           T.append({S_i})
6.           A.append(∅)
7.   while not done
8.       do (∗ Each team decides what to do, in parallel ∗)
9.           parallel for i ← 1 to len(T)
10.              do if A[i] = ∅
11.                  then (∗ No plan, so this team has only one member; call it s ∗)
```

```
12.                              s ← s : s ∈ T[j]
13.                              (∗ Consider teams of increasing size, up to t ∗)
14.                              for l ← 1 to t
15.                                  do (∗ Make some l-searcher plans, but also consider the null plan ∗)
16.                                      Q_l ← {q : q is a feasible l-searcher plan involving s} ⋃ {∅}
17.                                      Sample q̂_l from Q_l according to P(q)
18.                                      if q̂_l = ∅
19.                                          then (∗ We chose the null plan; keep looking ∗)
20.                                              continue
21.                                          else (∗ Assemble the team, maybe disbanding other teams ∗)
22.                                                  for j ← 1 to len(T), j ≠ i
23.                                                      do for r ∈ q̂_l.members()
24.                                                          do if r ∈ T[j]
25.                                                              then Disband T[j]
26.                                                                  T[i] = T[i] ⋃ r
27.                                                  (∗ Store the chosen plan and begin executing it ∗)
28.                                                  A[i] ← q̂_l
29.                                                  G ← G + first step of A[i]
30.                                                  (∗ We have a satisfactory plan; stop looking ∗)
31.                                                  break
32.                          else (∗ We already have a plan, so keep executing it ∗)
33.                              G ← G + next step of A[i]
34.                              if just executed last step of A[i]
35.                                  then (∗ This team has finished its plan; disband it ∗)
36.                                      Disband T[i]
```

4.   Results

We implemented Parish as stated in the previous section and tested it on several environments. The tests were carried out using Stage, a sensor-based multi-robot simulator; experience has shown that results in Stage can be reliably replicated with with physical (indoor, planar) robots (Gerkey et al., 2003). Animations can be found at: **http://ai.stanford.edu/∼gerkey/research/pe/**.

The benefit heuristic b is the (possibly negative) number of regions that would be cleared by executing a given plan. The cost function c is proportional to distance traveled during a given plan, calculated as number of regions traversed. The maximum team size is $t = 2$, and the robots are restricted to making plans that move each team member once. Specifically, each robot S_i only considers plans of the following form:

- (team size 1) Move S_i to an adjacent region.

- (team size 2) Move another robot S_j ($i \neq j$) to cover the region currently covered by S_i, then move S_i into an adjacent region.

The stochastic selection rule is ϵ-greedy, in which the highest-valued plan is selected with probability $(1 - \epsilon)$, and otherwise one of the remaining options is chosen with uniform probability. For the results presented here, $\epsilon = 0.1$. We
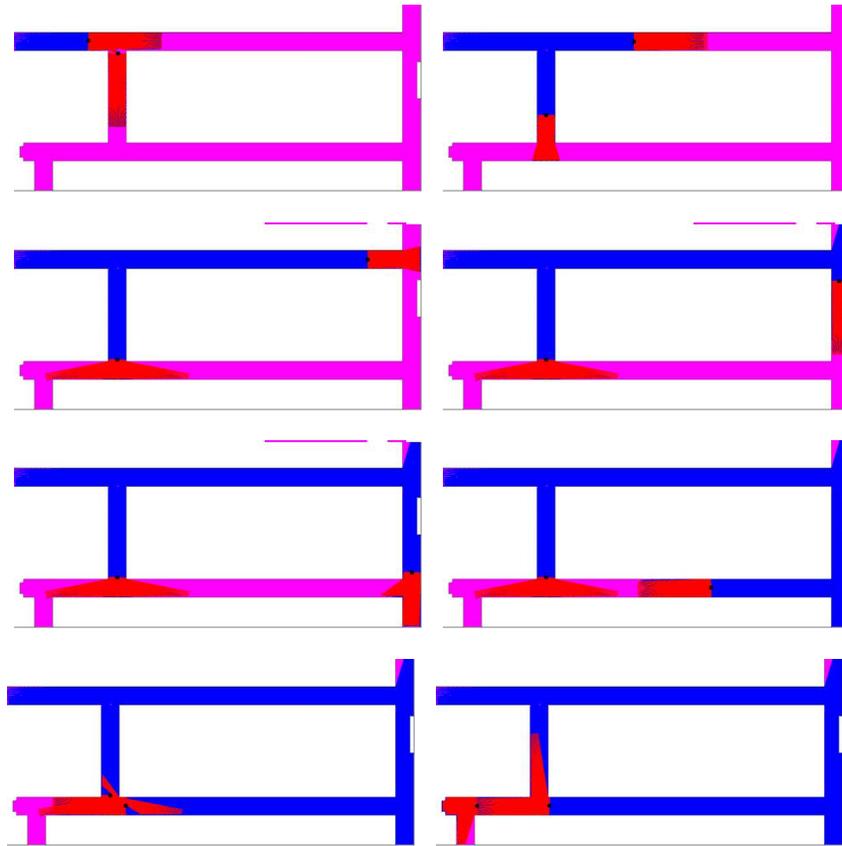
Figure 3.   (In color where available). Two robots searching an office-like environment. Black circles represent robots; blue areas are clear; red areas are in view; and purple areas are contaminated (i.e., the evader may be hiding there).

assume the environment is static, and so are free to run Parish off-line, then execute the resulting plan with the simulated robots.

Interestingly, adding just this limited and myopic coordination is sufficient to produce good solutions. For example, shown in Figure 3 are snapshots from a run with 2 robots in an office-like environment. As can be seen in that figure, the robots cooperate to clear the environment quite efficiently, without allowing recontamination. In fact, Parish reliably produces solutions for this and similar environments that are optimal in the total path length. (we compute optimal solutions using brute-force A* search in the joint action/state space of all the robots).

The effect of small-team coordination can be clearly seen in Figure 4, taken from a simulation run in which 5 robots work together to clear one floor of
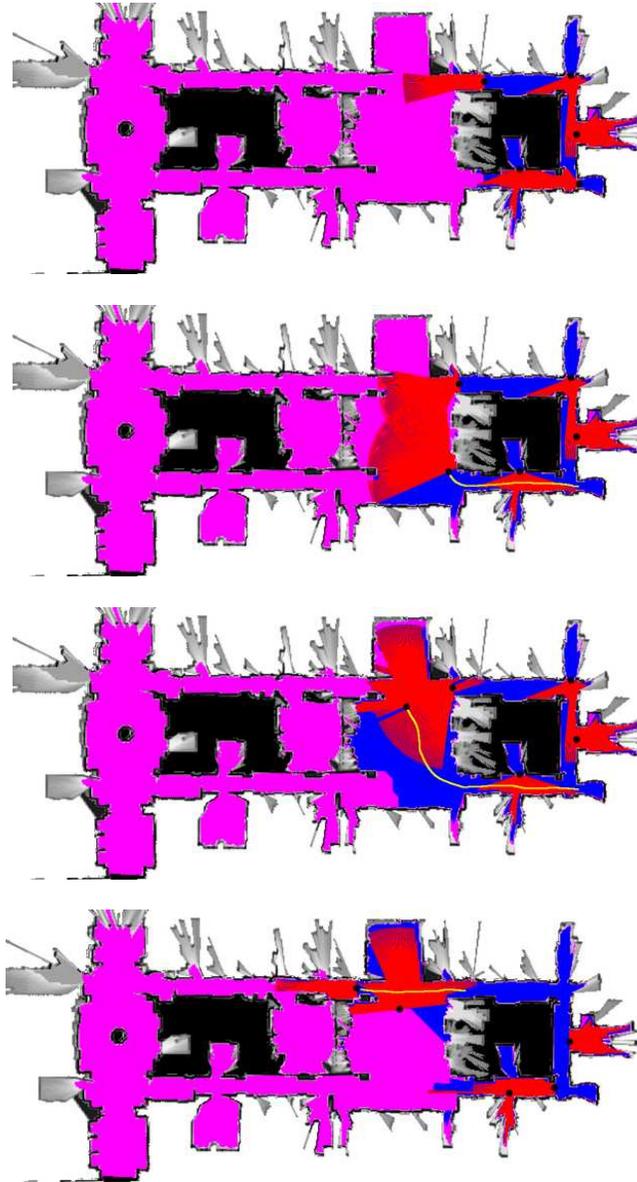
Figure 4. *(In color where available). An example of small-team coordination taken from a test in which 5 robots cleared a large building. As part of a 2-robot plan, the robot that is initially in the lower right corner moves up and left to block the central open area so the robot that another robot can move left and keep searching.*

an office building, using a sensor-based map. In this sequence, a 2-robot plan calls for the robot initially at the lower right to move up and block the central
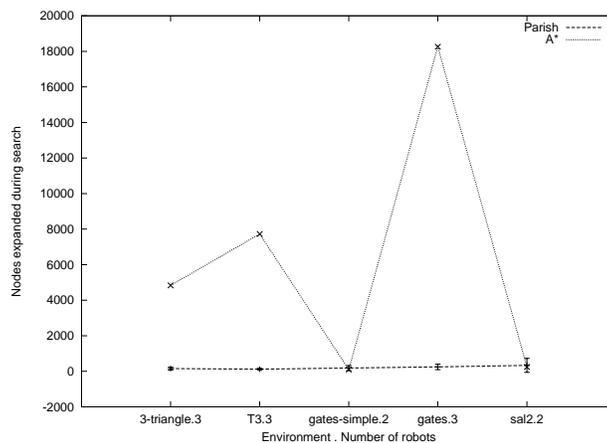
(a)
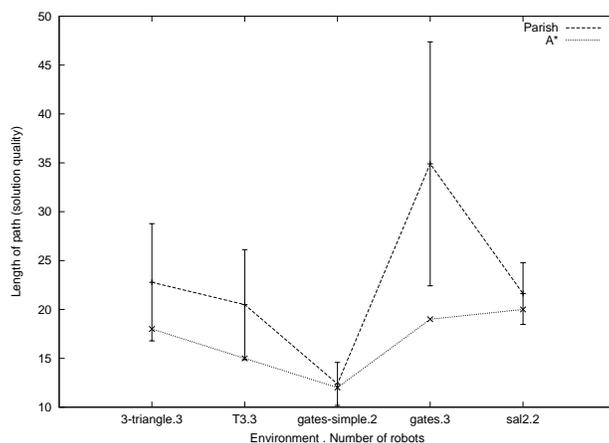
(b)

Figure 5.　　*Comparison of Parish and A* in planning pursuit strategies in various environments. Shown in (a) is the number of nodes expanded during the search, and in (b) is the length of the solution found (smaller is better in both cases). Results for Parish, which is stochastic, show the experimental mean and standard deviation, computed from 100 runs in each environment.*

open area so that another robot can move left and keep searching. Without such interactions, the robots are not capable of clearing this complex environment.

5.　　Summary and future work

We introduced the *Parish* algorithm, which allows for scalable and efficient coordination in multi-robot systems. The key insight of the algorithm is that the

combination of small teams, simple heuristics, and stochastic action selection can be extremely effective in solving otherwise difficult multi-robot problems. Our algorithm is easily distributable and can run on-line or off-line, making it especially suitable for use in physical robots systems. We presented results from simulation that demonstrate the efficacy of Parish in coordinating robots engaged in a pursuit-evasion task.

Our current work on this algorithm follows 3 paths. First, we are moving to physical robots, where Parish will run on-line, and fully distributed. Second, we are rigorously analyzing Parish and comparing it to competitor algorithms, such as non-cooperative greedy, and centralized A*. It will be important to establish the average-case and worst-case performance of Parish, in terms of solution quality and computational requirements (i.e., amount of the search space that is actually explored), as compared to existing alternatives (Figure 5). Finally, we are applying Parish to other multi-robot coordination problems.

References

Adler, M., Räcke, H., Sivadasan, N., Sohler, C., and Vöcking, B. (2003). Randomized Pursuit-Evasion in Graphs. *Combinatorics, Probability and Computing*, 12(3):225–244.

Dias, M. B. and Stentz, A. (2003). TraderBots: A Market-Based Approach for Resource, Role, and Task Allocation in Multirobot Coordination. Technical Report CMU-RI-TR-03-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gerkey, B. P. and Matarić, M. J. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768.

Gerkey, B. P., Thrun, S., and Gordon, G. (2004). Visibility-based pursuit-evasion with limited field of view. In *Proc. of the Natl. Conf. on Artificial Intelligence (AAAI)*, pages 20–27, San Jose, California.

Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proc. of the Intl. Conf. on Advanced Robotics (ICAR)*, pages 317–323, Coimbra, Portugal.

Guibas, L. J., Latombe, J.-C., LaValle, S. M., Lin, D., and Motwani, R. (1999). A Visibility-Based Pursuit-Evasion Problem. *Intl. J. of Computational Geometry & Applications*, 9(4 & 5):471–493.

Isler, V., Kannan, S., and Khanna, S. (2003). Locating and capturing an evader in a polygonal environment. Technical Report MS-CIS-03-33, Dept. of Computer Science, Univ. of Pennsylvania.

Lee, J.-H., Park, S.-M., and Chwa, K.-Y. (2002). Simple algorithms for searching a polygon with flashlights. *Information Processing Letters*, 81:265–270.

Parsons, T. (1976). Pursuit-evasion in a graph. In Alavi, Y. and Lick, D., editors, *Theory and Applications of Graphs*, Lecture Notes in Mathematics 642, pages 426–441. Springer-Verlag, Berlin.

Suzuki, I. and Yamashita, M. (1992). Searching for a mobile intruder in a polygonal region. *SIAM J. on Computing*, 21(5):863–888.