Distributed Path Planning for Mobile Robots using a Swarm of Interacting Reinforcement Learners

Christopher M. Vigorito
Dept. of Computer Science
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA, United States
vigorito@cs.umass.edu

ABSTRACT

Path planning for mobile robots in stochastic, dynamic environments is a difficult problem and the subject of much research in the field of robotics. While many approaches to solving this problem put the computational burden of path planning on the robot, *physical path planning* methods place this burden on a set of sensor nodes distributed throughout the environment that can communicate information to each other about path costs. Previous approaches to physical path planning have looked at the performance of such networks in regular environments (e.g., office buildings) using highly structured, uniform deployments of networks (e.g., grids). Additionally, these networks do not make use of real experience obtained from the robots they assist in guiding. We extend previous work in this area by incorporating reinforcement learning techniques into these methods and show improved performance in simulated, rough terrain environments. We also show that these networks, which we term SWIRLs (Swarms of Interacting Reinforcement Learners), can perform well with deployment distributions that are not as highly structured as in previous approaches.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Experimentation, Performance

Keywords

Physical path planning, Robot Navigation, Reinforcement Learning, Swarm intelligence

1. INTRODUCTION

Navigation and path planning for mobile robots in stochastic, dynamic environments are fundamental and difficult problems for which many approaches have been developed. The majority of these approaches place the burden of computing a solution to each of these problems on the robot itself. For the local navigation (obstacle avoidance) problem, i.e. navigation in which obstacles and the goal destination are within sensing range of the robot, some egocentric solutions have been developed that work very well in dynamic, stochastic environments with minimal computation [3]. The global navigation (path planning) problem, however, in which potential obstacles and the goal destination are not immediately observable to the robot, introduces a higher level of uncertainty that poses a serious problem for such egocentric solutions.

Some methods for solving the global navigation problem egocentrically rely on building a probabilistic model of the environment based on sensor information and past experience, and performing computationally expensive algorithms using the model to infer the least cost path to a desired goal [5]. Other less computationally expensive methods build topological maps based on landmarks [8, 10], but these approaches generally suffer from the problem of aliasing, in which partial observability and sensor noise prevent deterministic discrimination between landmarks. While these approaches have met with moderate success in certain types of environments, they are all generally limited to specific classes of environments about which certain restrictive assumptions can be made a priori.

More recently, the path planning problem has been approached from a distributed intelligence perspective [1, 9, 11]. In these approaches, sensor networks of stationary nodes embedded in an environment provide global sensing, computational, and communication resources for mobile robots navigating stochastic, dynamic terrains. In these approaches, a mobile robot interacts with a sensor network by requesting suggested routes to a goal, which the sensor network then computes by invoking distributed algorithms that arrive at globally optimal solutions using only local communication. Such approaches have been termed *physical path planning* methods [11]. The primary benefit of these distributed approaches is the removal of the computational burden of path planning from the robot and the distribution of this burden over a large number of unsophisticated sensor nodes. Each node performs simple computations asynchronously and in parallel, sharing information about its environment with its neighbors through local communication. This disbursement of data collection and computation over a distributed sensor network allows for much faster convergence of global solu-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS '07 Honolulu, Hawaii, USA
Copyright 2007 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

tions than would be possible given a single robot with only local sensing capabilities.

We introduce an extension of previous approaches to physical path planning and show that our approach performs well in broader classes of environments and deployment configurations than can be handled by existing methods. The rest of this paper is organized as follows. Section 2 outlines previous approaches to physical path planning in more detail and presents some theoretical background in the fields of reinforcement learning and distributed network routing protocols. Section 3 discusses our approach to physical path planning. Section 4 presents some preliminary results of a SWIRL performing physical path planning in various simulated environments. Discussion of our results and future directions of research are presented in Section 5.

2. BACKGROUND AND PREVIOUS WORK

2.1 Distributed Path Planning

Previous distributed approaches to path planning [1, 9, 11] have been tested only in highly structured, uniformly distributed deployments of sensor networks (i.e., grid-like distributions) over topologically uniform environments (e.g., office buildings). A more interesting application for physical path planning is unstructured network deployment over rough, topologically diverse terrain (e.g., a sensor network dropped from a plane over a dense forest). Such deployment-environment combinations create several problems for previous approaches, primarily because all previous approaches have used hop count as a cost metric for finding shortest paths between nodes. While this assumption is valid for uniform, structured network deployments and environments with even terrain and few obstacles, this cost metric does not generalize well to the more difficult cases mentioned above.

The validity of hop count as a cost metric for distributed path planning algorithms requires the restrictive assumption that one can place reasonable bounds on the differences in time it takes a robot to travel between any two nodes. If the terrain is regular and all nodes are nearly equidistant from each other, a robot's travel time between any two nodes (one hop) will be relatively constant. However, in cases where the terrain is irregular, nodes are not evenly distributed, or there are obstacles not detectable by the nodes, the variance of one-hop travel time may be very high, and a network using hop-count as a cost metric will likely converge on suboptimal solutions.

Essentially, the hop-count cost metric ignores information about the difficulty a mobile robot may have in actually traversing the terrain between any two nodes and about the actual distance between those nodes. One approach [9] attempts to correct for this by having the network compute a "danger" potential field over the environment, which the network takes into account when computing the least cost path. However, the cost metric in this case is simply this potential field summed with the traditional hop-count metric. The approach also assumes that the individual nodes can sense relevant stimuli that indicate potential danger to a robot, which is not often a valid assumption.

Another drawback of prior physical path planning methods is that once the network converges on a solution given a specific goal, it holds all routes in the solution fixed until the goal is changed or obstacles that can be sensed by the nodes are added or removed. Because no information aside from the number of hops is taken into consideration when computing least-cost paths, real data from robots traversing the network cannot be used to update true path cost estimates (e.g. actual time to traverse a path, energy consumed by a robot traversing the path, etc.). Thus, using a hop-count cost metric precludes any potential learning by the network based on actual experience obtained from the robots it guides.

We address these issues in our approach and develop a reinforcement learning [13] approach to physical path planning. More specifically, we introduce a more realistic and relevant cost metric that is used to compute least cost paths over a wide range of environments given various (possibly unstructured) deployments of sensor networks. Using basic concepts from swarm intelligence [7], which deals with collections of relatively unsophisticated agents that communicate locally to achieve sophisticated, globally optimal behavior, we model our approach as a Swarm of Interacting Reinforcement Learners (SWIRL).

2.2 Distance Vector Routing

The algorithms used in distributed approaches to path planning are very similar to common network routing protocols, where the sensor nodes are analogous to packet routers and the individual robots can be likened to packets routed through the network. The objective of the network is to route each packet from a given source to a given destination with minimal latency. One of the more common techniques for packet routing is distance vector (DV) routing [12], in which each node maintains an estimate of the minimum cost to each other node in the network (a distance vector). Each node broadcasts its DV to neighboring nodes and updates its current estimates based on the DV estimates it receives from its neighbors. Updates are made to a node's DV by computing the direct cost to each of its neighbors and, for each destination node, taking the minimum over all of its neighbors of the sum of this cost and the minimum cost to the destination advertised by that neighbor.

More formally, if $d(x, y)$ represents the cost of a direct link between node x and its neighbor y, the minimum cost $D(x, z)$ of routing a packet from node x to a destination node z is computed as

$$D(x,z) = \min_{y \in N(x)} d(x,y) + D(y,z), \tag{1}$$

where $N(x)$ is the set of neighbors of x.

Every node in the network maintains an estimate for reaching each destination node $z \in V$ (where V is the set of all nodes in the network) and routes packets bound for a specific destination along the link that minimizes this value according to equation 1 (i.e., the arg min of Equation 1). Distance vector routing is a distributed form of the Bellman-Ford algorithm for computing shortest paths in a graph [4]. Although this method is guaranteed to converge to a global optimum even with asynchronous updates, convergence can be slow in many cases, and various enhancements are often used in network routing protocols to alleviate this drawback.

In the case of physical path planning, however, communication time is much faster than the time it takes to route robots through the network, since robots have much stricter motion constraints than radio or infrared waves. By adjusting the frequency of message passing employed by each node, one can allow for more than enough distance vector estimates to pass through the network to ensure convergence

rapid enough to be useful to a robot navigating through the network. Therefore, as network topology changes, or new obstacles are introduced, convergence time for computing a new solution is much less of an issue than it is in packet routing networks where packet traversal time is the same as traversal time of the distance vector estimates.

2.3 Reinforcement Learning

Reinforcement Learning (RL) [13] is a well established theoretical learning framework that lies between supervised learning and unsupervised learning. A reinforcement learning agent takes actions that affect the state of its environment and receives state information and a scalar reward signal from that environment. The agent's goal is to maximize its expected return, which is defined as the expected sum of future rewards received from the environment. It generally does this by learning a policy, π, that maps states to actions. RL problems are often formulated as Markov Decision Processes (MDPs), since there are formal convergence proofs for various RL methods if one assumes that the state representation of the problem is Markov (i.e., that the representation of the current state is a sufficient statistic for determining the distribution of possible next states given an action).

Formally, an MDP is defined as a four-tuple $< S, A, T, R >$, where S is a set of states, A is a set of actions, T is a transition probability function that maps transitions between states (given an action) to probabilities ($T(s, a, s') \mapsto [0, 1]$), and R is a reward function that maps state-action pairs to real numbers ($R(s, a) \mapsto \Re$). One of the more common RL methods, Q-Learning [14], estimates a value function $Q^\pi(s, a)$, $s \in S, a \in A$ for a policy, π, that maps state-action pairs to scalar values representing the expected return for taking action a in state s and from then on acting according to π. The value function estimated by the Q-Learning algorithm has been proven to converge to the optimal value function $Q^*(s, a)$ given sufficient exploration [15]. An agent who acts greedily according to $Q^*(s, a)$ when selecting actions (by choosing the arg max over all Q-values in a state s) acts optimally for a given MDP.

The network routing problem was formulated as a distributed Q-Learning problem by Boyan and Littman (1993) [2]. In this formulation, states are represented as source-destination pairs, and the set of actions admissible in a state is the set of neighbors of the source node — i.e., the source node can choose to route a packet to any one of its neighbors. The transition function arises from the topology of the network and the reward function is determined by packet latency between nodes (thus each agent is actually trying to minimize "reward" in this case). The value function in this formulation is actually distributed across the network, and thus the network can only converge on an optimal policy by sharing information about each node's portion of the value function through local communication.

We formulate the physical path planning problem as an MDP in a similar manner. States are source-destination pairs of sensor nodes in the network, and the set of admissible actions for each node n is the set of communicable neighbors of n. The transition function arises from the topology of the sensor network embedded in the environment and rewards are one-hop travel times measured and communicated to sensor nodes by the robots traversing the terrain.

3. PHYSICAL PATH PLANNING WITH SWIRLS

3.1 Model and Assumptions

In our approach to physical path planning, the sensor nodes of a SWIRL provide computational resources for computing the value function of the routing task as well as communication capabilities for transmitting value information to other nodes and guidance information to robots. The robots provide a means of actuation within the environment that allows for data collection about travel times that sensor nodes need to update their value function. In this way, the interaction of a single sensor node and one or more robots constitutes a single RL agent, and the interactions of many of these agents constitute a SWIRL.

The utilization of interactions between many (possibly diverse) unsophisticated agents using only local communication to perform complex global behavior is a hallmark of the field of distributed or swarm intelligence. Swarm intelligence [7] has been formalized into a powerful optimization framework that performs well on many tasks in which centralized control or information processing is either impossible or highly inefficient. Interesting applications of physical path planning, such as wireless sensor networks deployed over large geographical regions (e.g. a large forest or mountain range) would render central control extremely inefficient, whereas distributed control methods have the potential to perform quite well.

Our approach assumes that each sensor node and robot has a method of obtaining the location of and Euclidian distance to each of its neighbors (nodes and robots within its communication range). In practice this can be accomplished in many ways (e.g., GPS, radio signal strength, etc.), some more accurate than others.[1] Note that this does not trivialize the path planning problem because such an assumption provides no information to the network about the difficulty of traversing different types of terrain, with the possible exception of altitude, which can only provide a crude estimate of such difficulty. We further assume that each node communicates through radio transmission (e.g., standard 802.11 wireless) and thus does not necessarily have line of sight visibility to its neighbors, as was assumed in previous approaches.

Robots are assumed to posses a scheme for local navigation (i.e., simple obstacle avoidance) that allows them to move towards a given position in as straight a line as possible while avoiding obstacles along that route (e.g., see [3]). We also assume that all robots traversing the network are similar enough in their locomotion capabilities to have the same maximum velocity over equivalent terrains. In this way we can abstract out obstacle avoidance and rough terrain and model them simply as reductions in average speed. Robots are also assumed to be able to communicate with nearby nodes to submit queries for next hops given a destination node and to transmit travel time information that nodes use to update their value functions. Finally, we assume that robots are always able to travel between any pair of nodes. While we could assume that robots have a method

[1]We use absolute node location in our simulations to provide a heading for robots. If robots have local sensing capabilities sufficient to obtain both heading and distance to a given node then this assumption can be dropped.

```
d ← destination node
current ← null
next ← closest neighboring node
while(distance to d > Δ):
  restart timer
  while(distance to next > Δ):
    move towards next using local navigation
  end
  if(current != null):
    send UPDATE(next, timer) to current
  endif
  send QUERY(d) to next and wait for TARGET(n)
  current ← next
  next ← n
end
```

Figure 1: Pseudocode for algorithm run by robots.

of measuring travel progress and modify our algorithms to allow robots to send infinite travel times to nodes in cases where they decide that a certain link is impassable, we do not address this scenario in our current work.

3.2 Formal Specification and Algorithms

Each sensor node in the SWIRL will store a local portion of the value function in tabular form indexed by neighbor and destination. Formally, each node x will learn path cost estimates $Q_x(n,d)$, $n \in N(x), d \in V$, where $N(x)$ is the set of communicable neighbors of x and V is the set of all nodes in the network. Each of these estimates represents the expected travel time of a robot sent along a path from x to neighbor n bound for destination d.

A node x joins the network by initializing its Q-value $Q_x(x,x)$ to 0 and broadcasting its position x_{pos} to neighboring nodes in a HELLO message. Upon receipt of a HELLO message from a neighbor n, a node x checks to see if n is in its current set of neighbors $N(x)$. If $n \notin N(x)$, x adds n to $N(x)$, sends a HELLO message back to n containing its own position x_{pos}, and initializes the Q-value $Q_x(n,n)$ to an optimistic estimate of the time a robot will take to travel directly from x to n. In our simulations we simply made this value the quotient of the Euclidian distance between the two nodes and the maximum velocity of a robot over even terrain with no obstacles. This optimistic initialization of the value function has been shown to provide a good method of initial exploration [13]. Otherwise, if $n \in N(x)$, x ignores the HELLO message.

After broadcasting its HELLO message, a node x starts regularly broadcasting a portion of its current value function out to its neighbors in ESTIMATE messages. The distance vector $\vec{dv_x}$ that it broadcasts is computed by taking the minimum path cost estimate for each destination node $d \in V(x)$ over all neighbors $n \in N(x)$, where $V(x)$ is the set of nodes in the network of whose existence x is aware. That is, for each destination node $d \in V(x)$, $\vec{dv_x}[d]$ is the estimate $Q_x(n^*,d)$ that satisfies

$$Q_x(n^*,d) = \min_{n \in N(x)} Q_x(n,d). \quad (2)$$

Upon receipt of an ESTIMATE message from neighbor n, node x updates its value function table with the Q-values for

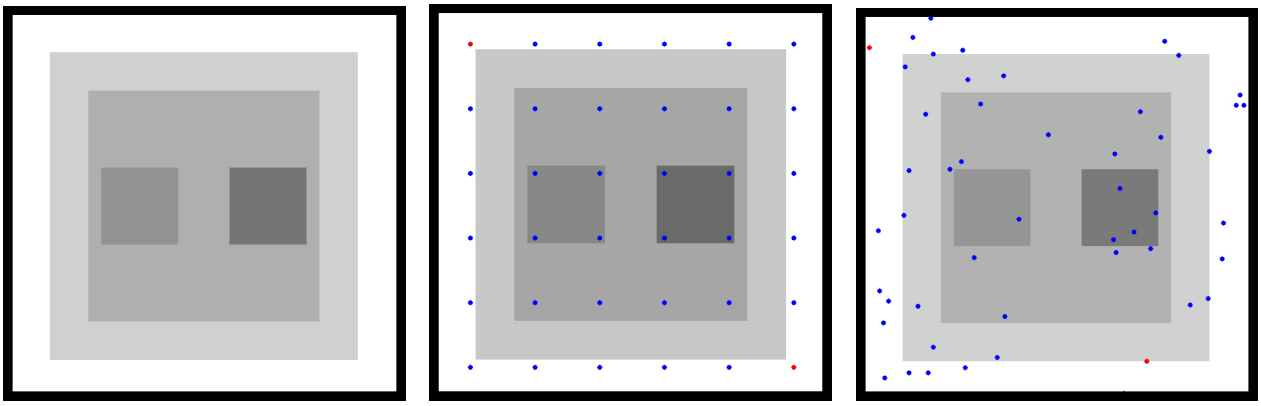
```
Initialize Q_x(x,x) ← 0
broadcast HELLO(x_pos)
loop forever:
  process any received messages m using receive(m)
  compute distance vector dv_x according to
    dv_x[d] ← min_{n∈N(x)} Q_x(n,d), ∀d ∈ V(x)
  broadcast ESTIMATE(dv_x)
end

receive(m):
  if(m is HELLO(n_pos) && n ∉ N(x)):
    N(x) ← n ∪ N(x)
    initialize Q_x(n,n) optimistically based on n_pos
    send HELLO(x_pos) to n
  endif
  if(m is ESTIMATE(dv_n)):
    Q_x(n,d) ← dv_n[d] + Q_x(n,n), ∀d ∈ dv_n
    if(d ∉ V(x)): V(x) ← d ∪ V(x), ∀d ∈ dv_n
  endif
  if(m is QUERY(d)):
    send TARGET(arg min_{n∈N(x)} Q_x(n,d))
      to querying robot
  endif
  if(m is UPDATE(n, timer)):
    Q_x(n,d) ← Q_x(n,d) + α[timer − Q_x(n,n)], ∀d ∈ V(x)
  endif
end
```

Figure 2: Pseudocode for algorithm run by a given sensor node x.

routing robots through n with these new estimates, adding its current estimate of $Q_x(n,n)$ to each one. If any of the estimates are for a destination node d which is not in $V(x)$, x adds d to $V(x)$ as well. This specification for message passing of path cost estimates is sufficient for computing least cost paths through the network assuming even terrain over which robots can travel at maximum speed. To address the issue of navigating rough terrain with potential obstacles, we next specify the types of messages robots and nodes can send to each other.

A robot begins a path through the network towards a specified goal node by approaching the nearest node x to its current position, which is assumed to be in communication range of the robot. Once within a small distance Δ of x, the robot transmits a QUERY message to that node containing its destination node d. The node x responds with a TARGET message containing the ID of the neighbor node $n \in N(x)$ for which $Q_x(n,d)$ is minimized. The robot then starts a timer t and travels towards node n using its local navigation algorithm. Once within Δ units of n, the robot stops the timer and transmits an UPDATE message back to node x containing the ID of node n and the value of the timer t. It then sends a QUERY message to node n and the process repeats until the robot comes within Δ units of node d, at which point it has reached its destination.

Sensor nodes receiving UPDATE messages update their value function as follows. For every destination node $d \in V(x)$, node x updates its Q-value $Q_x(n,d)$ according to the



(a) (b) (c)

Figure 3: (a) The rough terrain environment used in our experiments. Darker regions indicate areas with higher impedance. (b) The grid-structured network deployment used in our experiments. (c) A sample random network deployment.

following update rule

$$Q_x(n,d) \leftarrow Q_x(n,d) + \alpha[t - Q_x(n,n)], \quad (3)$$

where n is the node specified in the UPDATE message, t is the value of the timer included in the message, and $\alpha \in (0,1]$ is a constant step size parameter.

This update has the effect of moving the estimate of the one-hop path cost between node x and node n towards an exponential recency-weighted-average (parameterized by α) of past travel times observed by robots that actually traversed this link, and adding the change in that estimate to all other estimates that route robots through node n. Therefore, as robots are sent through the network along various paths they will collect statistics about actual travel times between pairs of nodes and report them back to the network, allowing the network to adjust its value function towards an accurate estimate of expected travel time between all source-destination pairs. The parameter α should be set less than 1 to prevent oscillation or divergence of the value function as a result of variance in travel time statistics. In all of our simulations, α was set to 0.7. Figures 1 and 2 present pseudocode for the algorithms run by the robots and sensor nodes, respectively, of our SWIRLs.

4. EXPERIMENTAL RESULTS

This section outlines the experimental setup used in our simulations with SWIRLs and some results with different types of network deployments over simulated rough terrain environments. In all of our experiments we generated an artificial terrain that was 300x300 arbitrary units of distance. Robots were arbitrarily set to move at a speed of 80 units per second over smooth terrain. Rough terrain was modeled as a noisy impedance factor which divided the maximum speed of a robot passing over that portion of the terrain. In our experiments, uneven terrain patches of the environment's surface were set to have a certain average impedance $i \in \{2,3,4,5\}$ defined by a mean i Gaussian distribution with variance σ^2. The proximity constant Δ was set to 1 unit of distance in all simulations.

A sample rough terrain environment is shown in Figure 3(a). White areas are those with no impedance, while gray patches indicate areas with non-zero impedance, with darker grays indicating higher impedance. In all of our experiments, the impedance distribution for the lightest gray region was $N(2,1)$, where $N(i,\sigma)$ is the Gaussian distribution with mean i and standard deviation σ. The next darkest gray region's impedance distribution was $N(3,1)$, and the left and right innermost gray regions had impedance distributions $N(4,2)$ and $N(5,2)$, respectively.

We experimented with two different types of sensor network deployments. For highly structured, uniform deployments we generated a grid network structure over the terrain by placing a node 20 units in each direction from the top left corner and spacing all other nodes out by a distance of 50 units in each direction until the terrain was covered. This grid structure resulted in a network of 36 nodes. An illustration of the grid deployment is shown in Figure 3(b). Additionally, we tested random grid deployments which were chosen by selecting a pre-specified number of (x,y) positions over the terrain uniformly at random and placing a node at each of these positions. Figure 3(c) shows a sample random network deployment.

The communication radius of sensor nodes was arbitrarily set to be 75 units. We observed empirically that 50 nodes in a random deployment were enough to provide sufficient coverage of the terrain (in terms of connectivity of the network) 95% of the time, given this communication range. This choice of communication range and number of nodes in a random deployment was chosen to facilitate data collection in our simulations. Network coverage is an important issue that must be dealt with more carefully in practice and we discuss this issue further in Section 5.

In our first experiment we compared the performance of a SWIRL containing ten robots to that of a network with ten robots that used a hop count cost metric on the terrain and grid deployment shown in Figure 3(b). Each robot was started at the upper left node of the network and assigned the lower right node of the network as a goal node. Robots were introduced into the network at two second intervals until all ten robots were in the network. When a robot reached the goal it was transported back to the start node and began moving towards the goal again. A run for

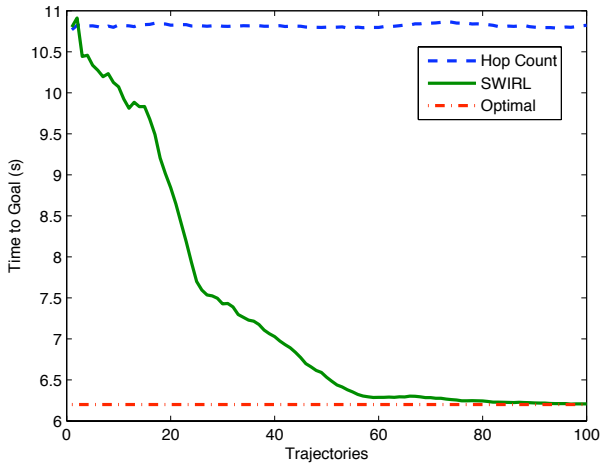


Figure 4: Average time to goal comparison between a hop-count based network and a SWIRL on a rough terrain, grid-structured network deployment task with 10 robots and a single start/goal node pair.

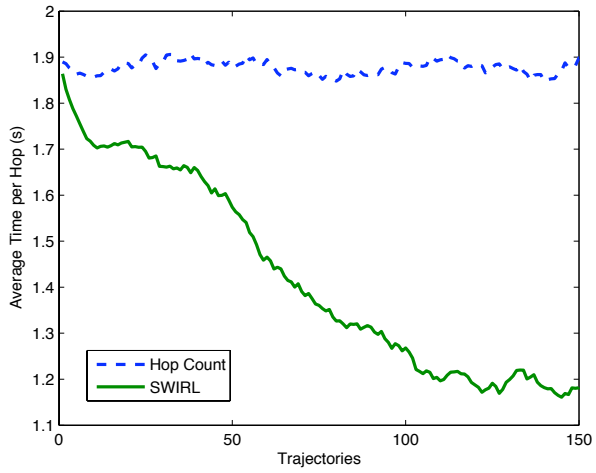


Figure 5: Average one-hop travel time comparison between a hop-count based network and a SWIRL on a rough terrain environment task with a grid-structured network deployment and 10 robots. Start/goal node pairs were chosen randomly for each trajectory.

each type of network consisted of 100 trajectories, each one ending when a robot reached the goal node. The dynamics of the simulation and the choice of impedance parameters in this experiment make it such that the optimal route from the selected start node to goal node is to go around either edge of the terrain, remaining in the white region for the entire route. This optimal path will, on average, result in a travel time of approximately 6.2 seconds. A direct line path from the start node to the goal node through the center of the terrain will, on average, result in a travel time of about 10.8 seconds.

The results of this experiment, shown in Figure 4, show the average time to goal for a robot starting at the start node as a function of the number of trajectories experienced by the network. Average time to goal was computed using a moving windowed average of trajectory length with a window size of 10, and was updated every time a robot reached the goal. The curves represent an average over 100 runs of the experiment. These result clearly show the limitations of the hop-count cost metric in rough terrain environments, even given a highly structured network deployment in which most node-to-node links are equal in length.

Since the hop-count-based network does not take into account the travel time information provided by the robots, no learning takes place and the performance of the network stays constant. The SWIRL uses the updates reported by the robots to refine its estimates of time to goal for given state-action pairs and switches its action (routing) choices to new routes not yet tried in practice until the minimal-time path is found. This exploration of untested paths is an implicit result of the optimistic initialization of the value function at each node. It should be noted that the SWIRL converges to the optimal path within about 60 trajectories using 10 robots. The effect of the number of robots in a SWIRL on convergence time will be addressed shortly.

Our next experiment looked at the difference in average one-hop travel time for each network type given random choices of start and goal nodes for robots. This experiment used the same setup as experiment 1 (rough terrain, grid-structured network deployment, 10 robots), but now robots were placed at random nodes to begin each trajectory towards goal nodes that were also randomly chosen. When a robot reached a goal, it updated a global moving average of one-hop time by dividing its total travel time for the trajectory by the number of hops in the trajectory, and was then restarted at a new random node with a new random goal. A run consisted of 150 trajectories and an average of 100 runs for each network type is shown in Figure 5.

Again we see the benefit of the real-time cost metric and its facilitation of learning from real data. The performance curves show that after about 100 trajectories the nodes learn to route robots along the outer edge of the terrain whenever possible, even if it means more hops to a destination. We see from this graph that the learning ability of the SWIRL allows it to achieve a significant decrease in average latency for every link traversed on the path to a goal as compared to the network using the hop-count cost metric.

We next experimented with random network deployments over rough terrain environments. Using the same terrain as in experiments 1 and 2, we deployed 100 different random networks of 50 nodes each, whose individual locations were chosen uniformly at random using a pseudo-random number generator with 100 different randomly generated seeds. This was done so that we could reproduce the same 100 deployments for both the hop-count based network and the SWIRL. As before, we placed each of 10 robots at a start node that was determined to be the node closest to the upper left corner of the terrain and assigned them a goal node which was the node closest to the lower right corner of the terrain. A sample random network deployment is shown in Figure 3(c). Figure 6 presents the results of this experiment, plotting average time to goal as a function of the number of

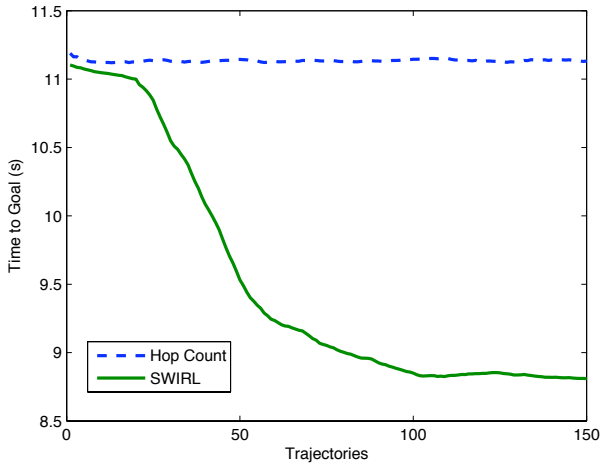


Figure 6: Average time to goal comparison between a hop-count based network and a SWIRL on a rough terrain, random network deployment task with 10 robots and a single start/goal node pair.

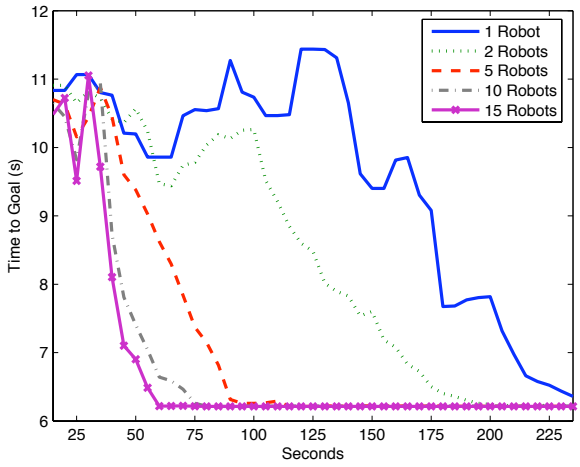


Figure 7: Average time to goal using a SWIRL with varying numbers of robots as a function of total experience time on a rough terrain, grid-structured network deployment task with a single start/goal node pair.

trajectories.

It should be noted that in all of our experiments we have shown convergence speeds as a function of the number of sampled trajectories. If one is interested in the speed of convergence as a function of real time, then the rate at which these trajectories can be sampled (which depends on the number of robots in the network) becomes relevant. Clearly, having more robots simultaneously traversing the network on different routes allows for simultaneous trajectory sampling and thus a decrease in convergence time. However, one would also expect a saturation effect (or diminishing returns) as the number of robots increases due to the increased probability of overlap in trajectories during the early learning phase. We next conducted an experiment to determine the effect the number of robots in the network has on convergence time.

Figure 7 shows the results of this experiment, which used the same setup as our first experiment. We see that with only one or two robots available in the SWIRL, convergence time is severely hampered since the network essentially has to wait for "access" to the robots to try out unexplored routing paths. However, with just a few more robots the rate of convergence seems to saturate very quickly for this particular task. As the size of the network increases, however, this saturation effect will likely be observed only for much larger numbers of robots.

5. CONCLUSIONS AND FUTURE WORK

We have presented a distributed approach to the path planning problem faced by mobile robots in dynamic, stochastic environments by combining ideas from swarm intelligence and reinforcement learning. Our SWIRLs were shown to perform much better than static, hop-count based sensor networks on simulated, rough terrain environments with non-deterministic dynamics using both highly structured and uniformly random network deployments. The interaction of stationary sensor nodes with mobile robots as we formulate it provides a sufficient framework for performing distributed reinforcement learning over the topological map defined by the sensor network.

It is worthy to note that although we only addressed one cost metric about which robots could provide data from real experience, there are certainly other metrics relevant to robot navigation that could be substituted or combined into the "reward" function for the SWIRL (e.g., robot energy consumption or sources of danger, such as high temperatures). Although defining an effective reward function can sometimes be the most difficult part of designing an RL system, the versatility of RL agents stems from their sole objective of maximizing (or minimizing) this scalar signal. If a decent reward function can be defined for a given cost metric, then a SWIRL will likely perform very well at the task posed to it.

We next present some possible directions of future research by addressing some of the assumptions made in our simulations. First, it may be unrealistic to assume that low power wireless sensor nodes can regularly broadcast their current value function at a specific frequency. More often than not such nodes are highly energy constrained and must conserve power as much as possible. There are a number of possible approaches that may alleviate this problem, including ongoing research involving performance-aware power management in systems that can harvest energy from the environment [6]. Another approach is to implement ad-hoc on-demand distance vector (AODV) routing [12] to propagate the value function through the network. Nodes implementing AODV routing for the most part only send routing messages when requested by other nodes (hence, on-demand). In the SWIRL specification this would apply not only when neighboring nodes make requests for value function estimates, but also when update messages from robots are received, since the value function has likely changed in some capacity. AODV routing would greatly de-

crease power consumption in a SWIRL, and future research with SWIRLs should focus on such an implementation.

Second, increasing the number of robots available to the network was shown to increase speed of convergence towards an optimal policy, but it may not always be possible to have large numbers of robots in any given SWIRL. Future work should also focus on decreasing convergence time by making efficient use of the data provided by robots in a SWIRL. One possible way this can be achieved is through the use of eligibility traces in value function updates [13]. As implemented in this paper, the updates performed after robot data is received only update one-hop time estimates. It is possible, however, to have the robot report back temporal data about larger sections of its trajectories to provide nodes with more unbiased samples of trajectory times. By introducing a parameter λ that controls the rate of decay of the traces, information about large sections of trajectories could be used in updates by weighting them according to their eligibility. This is a common technique in reinforcement learning methods to speed convergence, but whether it will help significantly in the physical path planning problem as posed in this paper is a topic worthy of future research.

Another avenue of future research relates to the algorithm currently run by the robots in the SWIRL; namely, to query the closest node for the optimal next hop and move all the way to that node before querying again. This policy imposes restrictions on the optimality of the paths that a robot can take to a goal, since it may have to deviate from the true optimal path to the goal to reach the node it has to query next. One can imagine, however, the robot querying multiple nodes surrounding its current position and estimating a value surface over these nodes using some interpolation method (e.g., linear), and then proceeding in the direction that maximizes (or minimizes, in the case of cost) the gradient of the surface, reestimating the surface at regular intervals as it travels. In this way, the robot need not complete its journey to a particular node to determine which direction to travel next, and thus can learn paths whose optimality is not constrained by the implicit discretization imposed by the sensor network.

Finally, one of our assumptions in using randomly deployed sensor networks was that there were sufficient numbers of nodes to provide connectivity of the network. This may not always be the case in practice and so another direction for future work lies in considering the case where the sensor nodes are themselves (perhaps limitedly) mobile. If this is the case, one could envision having the sensor net self-organize itself from an initial random configuration to one that is more uniform. Additionally, if the nodes are mobile, data from robots could also be used to learn optimal node positions as well as routing policies. For example, it may be the case in some rough terrains that a uniform distribution of sensor nodes over the terrain might be inefficient, and the network may exhibit better performance if nodes are clustered closer together in regions through which robots are routed frequently or where goals are often located. A SWIRL could potentially perform very well on such a parallel learning task. In short, mobility of the sensor nodes in a SWIRL opens up many potential directions for future research.

Acknowledgements

The author would like to thank George Konidaris and Andrew Barto for their helpful comments and insightful discussions.

6. REFERENCES

[1] M. Batalin, G. S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation (ICRA 04)*, pages 636–642, New Orleans, LA, April 2004.

[2] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 671–678, San Francisco, CA, 1993. Morgan Kaufman.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[4] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[5] D. Fox. *Markov localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, Institute of Computer Science III, University of Bonn, 1998.

[6] A. Kansal, D. Potter, and M. Srivastava. Performance aware tasking for environmentally powered sensor networks. In *ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2004.

[7] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[8] G. D. Konidaris and G. M. Hayes. An architecture for behavior-based reinforcement learning. *Adaptive Behavior*, 13(1):5–32, March 2005.

[9] Q. Li, M. DeRosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In *9th International Conference on Mobile Computing and Networking*, pages 313–325, September 2003.

[10] M. Mataric and R. Brooks. Learning a distributed map representation based on navigation behaviors. In R. A. Brooks, editor, *Cambrian Intelligence: The Early History of the New AI*. MIT Press, Cambridge, MA, 1990.

[11] K. J. O'Hara, V. Bigio, S. Whitt, D. Walker, and T. Balch. Evaluation of a large scale pervasive embedded network for robot path planning. In *IEEE International Conference on Robotics and Automation (ICRA 06)*, April 2006.

[12] E. M. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications*, April 1999.

[13] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[14] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[15] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.