

Chapter 10

Information-Directed Routing in Sensor Networks Using Real-Time Reinforcement Learning

Ying Zhang
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304
E-mail: yzhang@parc.com

Juan Liu
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304
E-mail: Juan.Liu@parc.com

Feng Zhao
Microsoft Research
One Microsoft Way, Redmond, WA 98052
E-mail: zhao@microsoft.com

1 Introduction

The primary task of a sensor network is sensing, that is, to collect information from a physical environment in order to answer a set of user queries or support other decision-making functions. Typical high-level information processing tasks for a sensor network include detection, tracking, or classification of physical phenomena of interest such as people, vehicles, fires, or seismic events. Routing in a sensor network is not just about getting data from one point to another in the network. It

must be optimized with respect to both data transport and information gathering. In other words, the routing structure must match the way the physical information is generated and aggregated by the network. For example, tracking a moving signal source may require the routing algorithm to combine information sequentially along a path, whereas querying the average temperature over an extended region may use a tree structure to aggregate the data from the region.

A broad class of sensor network problems can be characterized as collaborative signal and information processing problems. In such problems, a number of sensor nodes may possess useful information for a sensing task. The goal is to define and manage dynamic groups of such nodes, maximizing information extracted while keeping resource usage to a minimum. A number of approaches along this line have been reported in the literature (see, for example, [1, 2, 3]). As these approaches have demonstrated, a routing decision each local node makes during the information gathering process depends on the data generation model of the signal sources. This blurring of the abstraction barrier between applications and data transport is one of the characteristics of sensor networks. The key is for routing algorithms to handle and exploit constraints from data generation in a principled way. One would like to find a path that is not only efficient in terms of communication cost, but also aggregates as much information as possible along the path, so that the user can have a good estimate or description about the phenomenon of interest at the query or the destination node. In this sense, routing is more than a message-transporting mechanism, but also contributes to the successive message refinement.

Routing for ad hoc networks is a well-studied problem. Examples include OLSR (Optimized Link State Routing) [4], DSDV (Destination-Sequenced Distance Vector) [5], and AODV (Ad hoc On-demand Distance Vector routing) [6] protocols. Of recent interest is the topic of energy-aware routing; methods have been proposed, for example, in [7, 8], to plan paths minimizing the chance of node energy depletion. Geographic routing has also been developed. GPSR [9] routes data around a network hole, using a stateless protocol over a planar subgraph of the network topology. Geocasting [10] routes data to a geographically defined region. However, none of the above ad hoc routing algorithms considers information gathering and aggregation while routing data to a node or region, which is a major concern for sensor networks.

Routing protocols that explicitly explore data aggregation have also been developed. For example, directed diffusion [11] is a type of publish-

and-subscribe that sets up network paths between data source nodes and data sink nodes. It floods the network with data interest, and uses network parameters such as latency to autonomously reinforce good paths. However, directed diffusion does not integrate collaborative signal processing, so the paths generated can be inefficient for target tracking.

Quality of Service (QoS) routing strategies for mobile ad hoc networks have also been proposed [12, 13], where routing objectives and constraints can be specified. However, the types of objectives and constraints are limited to be *additive* (i.e., $\min \sum c_i$, e.g., delay) or *concave* (i.e., $\max \min b_i$, e.g., bandwidth),¹ to which, as we show later in this chapter, collaborative signal processing does not belong. Also in most cases, those strategies are not suitable to highly dynamic networks, because they first establish a route between the source and the sink and then follow up with a route maintenance phase if the route is broken. Message-initiated Constraint-Based Routing (MCBR) [14] developed a set of QoS-aware meta-routing strategies based on real-time reinforcement learning [15]. A couple of MCBR meta-strategies have been implemented in Berkeley nodes, a widely used platform for sensor networks, and have shown good performance both in simulation and in real hardware.

In this chapter, we apply real-time reinforcement learning to Information Directed Routing (IDR) in sensor networks, where learning is used to locate the target or the destination. Combining with information utilities, this method effectively generates paths which co-optimize for both communication metrics (e.g., small number of hops) and information metrics (e.g., tracking errors), even in the existence of network holes and unpredictable moving targets. Simulation results show that such a strategy is effective for both common query routing scenarios: routing a user query from an arbitrary node to the vicinity of signal sources and back, or to a prespecified destination, maximizing information accumulated along the path.

The rest of the chapter is organized as follows. Section 2 and Section 3 summarize the general problem of information-directed routing, with target tracking as a canonical problem for IDR [16, 17]. Information models, routing protocols, and performance metrics are also discussed, and the existing solutions are presented. Section 4 develops new near-optimal solutions to information-directed routing for the two common

¹Note that the definition of concave here is not related to the convexity in non-linear optimization.

information extraction scenarios, using real-time reinforcement learning techniques. Section 5 presents simulation results, demonstrating the benefit of using real-time reinforcement learning for information-directed routing. Section 6 concludes the chapter with discussions and future work along this direction.

2 Information-Directed Routing

As we have discussed in Section 1, routing in sensor networks is often coupled with sensing applications. Here we consider two source-initiated on-demand routing scenarios.

- **Routing query to a high-activity region:** This scenario is illustrated in Figure 1(a). The user issues a query from an arbitrary peripheral sensor node, which we call a query proxy node, requesting the sensor network to collect information about a phenomenon of interest. The query proxy has to figure out where such information can be collected and routes the query toward the high information content region. This differs from routing in communication networks where the destination is often known a priori to the sender. Here, the destination is unknown and is dynamically determined by the routing state and physical phenomenon.
- **Routing from a proxy to an exit, collecting information along path:** This scenario is pictured in Figure 1(b). The user, for example, an police officer, may issue a query to a query proxy, asking the sensor network to collect information and report to an exit node or a base station, for example, a police station, where the information can be extracted for further processing. In this scenario, the query proxy and the destination node may be far away from the high information content region. A path taking a detour toward the high information region shall be preferable to the shortest path from the query proxy to the destination.

We formulate the information-directed routing problem as an optimization problem. We use a graph $G = (V, E)$ to describe the sensor network structure. V is a collection of vertices corresponding to sensor nodes. E is a collection of edges corresponding to internode connectivity. Associated with an edge between two nodes v_i and v_j is a communication

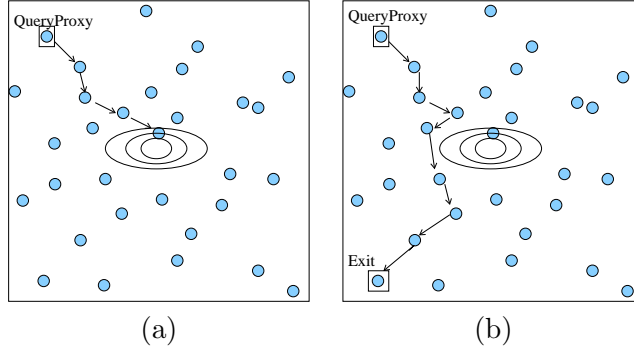


Figure 1: Routing scenarios: (a) Routing from a query proxy to the high activity region and back. (b) Routing from a query proxy to an exit or destination node. The co-centric ellipses represent isocontours of an information field, which is maximal at the center.

cost c_{v_i, v_j} . The information-directed routing problem can be formulated as finding a path $\{v_1, v_2, \dots, v_T\}$ minimizing the total cost

$$E = \sum_i c_{v_i, v_{i+1}} - \gamma I(v_1, v_2, \dots, v_T). \quad (1)$$

The first term measures the communication cost. The second is the negative information $-I(v_1, v_2, \dots, v_T)$, representing the total contribution from the sensors v_1, v_2, \dots, v_T . Under this formulation, routing is to find a path with maximum information gain at moderate communication cost. The regularization parameter γ controls the balance. In a network where shortest path is desired, γ is set to zero. For applications where information aggregation is of primary concern and communication cost is relatively low, γ should be set to high.

When the information gain is additive, i.e.,

$$I(v_1, v_2, \dots, v_T) = I(v_1) + I(v_2) + \dots + I(v_T), \quad (2)$$

the path-finding problem can be simplified considerably. It can be converted to the equivalent problem of finding the shortest path in a modified graph G' . G' has the same set of vertices and edges as G , but has a modified cost $c'_{v_i, v_j} = c_{v_i, v_j} - \gamma I(v_j)$ associated with each edge.

Strictly speaking, the additive condition (2) may not always hold in sensor network applications. The phenomenon of interest may have evolving *state*, and the information that a sensor contributes may be

state-dependent. For instance, in the target-tracking example discussed later in Section 3, the state is the probability distribution (belief) of target location. Suppose sensors v_1 and v_2 have information value I_{v_1} and I_{v_2} with respect to the current belief state. After applying sensor v_1 's measurement, the belief state has changed; hence I_{v_2} is obsolete and needs to be re-computed based on the new state. The information contribution I_{v_1} and I_{v_2} cannot be added together to account for the total contribution. The state-dependency property adds difficulty to routing, making it a combinatorial problem. Our strategy is to use heuristics to approximate the real cost (1). Though information is not strictly additive, the sum of individual information $\sum I(v_k)$ can often be considered as a reasonable approximation of $I(v_1, \dots, v_T)$ in cases where the belief state varies slowly.

3 IDR in Target-Tracking Applications

As an example of data generation processes in a sensor network, consider tracking a point signal source, or target, in a 2-D region. The goal of tracking is to estimate target location $x^{(t)}$ based on a set of measurements $\overline{z^{(t)}} = \{z^{(0)}, z^{(1)}, \dots, z^{(t)}\}$, indexed by time t , and collected by a set of nodes. To accomplish this, we use a statistical framework of sequential Bayesian filtering, a generalization of the well-known Kalman filtering [3]. For space reasons, we only briefly review the key concepts here. At time t , one has some rough prior knowledge about where the target is, usually in the form of a probability density function $p(x^{(t)}|\overline{z^{(t)}})$ (called belief). At time $t + 1$, a new measurement $z^{(t+1)}$ is collected. Sequential Bayesian filtering incorporates the measurement and updates the belief to $p(x^{(t+1)}|\overline{z^{(t+1)}})$ via Bayesian inference:

$$p(x^{(t+1)}|\overline{z^{(t+1)}}) \propto p(z^{(t+1)}|x^{(t+1)}) \cdot \int p(x^{(t+1)}|x^{(t)}) \cdot p(x^{(t)}|\overline{z^{(t)}}) dx^{(t)}. \quad (3)$$

The integral represents how prior belief is propagated to the current time step via the target dynamics $p(x^{(t+1)}|x^{(t)})$. The updated belief is the posterior of target location after observing all the measurements up to time $t + 1$. The method repeats as time advances. For more details about sequential Bayesian filtering in target tracking, please refer to previous work [3].

In sequential Bayesian filtering, sensor information is aggregated incrementally. Sensors along a routing path can contribute to target tracking via their measurements. To illustrate the aggregation of information,

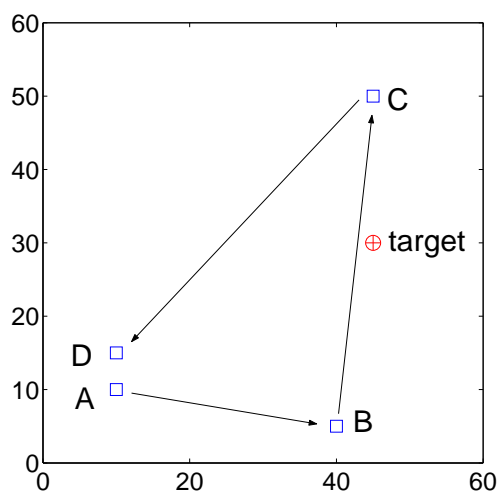


Figure 2: A sample sensor network layout: sensors are marked by squares, with labels A , B , C , and D . Arrows represent the order in which sensor data is to be combined. The target is marked by “ \oplus ”.

we consider a simple sensor network example consisting of four sensors, A , B , C , and D , as shown in Figure 2. Belief about the target location is shown in Figure 3 using grayscale grids. The brighter grid means that the target is more likely to be at the grid location. We assume a very weak initial belief, uniform over the entire sensor field, knowing only that the target is somewhere in the region. Figures 3(a)–(d) show how information about the target location is updated as sensor data is updated in the order of $A \rightarrow B \rightarrow C \rightarrow D$. At each step, the active sensor node, marked with a diamond, applies its measurement to update the belief. The localization accuracy is improved over time: the belief becomes more compact and its centroid moves closer to the true target location.

Routing, in this target-tracking setting, can be guided by the estimation result. The two IDR scenarios introduced in Section 2 become the following.

- **Target-Tracking Scenario:** Find where the target is, and route query to the vicinity of the target.
- **Target-Query Scenario:** Route query from a proxy to an exit, collecting information about the target location.

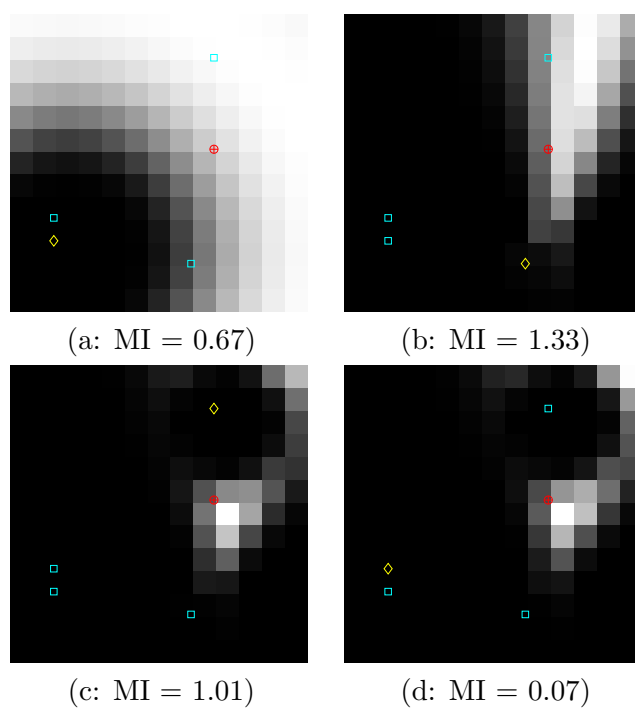


Figure 3: Progressive update of believe states of the target position, as sensor data is aggregated along the path $ABCD$. Figures (a)–(d) plot the resulting belief after each update. MI indicates information that is described in the next section obtained by Eq. (5).

3.1 Information Modeling

To quantify the contribution of individual sensors, we consider mutual information [3], a measure with a root in information theory and commonly used for characterizing the performance of data compression, classification, and estimation algorithms. As becomes clear shortly, this measure of information contribution can be estimated *without* having to first communicate the sensor data. The mutual information between two random variables U and V with a joint probability density function $p(u, v)$ is defined as $MI(U; V)$ which is

$$E_{p(u,v)} \left[\log \frac{p(u, v)}{p(u)p(v)} \right] = D(p(u|v) || p(u)), \quad (4)$$

where $D(\cdot || \cdot)$ is the Kullback–Leibler divergence [18] between two distributions. Under the sequential Bayesian filtering method in Eq. (3), the information contribution of sensor k with measurement $z_k^{(t+1)}$ is

$$I_{MI,k} = MI(X^{(t+1)}; Z_k^{(t+1)} | \overline{Z^{(t)}} = \overline{z^{(t)}}). \quad (5)$$

Intuitively, it indicates how much information $z_k^{(t+1)}$ conveys about the target location $x^{(t+1)}$ given the current belief. Other information metrics, such as the Mahalanobis distance [16], have also been proposed. They are computationally simpler to evaluate and are often good approximations to the mutual information under certain assumptions.

It is worth pointing out that information is an expected quantity rather than an observation. For example, in Eq. (5), the mutual information $I_{MI,k}$ is an expectation over all possible measurements $z_k^{(t+1)} \in \mathcal{R}$, and hence can be computed before $z_k^{(t+1)}$ is actually observed. In a sensor network, a sensor may have local knowledge about its neighborhood, such as the location and sensing modality of neighboring nodes. Based on such knowledge alone, the sensor can compute the information contribution from each of its neighbors. It is unnecessary for the neighboring nodes to take measurements and communicate back to the sensor. In our previous work [3], we provide a detailed algorithm describing how mutual information is evaluated based on knowledge local to the leader sensor. With little modification, this evaluation method can be extended to other information metrics.

In general, the information contribution of each sensor is state dependent. The information metric $I_{MI,k}$ of (5) depends on the belief state

$p(x^{(t)}|\overline{z^{(t)}})$. Revisiting the sensor network example in Figure 3, we compute the information contribution for each sensor. Note that sensors A and D are very similar and physically close by. Despite such similarity, the information values differ significantly (0.67 for A and 0.07 for D). Visually, as can be observed from Figure 3, sensor A brings significant changes to the initial uniform belief. In contrast, sensor D hardly causes any changes. The reason for the difference is that A applies to a uniform belief state, whereas D applies to a compact belief as shown in Figure 3(c).

State-dependency is an important property of sensor data aggregation, regardless of specific choices of information metric. Intuitively, how much new information a sensor can bring depends on what is already known. Note that in sensor networks, sensor measurements are often correlated. Hence a sensor's measurement is not "entirely new"; it could be just repeating what its neighbors have already reported. In the example above, sensor D is highly redundant with sensor A. Such redundancy shows up in the belief state, and thus should be discounted.

3.2 Protocols

We assume each node is aware of its own position, for example, using a GPS device or other location services. Each node also has knowledge about its local (one-hop) neighborhood, including node positions and sensor modalities, link quality, and one-hop communication cost. Such knowledge can be established through local message exchange between neighbors during network initialization and discovery. With these assumptions, the routing protocols described here can be regarded as a form of source-initiated on-demand routing.

We assume that a sensor field of size $X \times Y$ is discretized to $dX \times dY$ cells where d is the density. A belief state is a distribution $p : dX \times dY \rightarrow [0, 1]$ with $\sum p = 1$. We assume that every packet includes the current belief state about the target and the time when this belief was calculated. A packet sent from the source node combines the initial belief state (a uniform distribution) and its local sensor reading to obtain the current belief state. A forwarded packet uses the received belief state and the assumption of the maximum speed of the target to calculate its predicted belief state, then combines with its local sensor reading to obtain the current belief state. In other words, the Bayesian filter (3) is applied to each packet to update the belief state at each node.

For the target-tracking routing scenario, a forwarded packet is con-

sidered to be “arrived” at its destination if (1) the sensor reading is large and the increased utility is small, or (2) the maximum number of hops is reached. In this case, the final belief state at the “destination” will be sent back to the query (source) node.

For the target-query routing scenario, the destination is a known base station, addressed by its ID, or any sensor node, addressed by its attributes (e.g., location). The belief state at the destination will be used to estimate the target’s location.

3.3 Performance Metrics

There are various types of performance metrics for ad hoc routing in sensor networks [15], including, e.g., latency, throughput, loss/success rate, energy consumption, and lifetime of the network.

To measure the IDR tracking performance, we consider two quantities: (1) the mean-squared error (MSE) $E_{p(x^{(t)}|z^{(t)})} \|x^{(t)} - x_{true}^{(t)}\|^2$, and (2) the size of the belief state.

The MSE describes the tracking accuracy, and the belief size reflects uncertainty in the estimate. In this chapter, the belief size is calculated as follows. Let the number of cells with likelihood value exceeding a threshold be S , the belief size is obtained by \sqrt{S}/d where d is the density of cells.

We can combine these two metrics into one as a measurement ϵ for the quality of tracking; e.g.,

$$\epsilon = \mu + \sigma/k, \quad (6)$$

where μ is the mean error (square root of MSE), σ is the belief size, and k is a constant.

Furthermore, if we assume that the communication cost is identical for every hop (e.g., when all nodes use the same power level to transmit), we can compute the overall cost combining the quality of tracking and communication as

$$c = \lambda h + \epsilon, \quad (7)$$

where h is the number of hops from the source to the destination, λ is a coefficient, and ϵ is defined in Eq. (6).

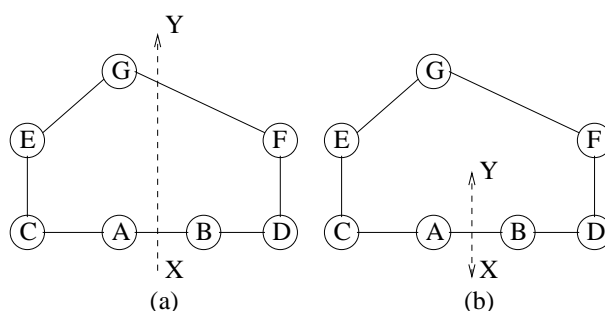


Figure 4: Routing in the presence of sensor holes. A through G are sensor nodes. All edges have unit communication cost. The dashed lines plot target trajectory. In (a), the target is moving from X to Y . In (b), the target is bouncing back and forth between X and Y .

3.4 Existing Solutions

Loosely speaking, there have been two types of approaches developed for IDR. The first type is greedy, routing packets to a neighbor with the highest information utility. An example is the Constrained Anisotropic Diffusion Routing (CADR) described in [1, 16]. This approach works well when the sensor network layout is uniform. For sensor networks with inhomogeneity such as holes, the relay may get trapped due to the greedy nature. Figure 4a provides a simple example. Here we use the inverse of Euclidean distance between a sensor and the target to measure the sensor's information contribution (assuming these information values are given by an "oracle"). The problem with greedy search is independent of the choice of information measure. Consider the case that the target moves from X to Y along a straight line (see Figure 4a). At time $t = 0$, node A is the leader, and can relay the information to its neighbor B or C . The relay goes to B because it has a higher information value. By the same criterion, B then relays back to A . The relay keeps bouncing between A and B , and the target moves away. The path never gets to nodes E , F , or G , who may become informative as the target moves closer to Y . The culprit in this case is the "sensor hole" the target went through. The greedy algorithm fails due to its lack of knowledge beyond the immediate neighborhood.

The second type uses a multiple step look-ahead to get around the problem of search getting trapped at sensor holes. The look-ahead hori-

zon should be large enough and comparable to the diameter of sensor holes, yet not too large to make the communication and computation cost prohibitive. For static sensor networks, the sensors can explore their local area in the network discovery phase and store in cache the information about inhomogeneity. This is done, for example, in [19]. Later in the path planning phase, such information will be helpful in selecting the value for the look-ahead horizon. Our earlier work [17] routes queries to the target vicinity using this look-ahead method. In the simulation study, it has been shown that the algorithm, with a suitable look-ahead horizon, is much more likely to succeed in routing a message around sensor holes (by a factor of 4–11) when compared to a greedy approach. At the same time it produces less error (by a factor of 2–4 in the simulation) in tracking a signal source.

The look-ahead approach has also been successfully used in the second IDR scenario, as studied in [17]. To route to a destination with a query of the target location, a real-time A* (RTA*) search [20] has been used. The baseline A* search is a best-first search, where the merit of a node is assessed as the sum of the actual cost g paid to reach it from the query proxy, and the estimated cost h to pay in order to get to the exit node (often known as the “cost-to-go”). It keeps a moving frontier of $g + h$, and iteratively expands the nodes on the frontier until the exit is reached. The resulting path is guaranteed to be optimal if the estimated h never exceeds the true cost-to-go; i.e., h is admissible. The well-known Dijkstra’s algorithm can be considered as a special case of A* search, with $h = 0$, always an underestimate. RTA*, as a real-time variation, guarantees to find a path if it exists, but as a price to pay for real-time operations, the solution may lose the optimality of the baseline A* search and may be suboptimal. The result of RTA* largely depends on the quality of the cost-to-go estimation h : one has to estimate the information contribution of sensors lying ahead, based on the currently available information alone without further querying or communication. The multiple step look-ahead is used for the estimation of h . It has been shown in simulation that the RTA* routing for IDR, compared to the shortest path from source to destination, produces less error in locating a signal source. Thus the RTA* routing is more effective in collecting information.

4 Real-Time Learning for IDR

There are two limitations for the existing solutions. Maintaining an M -hop neighborhood, where M is the look-ahead horizon, can be expensive in communication and nontrivial in computation. For the RTA* algorithm, the location of the destination has to be known to the query node which may not be the case for mobile sensors. To overcome these problems, we propose new IDR solutions based on real-time reinforcement learning.

4.1 Real-Time Reinforcement Learning for Routing

Real-time or agent-centered search has been an active research area in AI for the past decade [21]. Routing with an additive objective can be formulated as a weighted shortest-path problem, which can be solved by dynamic programming off line or real-time reinforcement learning on-line. Q-learning, a type of reinforcement learning [22], has been applied to telecommunication network routing, i.e., Q-routing [23] and sensor network routing [15]. It has been shown in both simulation and real hardware that this type of routing performs well in dynamic situations, e.g., forwarding packets to a mobile destination.

From a learning perspective, the problem of routing can be considered as designing a policy at each node, so that the overall path from source v_0 to destination v_d — v_0, v_1, \dots, v_d — is *optimal*, i.e., $\min \Sigma c_{v_i}$, where c_v is the *local* cost function at node v and Σc_{v_i} is the global *additive* objective. Routing for additive objectives is a *weighted* shortest-path problem.

All learning-based strategies typically consist of an *initialization* phase and a *forwarding* phase. The initialization phase normally either establishes a neighborhood by sending “hello packets” from each node, or generates a spanning tree by broadcast from the base station. The forwarding phase is a policy improvement phase, deciding which neighbor to pass the packet to according to its current estimates.

One can define a cost function on each node, called *Q-value*, indicating the minimum cost from this node to the destination. For a distributed sensor network, Q-value is initially unknown, and an initial estimation is made according to the type of message or by an initial flooding from the destination. Furthermore, a node also stores its neighbors’ Q-values, *NQ-values*, which are estimated initially according to the neighbors’ attributes and updated when packets are received from neighbors.

For wireless networks, value iteration in reinforcement learning can be realized in a cost-effective way [15]. For each packet sent out from a node, the current Q-value of the node with respect to that destination is attached. All the nodes are set to be in *promiscuous* listening mode. In other words, when a packet is sent, all neighbors, not just the designated receiver, will hear the packet and make use of the Q-value sent with the packet. Learning is triggered whenever a node overhears a packet, and its own Q-value is updated by

$$Q_m \leftarrow (1 - \alpha)Q_m + \alpha(c_m + \min_n NQ_m(n)) \quad (8)$$

where $0 < \alpha \leq 1$ is the learning rate, $c_m > 0$ is the local objective or cost, and n is a neighbor of this node.

In a forward search-based routing strategy, a node holding the current packet forwards it to the neighbor for which it has the best NQ-value. For dynamic and asymmetric networks, implicit confirmation is also used [15], which we do not discuss further in this chapter.

4.2 Routing a Query to Track a Target

In target-tracking applications, it is important to be able to initiate a query from an arbitrary entry node to find out the current status of a target, as illustrated in Figure 1(a). Ideally, the entry point node (query proxy node) would like to contact the nodes in the vicinity of the target, or the high information content region. Due to the distributed nature of ad hoc sensor networks, the query proxy may not be aware of existence and whereabouts of the high information content region. Using real-time reinforcement learning, packets discover the whereabouts of the target during routing.

Figure 5 shows the algorithm on each node. When a node receives a packet, it learns its Q-value, which is the number of hops to the sensors at the vicinity of the target in this case. If the packet is addressed to this node, it will either forward the packet, or, when the maximum number of hops is reached, send the packet back to the query node with the belief. The algorithm selects the next node to forward the packet according to the following rule: if the sensor is close to the target, the costs of the neighbors are the combination of the hop counts to the target and the information gains; otherwise, the costs are the hop counts to the target only. In either case, the neighbor with the minimum cost is selected as the next forward node.

Q: Q value
NQ: NQ values
R: sensor reading
Rm: threshold for at target
Ru: threshold for near target
Is: information gains
Cs: costs
 C_0 : maximum hops
 λ : coefficient
 $0 < \alpha < 1$: learning rate

```

received (m) at  $w$  from node  $u$  do
  if  $R > R_m$  then
     $Q \leftarrow 0$ ; //at target
  else // learn hops to target
     $NQ(u) \leftarrow m.Q$ ;
     $Q \leftarrow (1-\alpha) Q + \alpha (1+\min_v NQ(v))$ ;
  end
  if m is addressed to  $w$  then
     $m.hops \leftarrow m.hops + 1$ ;
    if  $m.hops < C_0$  then forward(m);
    else send m back to the query node;
  end
end

forward (m) at  $w$  do
  calculate Is for neighbors according to Eq. (5);
  if  $R > R_u$  then
     $Cs \leftarrow \lambda NQ - Is$ ; //close to target, use information gain also
  else
     $Cs \leftarrow NQ$ ; //too far from target, use hop counts only for learning
  end
   $v \leftarrow \operatorname{argmin}_n Cs(n)$ ;
   $m.Q \leftarrow Q$ ;
  send(m) to  $v$ ;
end
  
```

Figure 5: Learning-based IDR for tracking a target.

When the target is moving, the nodes that detect the target (when sensor reading $R > R_m$) will broadcast with their new Q-value, 0. The other sensors that are not at the vicinity of the target will update their Q-value in two cases: (1) received a broadcast packet from the destination node with Q-value 0; or (2) overheard a packet sent by a sensor node. For a network with enough packets, the Q-value field will be established soon, following the motion of the target. For a network without much traffic, extra control packets can be added for learning purposes, e.g., broadcast M -hops from the destination, or broadcast until the Q-value change is small. There is a tradeoff between the energy cost and the tracking performance in this case, which shall be investigated in the future.

With the learning algorithm, we revisit the examples in Figure 4. If the target is traveling in a straight line as in Figure 4a, starting from A , the path will bounce between A and B for a while. But as the target gets far from A and B , the hop counts (Q-value) of A and B will be increased so packets start to forward to C or D . When the target gets close to G , G will be the destination, and the path will extend to G via $ACEG$ or $BDFG$. On the other hand, if the target is traveling as in Figure 4b, then B is always the most informative sensor in A 's neighborhood, and vice versa, assuming the sensor readings at both A and B are above the threshold. The learning algorithm selects the path alternating between A and B .

4.3 Routing a Query About a Target to a Destination

In the scenario pictured in Figure 1(b), the goal is to route a query from the query proxy to the exit point and accumulate as much information as possible along the way, so that one can extract a good estimate about the target state at the exit node.

As in [17], for this task, we set the total communication cost (hop counts) close to some prespecified amount C_0 . Here the total cost C_0 is treated as a *soft* constraint, which is a “hypothetical” cost that the routing algorithm aims to achieve.² The value of C_0 controls the tradeoff between the communication cost and information aggregation.

Figure 6 shows the algorithm on each node. When a node receives a

²An alternative formulation of treating C_0 as a hard constraint that must be satisfied strictly. However, finding an optimal path under this hard constraint will require global knowledge about the sensor network and thus is inapplicable to ad hoc sensor networks.

Qd: Q value for destination
NQd: NQ values for destination
Qt: Q value for target
NQt: NQ values for target
 δ : estimation error
R, Rm, Is, Cs, C₀, λ , α : same as in Figure 5

```

received (m) at  $w$  from node  $u$  do
  if  $R > R_m$  then
     $Q_t \leftarrow 0$ ; m.foundTarget  $\leftarrow$  true; //at target
  else // learn hops to target
     $NQ_t(u) \leftarrow m.Q_t$ ;
     $Q_t \leftarrow (1-\alpha) Q_t + \alpha (1+\min_v NQ_t(v))$ ;
  end
  if  $w$  is destination then
     $Q_d \leftarrow 0$ ; //at destination
  else // learn hops to destination
     $NQ_d(u) \leftarrow m.Q_d$ ;
     $Q_d \leftarrow (1-\alpha) Q_d + \alpha (1+\min_v NQ_d(v))$ ;
  end
  if m is addressed to  $w$  then
    m.hops  $\leftarrow$  m.hops +1;
    if  $w$  is not destination then forward(m); end
  end
end
forward (m) at  $w$  do
   $L \leftarrow C_0 - m.hops$ ; //hops left
  if  $L > (Q_d + \delta)$  then //get more information
    calculate Is for neighbors according to Eq. (5);
    if m.foundTarget then  $C_s \leftarrow NQ_d$ ; else  $C_s \leftarrow NQ_t$ ; end
     $C_s \leftarrow \lambda C_s - I_s$ ;
     $v \leftarrow \operatorname{argmin}_n C_s(n)$ ;
  else //go to destination as soon as possible
     $v \leftarrow \operatorname{argmin}_n NQ_d(n)$ ;
  end
  m.Qt  $\leftarrow$  Qt; m.Qd  $\leftarrow$  Qd;
  send(m) to  $v$ ;
end
  
```

Figure 6: Learning-based IDR for query about a target.

packet, it learns both of its Q-values for the target and for the destination. A packet will be marked if it has visited the sensor nodes at the target. If the packet is addressed to this node and it is not the destination, the packet will be forwarded. The algorithm selects the next node to forward the packet according to the following rule: if the number of hops left exceeds an estimation error δ plus the estimated hop counts of this node to the destination, the neighbor with the smallest hop counts to the destination is selected. Otherwise, the cost will be evaluated as the combination of Q-value and the information gain. In the latter case, before the packet arrives at the target, the Q-value of the target will be used; after the packet arrives at the target, the Q-value of the destination will be used. In both cases, the neighbor with the minimum cost is selected as the next forward node. The estimation error δ represents the estimated maximum difference between the actual hops to the destination and its Q-value to the destination.

This algorithm behaves in a way that packets are first attracted to the target and then attracted to the destination. Parameters C_0 , δ and λ can be tuned to alter the path. Low C_0 value favors shorter paths, and high C_0 allows longer paths with more effective information aggregation. The increase of δ will increase the probability that the maximum hop counts do not exceed C_0 . Coefficient λ in addition controls the tradeoff between information gains and the communication cost.

For this routing scenario, if the destination is known (e.g. a base station), it is worthwhile to learn the Q-values to the destination by flooding the network during initialization. Even if the destination is moving, the Q-values will be readjusted as the result of learning, similar to what we have discussed in the previous scenario. On the other hand, if the destination is unknown, the first few packets will travel extra hops in order to learn the whereabouts of the destination.

5 Experimental Results

5.1 Simulation Environment

We have simulated the IDR protocols using Prowler [24], a probabilistic wireless network simulator. Prowler provides a radio fading model with packet collisions, static and dynamic asymmetric links, and a CSMA MAC layer.

Two types of sensors are modeled for target tracking: acoustic amplitude sensors and Direction-Of-Arrival (DOA) sensors. The acoustic am-

plitude sensors output sound amplitude measured at each microphone, and estimate the distance to a target based on the physics of sound attenuation. The DOA sensors are small microphone arrays. Using beamforming techniques, they determine the direction the sound comes from, i.e., the bearing of the target. The detailed description of these two types of sensors can be found in [3].

We simulate a sensor field of dimension 6×15 . Sensor layout is generated as follows: first generate a uniform grid of 15 rows and 6 columns evenly covering the region, then perturb the grid points with a uniform noise distribution in $[-0.1, 0.1]$. The resulting sensor layout is plotted in Figure 7a. To test the routing performance in the presence of sensor holes, sensors within the rectangular region centered at $(2.5, 4)$ with size $(4, 2)$ are removed. The resulting sensor network is shown in Figure 7b. The sensor network consists of roughly 90% amplitude sensors and 10% DOA sensors (i.e., the probability of the DOA sensor is 0.1), randomly spread over the sensor region.

The signal strength of the radio is set so that the maximum communication range is about 2 grids. In this simulator, as in real situations, neighborhood is established by communication rather than calculated by distances, and the results of neighborhood vary from run to run. Figure 7 shows snapshots of the communication links.

The routing algorithms are evaluated in terms of the IDR performance metrics presented in Section 2.

5.2 Routing to a Stationary Target

A stationary target is simulated at location $(2.5, 7.5)$. We use the sensor closest to the lower-left corner $(0, 0)$ as the query proxy node. Starting from the proxy node, we would like to estimate the target location and shoot the query toward it. In simulation, we allow a path length of 20 hops and examine the IDR performance at the end of the path. The sensor network is inhomogeneous with a sensor hole, as shown in Figure 7b.

For this routing task, we compare the learning algorithm with the greedy CADR algorithm. Each method is simulated with 100 independent runs, with each run 15 packets, 5 seconds a packet, from the source. The results are summarized and reported in Figure 8. Compared to the greedy CADR algorithm, the learning algorithm significantly improves the tracking performance after the first few packets.

Figure 9 visualizes the paths produced by the greedy CADR and the

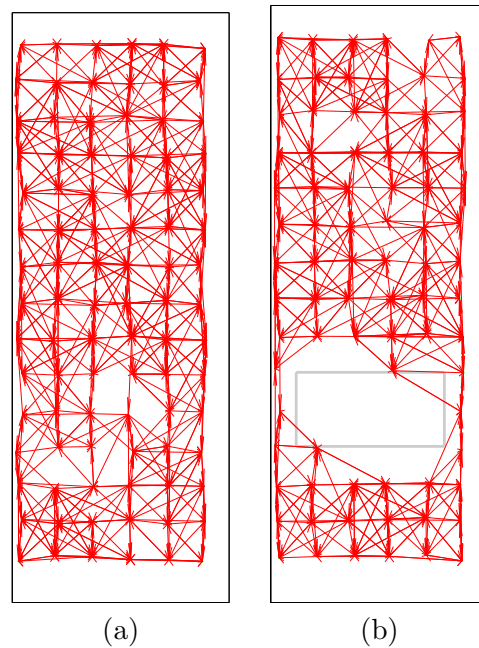


Figure 7: Examples of simulated sensor layout: (a) homogeneous and (b) with a sensor hole.

learning algorithm. The greedy algorithm path is plotted in Figure 9a. The path gets stuck and spends most of the hops bouncing between nodes on the lower side of the sensor hole. For comparison, the learning path of the tenth packet is plotted in Figure 9b. The path manages to get around the sensor hole, and ends at a node around the true target location. The tracking performance is also much improved; the target location estimate is more accurate.

5.3 Routing to a Moving Target

In tracking applications, the target is often nonstationary. In principle, a moving target can be considered as an extension of the stationary target case; the target can be considered as approximately stationary within a short time interval. Here we simulate a target moving along the straight line $x = 2.5$ (the center line of the sensor field along the vertical dimension) with speed $v = 0.1/s$. The query enters at the node closest to the initial target position $(2.5, 0)$.

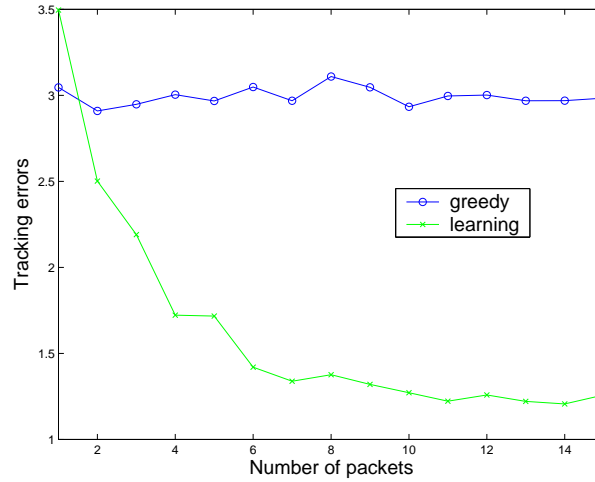


Figure 8: Tracking performance for a static target: greedy CADR algorithm vs. the learning algorithm. Tracking errors are calculated according to Eq. 6 with $k = 3$.

We compare the learning algorithm with the greedy CADR, with 100 independent runs for each. Same as the previous test, each run shoots 15 packets, 5 seconds a packet, from the source. The performance is summarized in Figure 10. Here we observe similar characteristics as in the stationary case. With the greedy algorithm, tracking errors are getting larger when the target moves into the hole and up. With the learning algorithm, the performance gets worse only for the first couple of packets and new paths are learned soon.

5.4 Routing to a Destination

Routing from a query proxy to an exit node or destination is tested with a stationary target at $(4, 7.5)$. The selection of query proxy and exit node can be arbitrary. We select the query proxy node as the node closest to the lower-left corner $(0, 0)$, and the exit node as the node closest to the upper-left corner $(0, 14)$. The tests are performed on a homogeneous sensor network (as in Figure 7a).

Recall from Section 2 that the information-directed routing problem is essentially a tradeoff between the communication expense and information aggregation. We simulated three different situations: (1) shortest

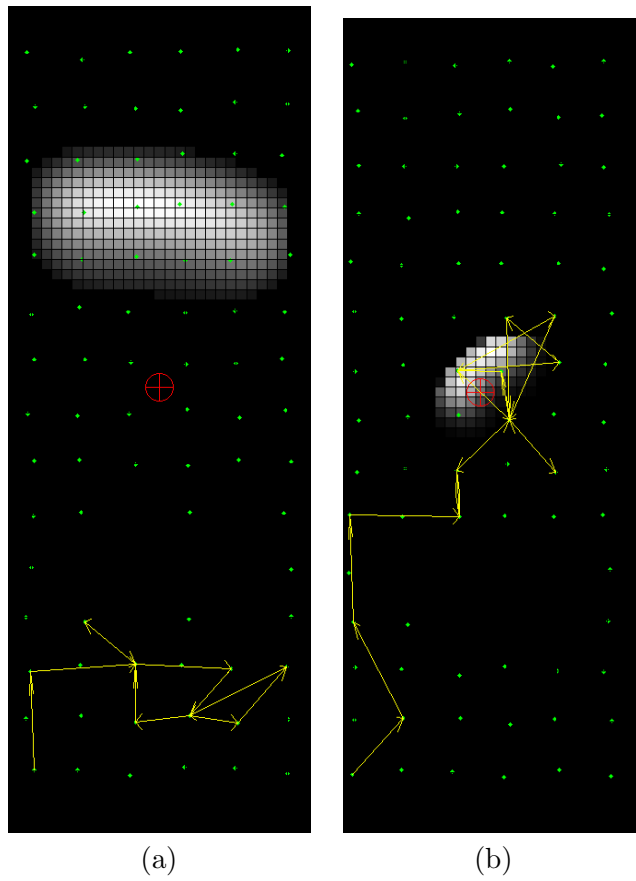


Figure 9: Query routing path produced by (a) the greedy CADR algorithm and (b) tenth packet of the learning algorithm. The target is located at $(2.5, 7.5)$, roughly in the middle of the sensor field. It is marked with " \oplus ". The selected paths are marked with lines.

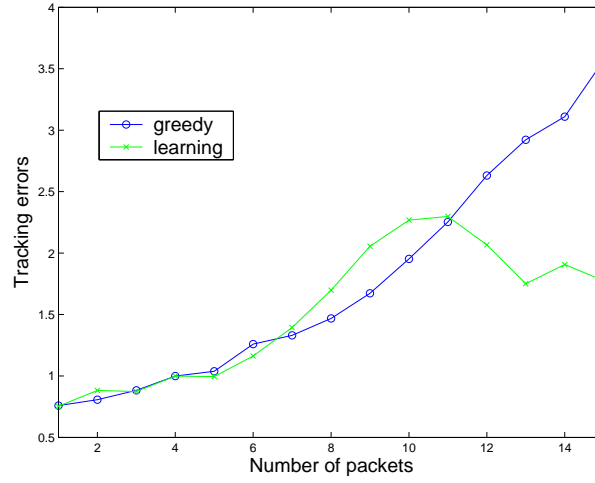
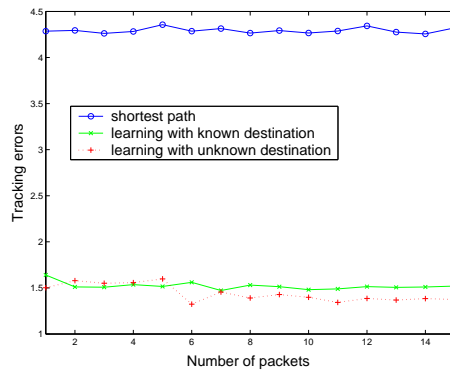


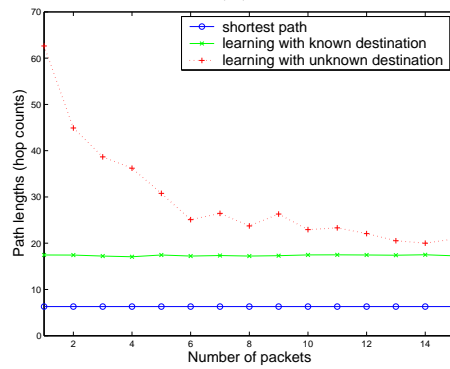
Figure 10: Tracking performance for a moving target: greedy CADR algorithm vs. the learning algorithm. Tracking errors are calculated according to Eq. 6 with $k = 3$.

path to the destination by setting C_0 small; (2) a detoured path (with $C_0 = 20$) to obtain the target information when the destination was known a priori; and (3) the same as (2) but the destination was unknown a priori. The second situation simulates the cases when the destination is a base station, and the third situation simulates the cases when the destination is an arbitrary node in the network. For the second situation, the Q-values to the destination are learned during the initialization by flooding from the destination. For the third situation, the Q-values to the destination are learned during routing.

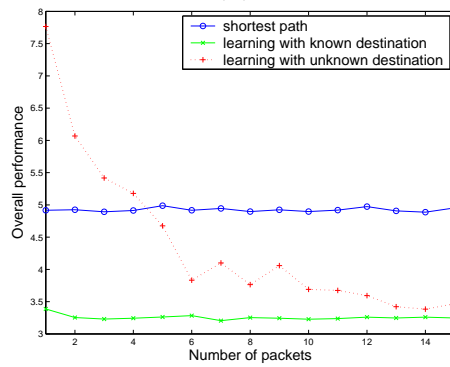
We tested these situations with 100 independent runs for each. Same as the previous test, each run shoots 15 packets, 5 seconds a packet, from the source. The results are compared in terms of tracking errors (Figure 11a), number of hops (Figure 11b) to the destination, and the overall performance (Figure 11c). We can see the learning algorithm works well for tracking whether or not the destination is known a priori. When the destination is unknown, the paths are long initially but gradually reduce to the allowed length C_0 as a result of learning. The learning algorithm with known destination gets the best overall performance, and the learning algorithm with unknown destination improves the performance over time due to shorter path lengths.



(a)



(b)



(c)

Figure 11: Shortest path vs. the learning algorithm with known and unknown destinations: (a) Tracking errors are calculated according to Eq. 6 with $k = 3$ (b) Path lengths are the hop counts from the source to the destination (c) Overall performance values are obtained according to Eq. 7 with $\lambda = 0.1$. Note that the same λ is used in the algorithm in Figure 6.

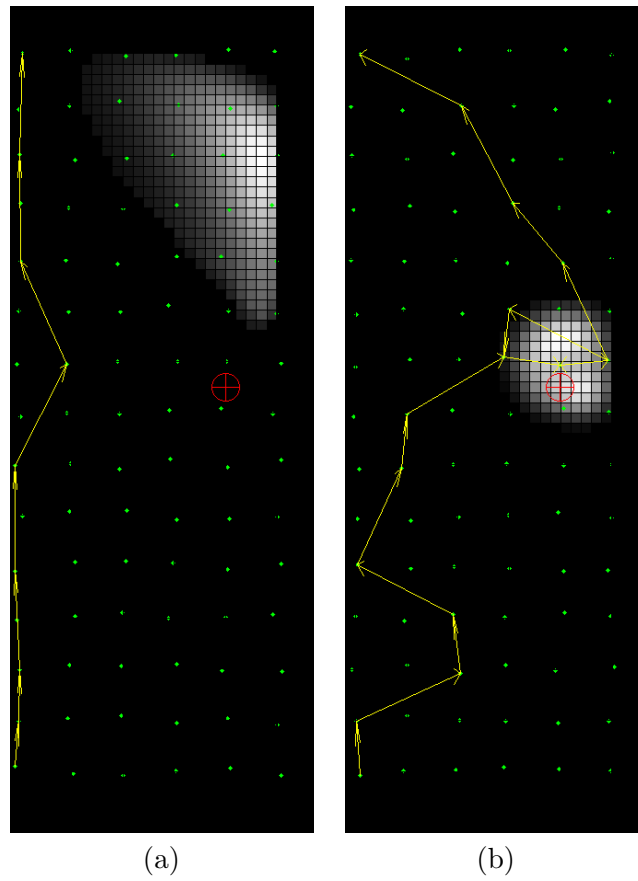


Figure 12: Query routing path produced by (a) the shortest path and (b) the learning algorithm. The target is located at $(4, 7.5)$, roughly in the middle-right side of the sensor field. It is marked with " \oplus ". The selected paths are marked with lines.

Figure 12 visualizes selected paths with different lengths. The shortest path is shown in Figure 12a. It has 6–7 hops and mostly follows a vertical line from the query proxy to the exit node. The belief state is fairly big, and cannot localize the target. Figure 12b shows a longer path of about 20 hops. Starting from the proxy, the path bends toward the target direction in an attempt to accumulate information. The tracking performance is vastly better than the shortest path.

6 Conclusions

We have demonstrated the benefits of using real-time reinforcement learning for information-directed routing that jointly optimizes for maximal information gain and minimal communication cost. In the simulation study, we have shown that: (1) for tracking targets, the learning algorithm, compared to the previous greedy algorithm, is able to route around the sensor hole after a couple of packets and to track the moving target effectively; and (2) for querying targets, the learning algorithm performs well in both situations with known or unknown destinations.

Unlike look-ahead routing algorithms [17], knowledge about the network structure or the application does not play an important role in learning-based algorithms for IDR. On the other hand, the more the system learns, (e.g., hop counts to the destination), the better the algorithm performs. For a static network, the performance should converge to its optimal. For a dynamic network, if the speed of change is commensurate with the speed of learning, the system should be adaptable to the dynamic situations.

Although we presented IDR usage in the context of localization and tracking problems, the general idea of using information to guide routing applies to other problems as well. For example, in monitoring and detection problems, information may be defined as the reduction of uncertainty in the hypothesis test of target presence. In classification problems, information may relate to how sensor measurement affects the overall classification error. The specific form of information model may vary; the basic structure of the routing algorithms stays the same.

We presented the algorithms for scenarios with a single stimulus. The algorithms can be generalized to handle multiple stimuli, using a spanning tree [25]. Generalizing the algorithms described here to handle dynamically moving stimuli while maintaining good approximations to the optimal routing tree remains a future research topic.

Acknowledgment

This work is funded in part by Defense Advanced Research Project Agency contract # F33615-01-C-1904.

References

- [1] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, pp. 61–72, Mar. 2002.
- [2] R. Brooks, C. Griffin, and D. Friedlander, "Self-organized distributed sensor network entity tracking," *International Journal of High-Performance Computing Applications*, vol. 16, no. 3, 2002.
- [3] J. Liu, J. E. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP, Journal on Applied Signal Processing*, vol. 2003, pp. 378–391, Mar. 2003.
- [4] T. Clausen, G. Hansen, L. Christensen, and G. Behrmann, "The optimized link state routing protocol, evaluation through experiments and simulation," in *IEEE Symposium on Wireless Personal Mobile Communication*, 2001.
- [5] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Computer Communications Review*, pp. 234–244, 1994.
- [6] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. 2nd IEEE Workshop on Mobile Computer System and Applications*, pp. 90–100, Feb. 1999.
- [7] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad hoc networks," in *Proc. MobiCom* (Rome, Italy), July 2001.
- [8] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. IEEE Wireless Communication and Networking Conference* (Orlando, FL), Mar. 2001.
- [9] B. Karp and H. T. Kung, "Greedy perimeter stateless routing for wireless networks," in *Proc. of MobiCom* (Boston, MA), Aug. 2000.

- [10] Y.-B. Ko and N. H. Vaidya, "Geocasting in mobile ad hoc networks: Location-based multicast algorithms," in *Proc. IEEE Workshop on Mobile Computer Systems and Applications* (New Orleans), Feb. 1999.
- [11] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. MobiCOM 2000* (Boston, MA), Aug. 2000.
- [12] S. Chen, "Distributed quality-of-service routing in ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, Aug. 1999.
- [13] K. Chen, S. H. Shah and K. Nahrstedt, "Cross-layer design for data accessibility in mobile ad hoc networks," *Wireless Personal Communication*, no. 21, pp. 49–76, 2002.
- [14] Y. Zhang and M. Fromherz, "Message-initiated constraint-based routing for wireless ad hoc sensor networks," in *Proc. IEEE Consumer Communication and Networking Conference*, Jan. 2004.
- [15] Y. Zhang, M. Fromherz and L. Kuhn, "Smart routing with learning-based QoS-aware meta-strategies," in *Proc. Quality of Service in the Emerging Networking, Lecture Notes in Computer Science*, vol. 3266, pp. 298–307, 2004.
- [16] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High-Performance Computing Applications*, vol. 16, no. 3, 2002.
- [17] J. Liu, F. Zhao and D. Petrovic, "Information-directed routing in ad hoc sensor networks," *IEEE Journal on Selected Areas in Communications*, Special issue on Self-Organizing Distributed Collaborative Sensor Networks, vol. 23, no. 4, pp. 851–861, 2005.
- [18] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, John Wiley and Sons, Inc., 1991.
- [19] Q. Huang, C. Lu, and G.-C. Roman, "Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints," in *Information Processing in Sensor Networks, Proc. of IPSN 2003*, Apr. 2003.

- [20] R. Korf, “Real-time heuristic search,” *Artificial Intelligence*, vol. 42, pp. 189–211, 1990.
- [21] S. Koenig, “Agent-centered search,” *AI Magazine*, vol. 22, no. 4, pp. 109–131, 2001.
- [22] R. S. Sutton and A. G. Barto, editors, “Reinforcement learning: an introduction,” Cambridge, MA, MIT Press, 1998.
- [23] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” edited by J. D. Crowan, G. Tesauero, and J. Alspector, in *Advances in Neural Information Processing Systems*, vol. 6, pp. 671–678, San Francisco, Morgan Kaufmann, 1994.
- [24] G. Simon, “Probabilistic wireless network simulator,” in <http://www.isis.vanderbilt.edu/projects/nest/prowler/>.
- [25] E. J. Cockayne and D. G. Schiller, “Computation of Steiner minimal trees,” in *Combinatorics (Conference on Combinatorial Mathematics)* (D. J. A. Welsh and D. R. Woodall, eds.), (Southend-on-Sea, Essex, England), pp. 53–71, Institute of Math. and Its Applications, 1972.