

# INFORMATION-DIRECTED ROUTING IN AD HOC SENSOR NETWORKS

*JUAN LIU*

PALO ALTO RESEARCH CENTER  
3333 COYOTE HILL ROAD  
PALO ALTO, CA 94304

*FENG ZHAO*

MICROSOFT RESEARCH  
ONE MICROSOFT WAY  
REDMOND, WA 98052

*DRAGAN PETROVIC*

UNIVERSITY OF CALIFORNIA, BERKELEY  
DEPT. ELEC. ENG. & COMP. SCI.  
BERKELEY, CA 94720

## ABSTRACT

In a sensor network, data routing is tightly coupled to the needs of a sensing task, and hence the application semantics. This paper introduces the novel idea of information-directed routing, in which routing is formulated as a joint optimization of data transport and information aggregation. The routing objective is to minimize communication cost while maximizing information gain, differing from routing considerations for more general ad hoc networks. The paper uses the concrete problem of locating and tracking possibly moving signal sources as an example of information generation processes, and considers two common information extraction patterns in a sensor network: routing a user query from an arbitrary entry node to the vicinity of signal sources and back, or to a prespecified exit node, maximizing information accumulated along the path. We derive information constraints from realistic signal models, and present several routing algorithms that find near-optimal solutions for the joint optimization problem. Simulation results have demonstrated that information-directed routing is a significant improvement over a previously reported greedy algorithm, as measured by sensing quality such as localization and tracking accuracy and communication quality such as success rate in routing around sensor holes.

**Keywords:** Routing, Ad hoc network, Information, Sensor network, Target localization.

## 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, 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.

Routing algorithms for a sensor network must be aware of the sensor network constraints and application requirements. A sen-

sor network is subject to a unique set of resource constraints such as limited on-board battery power and limited network communication bandwidth. In a typical sensor network, each sensor node operates untethered and has a microprocessor and limited amount of memory for signal processing and task scheduling. Each node also is equipped with one or more of acoustic microphone arrays, video or still cameras, IR, seismic, or magnetic sensing devices. Sensor nodes within each other's radio range communicate wirelessly. The tasks for a sensor network can be varied, depending on the nature of signal sources and how the information is used. For example, tracking a moving signal source may require the routing algorithm to combine information sequentially along a path, while 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 characteristic of resource-constrained sensor networks. The key is for routing algorithms to handle and exploit constraints from data generation and applications in a principled way.

Routing for ad hoc networks is a well-studied problem. Graph-based algorithms such as Dijkstra or Bellman-Ford type algorithms are commonly used to determine optimal paths. 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 in wireless or sensor networks. Methods have been proposed, for examples, in [7, 8], to plan paths minimizing the chance of node energy depletion. 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 consider information gathering and aggregation while routing data to a node or region, which is a major concern for sensor networks. Moreover, as will be discussed later in the paper, path dependency of the information aggregation problem for a sensor network renders shortest path algorithms, the basis for most of the ad hoc routing protocols, inapplicable.

---

THIS WORK WAS PRESENTED AT THE 2ND ACM MOBICOM SENSOR NETWORKS AND APPLICATIONS WORKSHOP (WSNA) IN SAN DIEGO, SEPTEMBER 19, 2003.

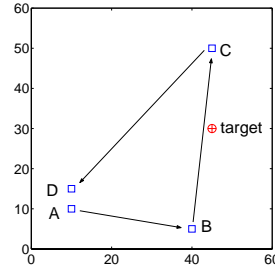
THE WORK WAS SUPPORTED IN PART BY THE DARPA SENSOR INFORMATION TECHNOLOGY (SENSIT) PROGRAM UNDER CONTRACT F30602-00-C-0139.

The work closest to what is reported here is directed diffusion routing protocol for sensor networks [11]. Directed diffusion 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. It is an elegant way to route data based on low-level data attributes, rather than node addresses, thus bringing routing closer to the application semantics. However, directed diffusion does not necessarily set up routing structures that are optimized for information gathering and aggregation, which is the primary focus of this paper. By considering the information content of the data packets, we argue that routing in a sensor network can be more than just a message-transporting mechanism. For example, it can contribute to successive message refinement as in a tracking application. CADR [12] considers both routing and data aggregation. However, the algorithm is greedy, and may suffer from getting trapped at local minima when network holes are present. This paper generalizes CADR.

The **contributions** of this paper are twofold: (1) The paper formulates a routing problem for a class of sensor network applications as a joint optimization of data transport and information aggregation, and presents a number of near-optimal algorithms for finding good quality routing paths. This builds on our earlier work on collaborative signal processing [1, 3, 12], and generalizes CADR to handle large sensor holes and more general query routing scenarios: routing a user query from an arbitrary entry node to the vicinity of signal sources and back, or to a prespecified exit node, maximizing information accumulated along the path. (2) As a concrete instantiation of the general information-directed routing, the paper derives a set of information models for realistic signal and sensing modalities, using a canonical problem for a sensor network — locating and tracking moving signal sources — as the information generation process. This is significant because performance evaluation of our routing algorithm depends crucially on meaningful data generation models. The information utility of individual sensors can be estimated without the need to communicate sensor data. Simulation results are presented to validate the routing algorithms using measures such as localization and tracking error, and to demonstrate the benefits of exploiting the trade-off between routing efficiency and information maximization. Despite the fact that in this paper we focus on target tracking for illustration, it should be noted that the concept of information-directed routing is more generally applicable. The notion of information utility is common to a broad class of sensing problems.

We assume each node is aware of its own position, for example, using a GPS device or other location services, and has knowledge about its local neighborhood, including node positions, 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 algorithms described here can be regarded as a form of source-initiated on-demand routing.

The rest of the paper is organized as follows. Sec. 2 introduces target tracking as a canonical problem for sensor networks and the associated data generation model. Sec. 3 derives the information models, and discusses the property of state dependency in information aggregation. Sec. 4 introduces the general formulation of information-directed routing, and approximations to information constraints. Secs. 5 and 6 develop near-optimal solution to information-directed routing for two common information extraction scenarios. Sec. 7 presents simulation results, demonstrating



**Fig. 1.** 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 with “ $\oplus$ ”.

the benefit of information-directed routing. Sec. 8 discusses possible extensions of the current algorithmic embodiment.

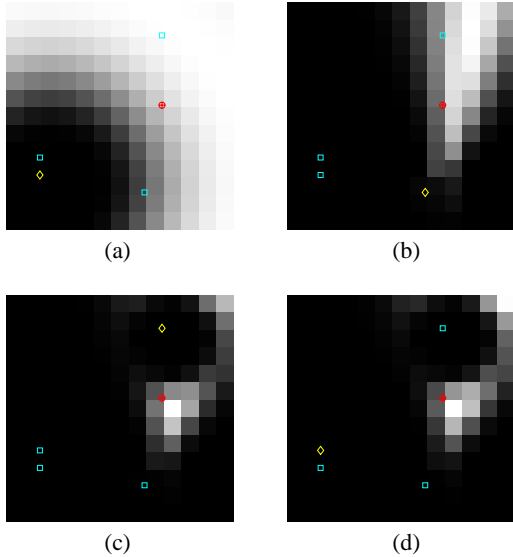
## 2. TRACKING AS A CANONICAL PROBLEM FOR SENSOR NETWORKS

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 conciseness, we only briefly review the key concepts here. At time  $t$ , one has some rough prior knowledge (called belief) about where the target is, usually in the form of a probability density function  $p(x^{(t)}|\overline{z^{(t)}})$ . 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:

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

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 our prior 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, we consider a simple sensor network example consisting of four sensors,  $A$ ,  $B$ ,  $C$ , and  $D$ , as shown in Fig. 1. Belief about the target location is shown using grayscale grids. 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. Fig. 2(a)-(d) shows how information about the target location is updated as sensor data is combined 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



**Fig. 2.** Progressive update of target position, as sensor data is aggregated along the path  $ABCD$ . Figures (a)-(d) plot the resulting belief after each update.

	Information	MSE	belief size
step1, sensor A	0.67	11.15	123
step2, sensor B	1.33	10.02	43
step3, sensor C	1.01	9.00	28
step4, sensor D	0.07	8.54	30

**Table 1.** Information aggregation in the sensor network pictured in Fig. 1. In the second column, information is measured using mutual information defined in (2).

the belief. The localization accuracy is improved over time: the belief becomes more compact and its centroid moves closer to the true target location.

To measure the 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 paper, the belief size is calculated as the number of cells with likelihood value exceeding 0.005. Table 1 lists the MSE and belief size values after each step. They generally decrease as the path is traversed, indicating that more information about the target position has been accumulated along the path.

### 3. MODELS OF INFORMATION

We introduce information models to formalize the intuition developed in Sec. 2. To quantify the contribution expected 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 will become 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

$$\begin{aligned} MI(U; V) &\triangleq E_{p(u,v)} \left[ \log \frac{p(u,v)}{p(u)p(v)} \right] \\ &= D(p(u|v) || p(u)), \end{aligned}$$

where  $D(\cdot || \cdot)$  is the Kullback-Leibler divergence [13] between two distributions. It indicates how much information  $V$  conveys about  $U$ . From a data compression perspective, it measures the savings in bits of encoding  $U$  if  $V$  is already known.

Under the sequential Bayesian filtering method (1), the information contribution of sensor  $k$  with measurement  $z_k^{(t+1)}$  is

$$I_{MI,k} = MI(X^{(t+1)}; Z_k^{(t+1)} | \overline{Z^{(t)}} = \overline{z^{(t)}}). \quad (2)$$

Intuitively, it indicates how much information  $z_k^{(t+1)}$  conveys about the target location  $x^{(t+1)}$  given the current belief. It can also be interpreted as Kullback-Leibler divergence between  $p(x^{(t+1)} | z_k^{(t+1)})$  and  $p(x^{(t+1)} | \overline{z^{(t)}})$ , the belief after and before applying the new measurement  $z_k^{(t+1)}$ , respectively [3]. Hence,  $I_{MI,k}$  reflects the expected amount of changes in the posterior belief brought upon by sensor  $k$ . Larger change means more information. Other information metrics, such as the Mahalanobis distance [12], have also been proposed. They are computationally simpler to evaluate and are often good approximations.

It is worth pointing out that information, which measures how much a sensor may contribute to the estimation, is an expected quantity rather than an observation. For example, in (2), the mutual information  $I_{MI,k}$  is an expectation over all possible measurements  $z_k^{(t+1)} \in \mathbb{R}$ , 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.

#### 3.1. State dependency

In general, the information contribution of each sensor is state-dependent. The information metric  $I_{MI,k}$  of (2) depends on the belief state  $p(x^{(t)} | \overline{z^{(t)}})$ . Revisiting the sensor network example in Fig. 2, we compute the information contribution for each sensor, and list the values in Table 1. 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 Fig. 2, 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, while D applies to a compact belief as shown in Fig. 2(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.

#### 4. INFORMATION-DIRECTED ROUTING

With the intuition developed in Sec. 2 and the information models presented in Sec. 3, we formulate the information-directed routing 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 inter-node connectivities. Associated with an edge between two nodes  $v_i$  and  $v_j$  is a communication 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

$$E = \sum_i c_{v_i, v_{i+1}} - \gamma I(v_1, v_2, \dots, v_T). \quad (3)$$

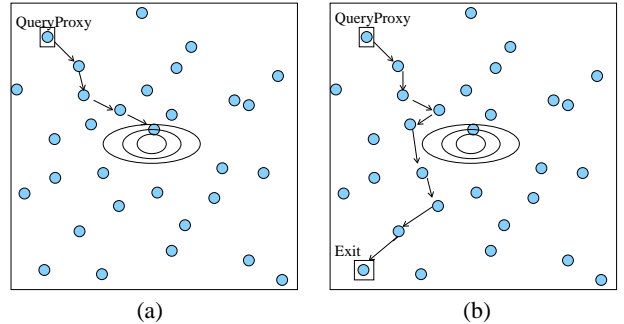
The first term measures the communication cost. The second is the negative information  $-I(v_1, v_2, \dots, v_T)$ , representing the total contribution from the sensors  $v_1, v_2, \dots, v_T$ . Under this formulation, routing is to find a path with maximum information gain at moderate communication cost. The regularization parameter  $\gamma$  controls the balance. In a network where shortest path is desired,  $\gamma$  is set to zero. For applications where information aggregation is of primary concern and communication cost is relatively low,  $\gamma$  should be set to a high value. The formulation (3) can also be interpreted as maximizing information gain under a communication cost constraint. In this case,  $\gamma$  is a Lagrange multiplier [14] whose value is determined by the constraint.

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

$$I(v_1, v_2, \dots, v_T) = I(v_1) + I(v_2) + \dots + I(v_T), \quad (4)$$

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. Dijkstra's or Bellman-Ford type of algorithms can be used to find optimal paths.

However, in sensor networks, the additivity condition (4) does not hold, due to the state-dependency property of information. For example, suppose sensors  $v_1$  and  $v_2$  have information value  $I_{v_1}$  and  $I_{v_2}$  with respect to a given 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 sets the information directed routing problem apart from traditional routing problems. Strictly speaking, the modified graph  $G'$  is not static. The edge cost depends on previously visited nodes and the signal source. Standard shortest-path algorithms are no longer applicable. Instead, a path-finding algorithm has to search through possible paths, leading to combinatorial explosion. To mitigate this problem, two strategies may be useful. We can restrict the search for optimal paths to be within a small region of the sensor network. Or we can apply heuristics to approximate the cost (3). 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



**Fig. 3.** Routing scenarios: (a) Routing from a query proxy to the high activity region and back. The co-centric ellipses represent iso-contours of an information field, which is maximal at the center. The goal of routing is to maximally aggregate information along a path while keeping the cost minimal. (b) Routing from a query proxy to an exit node, maximizing information gain along the path.

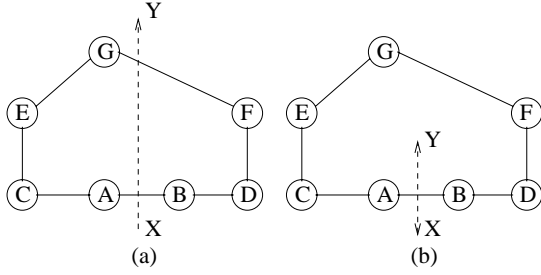
state varies slowly. In particular, one can show that, with mutual information as the information metric (2), the former is an upper bound of the latter.

In the rest of this paper, we consider two source-initiated on-demand routing scenarios. These two scenarios are common in ad hoc sensor networks. The first one is illustrated in Fig. 3(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. We will discuss this routing problem in more details in Sec. 5.

The second routing scenario is pictured in Fig. 3(b). The user, for example, a police officer, may issue a query to a query proxy, asking the sensor network to collect information and report to an extraction or exit node, for example, a police station, where the information can be extracted for further processing. In this scenario, the query proxy and exit nodes may be far away from the high information content region. A path taking a detour toward the high information region may be preferable than the shortest path. This routing problem is discussed in Sec. 6.

#### 5. ROUTING A QUERY TO WHERE INFORMATION IS

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 Fig. 3(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. Hence, it must find out which node in its local neighborhood may have better information, and then relay the query to that node. The relay process is similar to routing with gradient in the information field, only that the information field is



**Fig. 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  $Y$ .

dynamic and not directly observable. During the relay, the intermediate nodes incorporate their measurements to refine the target estimate.

<p><math>v_t</math>: current active sensor performing path planning  <math>S^{(t)}</math>: current state (e.g., prior belief) at time <math>t</math>  <math>P^{(t)}</math>: path planned up to time <math>t</math>  <math>L_0</math>: prespecified path length in the number of hops.  <math>\mathcal{N}</math>: list of neighbors within <math>M</math> hops to <math>v_t</math></p> <p><b>step 0.</b> Idle until receive routing request <math>(t, S^{(t)}, P^{(t)})</math>  <b>step 1.</b> Take new measurement <math>z^{(t+1)}</math>;  update to <math>S^{(t+1)}</math> as in (1)  <b>step 2.</b> For all <math>v_k \in \mathcal{N}</math>, evaluate information <math>I_k</math> as in (2);  Select node <math>v_{maxinfo} = \arg \max_{k \in \mathcal{N}} I_k</math>  <b>step 3.</b> Construct a graph of <math>v_k</math>'s <math>M</math>-hop neighborhood;  Compute the shortest path <math>P_{local}</math> from <math>v_k</math>  to <math>v_{maxinfo}</math> using Dijkstra's algorithm  <b>step 4.</b> <math>v_{next} =</math> node at the first hop on <math>P_{local}</math>;  <math>P^{(t+1)} = \{P^{(t)}, v_{next}\}</math>;  if <math>\text{length}(P^{(t+1)}) &lt; L_0</math> // continue relaying  relay to <math>v_{next}</math> the routing request  <math>(t+1, S^{(t+1)}, P^{(t+1)})</math>  else // done with length planning  send reinforcement message back to the query  proxy along the reversed path of <math>P^{(t+1)}</math>.  end  <b>step 5.</b> go back to step 0</p>
--

**Table 2.** Algorithm for min-hop routing at each node.

Previous approaches such as CADR [1] address the routing problem with a greedy relay strategy. Due to the greedy nature, the relay may get trapped near sensor holes. Fig. 4a provides a simple example. Here we use the inverse of Euclidean distance between a sensor and the target to measure 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 Fig. 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  $B$  since it has a higher information value. By the same criteria,  $B$  then relays back to

$A$ . The relay keeps bouncing between  $A$  and  $B$ , while 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.

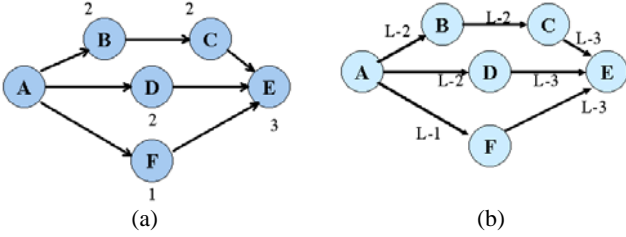
To route around holes in ad hoc networks, Karp and Kung [9] has proposed a greedy perimeter stateless routing (GPSR) method. Once the greedy routing gets stuck, it switches to the mode of following the perimeter of a hole. While this method guarantees successful routing in static planar graphs, it is not applicable to our scenario. One difficulty lies in detecting whether the greedy search is trapped, since the target state is not observable. Fig. 4b shows a counter example: observing the path alternating between  $A$  and  $B$  does not necessarily imply that the search is stuck. It may well be that the target itself is oscillating between points  $X$  and  $Y$ . In this case, the alternating path between  $A$  and  $B$  is desirable.

We resort to an information-directed multiple step look-ahead approach. We search for a path with maximum information aggregation among the family of paths with less than  $M$  hops. The look-ahead horizon  $M$  should be large enough and comparable to the diameter of sensor holes, yet not too large to make the computational cost prohibitive. For example, in simulations discussed in Sec. 7, with a  $4 \times 2$  sensor hole,  $M = 3$  or  $4$  works well. More generally, 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 [15]. Later in the path planning phase, such information will be helpful in selecting the value for  $M$ .

Here we describe a suboptimal path-finding algorithm called the *min-hop* algorithm, which can be implemented distributedly. Table 2 shows the algorithm on each node. A node wakes up upon receiving a query-routing request which consists of the tuple  $(t, S^{(t)}, P^{(t)})$  of time, state, and path, respectively. In target tracking, the state is the belief  $p(x^{(t)} | z^{(t)})$ . The active node incorporates a new measurement, plans the next move with a  $M$ -step lookahead horizon, and relays on, until the path meets a prespecified length  $L_0$ . To plan the path, the active node first selects the destination as the node with the highest information value within its  $M$ -hop neighborhood. It compares minimum hop paths from the active node to the destination, and selects the path with maximum information aggregation, measured as  $\sum_k I_k$ , where  $I_k$  is the information metric such as (2). The algorithm routes the query one hop down the selected path, and the path-finding procedure repeats.

To find a minimum hop path with maximum accumulated information, we perform the following conversion on the graph  $G$  and turn it into a shortest-path problem. The conversion is designed as follows: for each node  $i$ , we assign to each edge going into the node the cost of  $L - I_i$ , where  $L$  is some large number and  $I_i$  is the information value at node  $i$ . Fig. 5 shows an example of local neighborhood before and after the conversion. The information value of each node is marked next to the node in Fig. 5a. The cost of edges are marked in Fig. 5b. It is easy to show that the path with maximum information accumulation in the original graph is the shortest path in the converted graph. The path is then found efficiently using Dijkstra's algorithm, with the computational complexity  $O(|V| \log |V| + |E|)$ . The overhead here is the computation of information contribution. The complexity is  $O(|V|)$  at each planning step.

With the min-hop algorithm, we revisit the examples in Fig. 4.



**Fig. 5.** Conversion of  $M$ -hop local graph: (a) A local  $M$ -hop neighborhood of the current leader  $A$ , with information gain labeled at each node; (b) the converted graph that can be solved by a shortest-path algorithm.

Let the search depth  $M = 3$ . If the target is traveling in a straight line as in Fig. 4a, starting from  $A$ , the path will bounce between  $A$  and  $B$  for a while. But as the target gets close to  $G$ ,  $G$  will replace  $B$  as the most informative sensor in  $A$ 's neighborhood, and the path will extend to  $G$  via  $ACDG$ . On the other hand, if the target is traveling as in Fig. 4b, then  $B$  is always the most informative sensor in  $A$ 's neighborhood, and vice versa. The min-hop algorithm selects the path alternating between  $A$  and  $B$ .

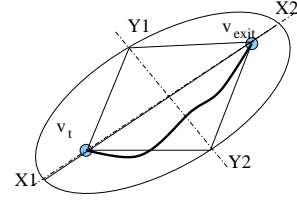
## 6. ROUTING A QUERY TO AN EXIT NODE

In the scenario pictured in Fig. 3(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, and yet keep the total communication cost 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.<sup>1</sup> The value of  $C_0$  controls the tradeoff between the communication cost and information aggregation. Low  $C_0$  value favors shortest path, and high  $C_0$  allows longer paths with more effective information aggregation.

For this task, we consider an  $A^*$  heuristic search which is commonly used for problems such as constraint satisfaction and motion planning [16]. The basic  $A^*$  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  $A^*$  with estimated cost-to-go  $h = 0$ , which is always an under-estimate. The complexity of  $A^*$  search depends on the choice of  $h$ . Conservative  $h$  takes longer to find an optimal solution. An optimistic estimate searches faster, yet an overly optimistic estimate (not admissible) may miss the optimal solution. For details analysis of  $A^*$  complexity, please refer to [16].

For real-time path-finding, we use an variation of the  $A^*$  method, namely the real-time  $A^*$  (RTA\*) search. It restricts search to small local region and makes real-time moves before the entire path is

<sup>1</sup>An alternative formulation of treating  $C_0$  as a hard constraint which 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.



**Fig. 6.** Node  $v_t$  is the current node;  $v_{exit}$  is the exit node. The ellipse covers all possible paths from  $v_t$  to  $v_{exit}$  satisfying communication cost constraint. The thick curve represent one sample path.

planned. Only local information is used in the RTA\* search, hence it can be implemented distributedly and is suitable for ad hoc networks. It 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 selected path may exhibit backtrack behavior.

RTA\* search is recursive. Given an estimated cost-to-go, the active node selects the best move (with some lookahead horizon). The algorithm repeats until the exit node is reached. The key to implementing a RTA\* algorithm is defining a suitable heuristic to estimate the cost-to-go  $h$ . The total cost (as in (3)) comprises a communication cost term and an information aggregation term. The estimation of communication cost is straightforward. One can use standard metric such as Euclidean distance.

The estimation of information contribution is more complicated. One has to estimate the information contribution of sensors lying ahead, based on the currently available information alone without further querying or communication. Here we describe an estimation method. Suppose we have planned the path  $P^{(t)}$ , and is now at  $v_t$ . To further reach the exist node  $v_{exit}$ , the remaining path length is upper bounded by  $C_0 - C_{P^{(t)}}$ , where  $C_{P^{(t)}}$  is the communication cost already paid. Thus, the locus of all feasible paths forms an ellipse with  $v_{exit}$  as one focus point and the current node  $v_t$  as the other. Fig. 6 shows such an ellipse. We denote its extrema along the major axis as  $X_1$  and  $X_2$ , and extrema along the minor axis as  $Y_1$  and  $Y_2$ . Within the ellipse there are infinitely many paths satisfying the length constraint. Some path may be complicated and hard to describe, such as the thick curly path in the figure. Rather than estimate  $h$  for all possible paths, we sample four paths as representatives:

- Path 1: the concatenation of two line segments:  $v_t \rightarrow X_1$  and  $X_1 \rightarrow v_{exit}$ .
- Path 2:  $v_t \rightarrow X_2 \rightarrow v_{exit}$
- Path 3:  $v_t \rightarrow Y_1 \rightarrow v_{exit}$
- Path 4:  $v_t \rightarrow Y_2 \rightarrow v_{exit}$

For each path, the information is computed via numerical integral, which samples the path at an interval inversely proportional to the sensor density. From  $v_t$  to  $v_{exit}$ , the information lying ahead is estimated as the maximum among the four paths, as an approximation to the admissible heuristic estimate. The computational complexity of estimating information is proportional to the number of sampling paths (4 in this case), times the number of integral intervals. On average we have about 10 samples along each path. Hence the overall complexity is about 40 times the computation of  $I$ .

<i>function to estimate information-to-go</i>	
<b>Input:</b>	$v_t$ : current active sensor doing path planning $P^{(t)}$ : path planned up to time $t$ $C_0$ : prespecified path length constraint $v_{exit}$ : exit node
<b>Output:</b>	$h_{info}$
<b>1.</b>	Compute constraint on remaining path: $C = C_0 - C_{P^{(t)}}$
<b>2.</b>	If $C \leq  v_t - v_{exit} $ $h_{info} = 0$ ; else compute ellipse $E_t :  l - v_t  +  l - v_{exit}  = C$ . compute extrema points $X_1, X_2, Y_1,$ and $Y_2$ define sample paths: Path1= $v_t \rightarrow X_1 \rightarrow v_{exit}$ . Path2= $v_t \rightarrow X_2 \rightarrow v_{exit}$ . Path3= $v_t \rightarrow Y_1 \rightarrow v_{exit}$ . Path4= $v_t \rightarrow Y_2 \rightarrow v_{exit}$ . compute $h_{path}$ for $path \in \{ \text{Path1, Path2, Path3, Path4} \}$ $h_{info} = \max h_{path}$ ; end
<b>3.</b>	Return $h_{info}$ .

**Table 3.** Function to estimate information-to-go

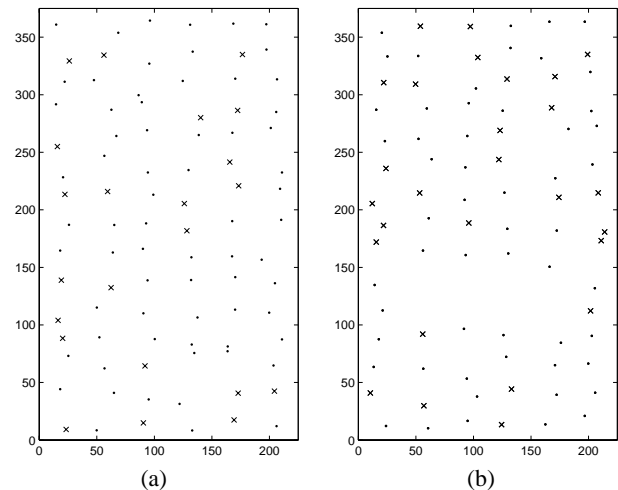
The estimation of information contribution is summarized in Table 3. If the remaining path length allowance  $C_t - C_{P^{(t)}}$  is smaller than the Euclidean distance between  $v_t$  to  $v_{exit}$ , the ellipse cannot be constructed. In this case, the estimation algorithm returns zero information. At this point, the forward search degenerates into a shortest-path problem based on the communication cost alone. Furthermore, if the initial allowance  $C_0$  is zero to start with, then the algorithm returns the shortest path from the querier to the exit.

## 7. EXPERIMENTAL RESULTS

Simulations were carried out to validate and characterize the performance of the proposed routing algorithms. We simulate a sensor field of dimension  $225 \times 375 \text{m}^2$ . Two types of sensors are used for target tracking: acoustic amplitude sensors and direction-of-arrival (DOA) sensors. The acoustic amplitude 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 beam-forming techniques, they determine the direction where sound comes from, i.e., the bearing of the target. The detailed description of these two types of sensors can be found in [3].

Sensor layout is generated as follows: first generate a uniform grid of 15 rows and 6 columns to evenly cover the region, then perturb the grid points with independent Gaussian noise of  $N(0, 25)$ . The resulted sensor layout is plotted in Fig. 7a. To test the routing performance in the presence of sensor holes, we remove the uniform grid points in rows 5-6 and columns 2-5 before adding perturbation. The resulting sensor network is shown in Fig. 7b. The sensor network consists of 70% amplitude sensors and 30% DOA sensors, randomly spread over the sensor region. Each sensor can directly communicate to neighbors within a 50m radius.

The routing algorithms are evaluated in terms of the efficiency of communication expenditure and the effectiveness of information aggregation in support of target tracking. We measure tracking performance using MSE and belief size. Besides these numer-



**Fig. 7.** Examples of simulated sensor layout: (a) homogeneous and (b) with a sensor hole. The points marked with a dot denote amplitude sensors, and the points marked with a “x” denote the DOA sensors.

ical measures, characteristics of selected path are also of interest. We pay attention to noticeable features such as whether the path successfully gets around holes, where it ends, and how it takes detour to accumulate information.

### 7.1. Query routing using a min-hop algorithm

Here we present the simulation results for routing a query from an arbitrary query proxy node to high information content region. Target may be stationary or moving.

#### 7.1.1. Stationary target

A stationary target is simulated at location (125, 200). 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 progressively estimate the target location and shoot the query toward it. In simulation, we allow a path length of 20 hops and examine the performance at the end of the path. The sensor network is inhomogeneous with a sensor hole, as shown in Fig. 7b.

For this routing task, we compare the min-hop algorithm using a look-ahead horizon  $M = 2, 3,$  and  $4$  with the greedy CADR algorithm. Each method is simulated with 100 independent runs. The results are summarized and reported in Table 7.1.1. As discussed in Sec. 5, some paths may get stuck and fail to route around the sensor hole. If the path has an ending point with y-coordinate below 100, we consider that as a “stuck” situation.

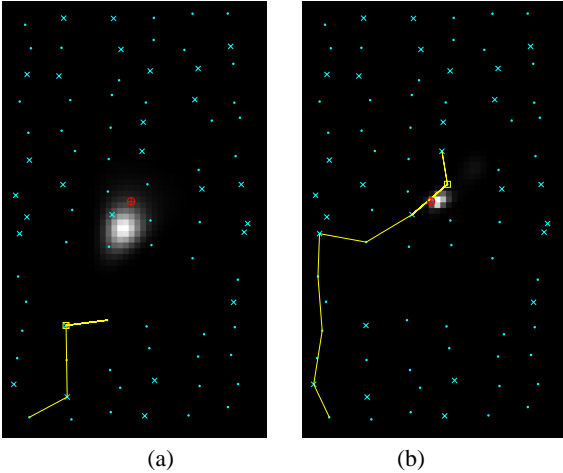
Compared to the greedy CADR algorithm, the min-hop algorithm significantly improves the tracking performance. For example, with a 3-step lookahead, the min-hop algorithm successfully routes around the sensor hole in 92 runs; only 8 runs are stuck at the sensor hole. By contrast, the CADR fails in 80 runs. Tracking performance is also improved: with  $M = 3$ , the min-hop algorithm reduces the square root of MSE by a factor of 4, and reduces the belief size by a factor of 3. The tracking performance comparison suggests that the min-hop algorithm aggregates information more effectively than the greedy CADR. Table 7.1.1 also lists the average distance between the path ending point and the

	sqrt(MSE)	belief size	# of stuck-runs	dist
CADR	29.88	197.90	80	108.91
$M = 2$	17.72	152.38	55	78.39
$M = 3$	7.41	72.27	8	32.94
$M = 4$	5.40	67.38	0	26.35

**Table 4.** Simulation results: routing a query to target vicinity using the greedy CADR and the min-hop algorithm with lookahead horizon  $M = 2, 3$ , and 4.

	# of lost runs	statistics of good runs	
		sqrt(MSE)	beliefsize
CADR	93	23.61	155.92
$M = 2$	57	14.91	118.72
$M = 3$	19	11.95	146.01
$M = 4$	5	11.82	103.05

**Table 5.** Tracking performance to route a query to high information region with a moving target. The numbers are averaged over 100 runs and all time steps.



**Fig. 8.** Query routing path produced by (a) the greedy CADR algorithm and (b) the min-hop algorithm with  $M = 3$ . The target is located at (125, 200), roughly in the middle of the sensor field. It is marked with a “ $\oplus$ ”. The selected paths are marked with solid lines. The nodes where the paths end are marked with a square.

true target. This distance indicates the capability of the routing algorithm to route a path to the vicinity of the target. The greedy CADR performs poorly here: the ending point is on average very far (108.91m) from the true target location. The min-hop algorithm routes the path to much closer positions. For example, with  $M = 3$ , the ending point is approximately 33m from the target. The overall computational complexity of path planning is roughly proportional to the size of the  $M$ -hop neighborhood. In our simulation, the average neighborhood size is 6.2 for  $M = 2$ , 20.2 for  $M = 3$ , and 39.2 for  $M = 4$ .

In general, the overall performance improves with the increase of the lookahead horizon  $M$ , but the improvement is non-uniform. It is most prominent for small  $M$ , and marginal as  $M$  increases. This is consistent with our intuition. The value of  $M$  should be selected based on the knowledge of network inhomogeneity such as sensor hole size. In our simulated sensor layout, the sensor hole is roughly the size of a two-row by four column grid. Hence 3-hop path is often sufficient to get to the side and further traverse around it. Further increasing  $M$  to 4 brings little gain in performance.

Fig. 8 visualizes the paths produced by the greedy CADR and the min-hop algorithm with  $M = 3$  applied to the same sensor network. The greedy algorithm path is plotted in Fig. 8a. The path gets stuck and spends most of hops (17 hops out of 20) bouncing between two nodes on the lower side of the sensor hole. The ending point, marked with a little square, is far away from the target.

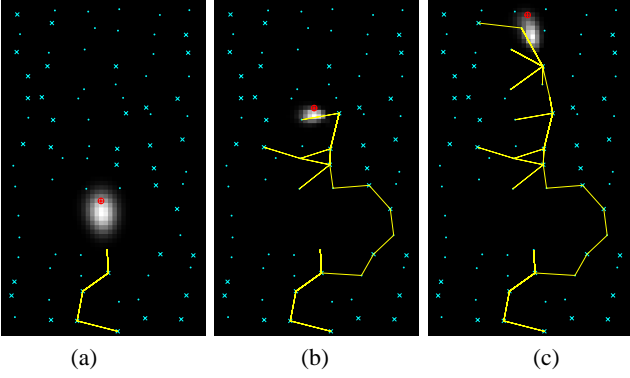
For comparison, the min-hop path is plotted in Fig. 8b. The path manages to get around the sensor hole, and ends at a node slightly above the true target location. The tracking performance is also much improved. The target location estimate is more accurate and has higher confidence.

### 7.1.2. Moving target

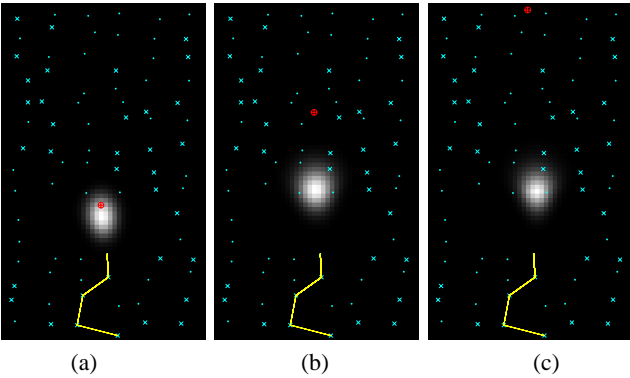
In tracking applications, target is often non-stationary. In principle, moving target can be considered as an extension of the stationary target case; the target can be considered as approximately stationary within a short time interval. With target moving, routing a query toward the high information content region is essentially routing a path to follow the target. Here we simulate a target moving along the straight line  $x = 125$  (the center line of the sensor field along the vertical dimension) with speed  $v = 7$  m/s. The query enters at the node closest to the initial target position (125, 0).

We compare the min-hop algorithm ( $M = 2, 3$  and 4) with the greedy CADR, with 100 independent runs for each. The performance is summarized in Table 5. Similar as in the stationary case, routing algorithms may get stuck at sensor holes. While the routing is stuck, the target estimate keeps worsening due to weaker signals. In simulation, a track is considered lost if by the time the vehicle reaches the upper side of the sensor field ( $y = 375$ ), the estimate of target location of the last five steps is on average more than 60m away from the true target location. We report the number of “lost” runs, and the statistics (MSE and belief size) for the good runs, averaged over all runs and all time steps. Here we observe similar characteristics as in the stationary case. With the greedy algorithm, most of the runs lost the target (93 out of 100). With increasing lookahead horizon  $M$ , the percentage of lost runs is reduced significantly. Among the good runs, the tracking performance improves with the increase of  $M$ . For example, with  $M = 3$ , the square root of the MSE is only one half of that obtained by the greedy algorithm. From these results we can see that the min-hop algorithm with a modest lookahead horizon ( $M = 2$  or 3) is much more robust against the presence of sensor holes.

Fig. 9 shows several snapshots of the routing path using the greedy algorithm. In all runs, the final path is 100-hop long. The figure shows the progressive path development after 30, 60, and 90 hops. Similar as in the stationary target case, the min-hop algorithm routes around sensor holes. The snapshots indicates that the path mainly follows the target movement, but occasionally reaches out to nearby sensors to aggregate information. The belief states follows the target fairly closely. For comparison, the path produced by the greedy CADR is plotted in Fig. 10. The path failed to get around the sensor hole within 30 hops and is stuck ever since. The belief state failed to track the target. As the target moves



**Fig. 9.** Routing query toward a moving target: path produced by the min-hop method with  $M = 3$ . The snapshots are at the end of 30, 60, and 90 hops. The target (marked with a “ $\oplus$ ”) is at location (125, 105), (125, 210), and (125, 315), respectively. The selected path is marked with solid lines.



**Fig. 10.** Routing query toward a moving target: path produced by the greedy CADR method. Simulation setting is the same as in Fig. 9.

away, the signal is too weak to correct the belief, hence the belief eventually missed the target completely. The failure is caused by the feedback between belief estimation and query routing: as the routing failed to aggregate information, the estimate can be poor; and as the estimation accuracy deteriorates, the estimated belief state provides little guidance to routing.

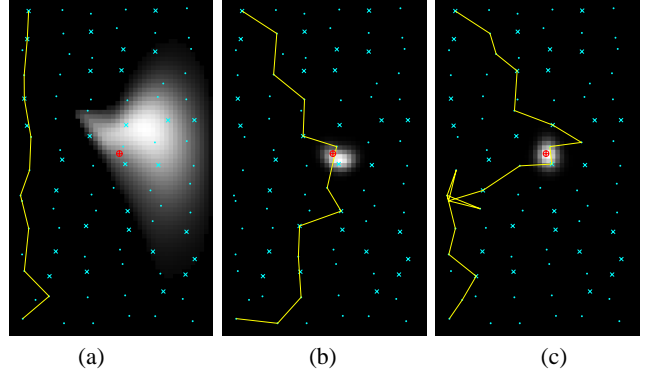
## 7.2. Query routing using forward search

The forward search routing from the query proxy to the exit node is tested with a stationary target at (125, 200). 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, 375). Forward search is performed on a homogeneous sensor network (as in Fig. 7a) to route query between these two nodes.

Recall from Sec. 6 that the information-directed routing problem is essentially a tradeoff between the communication expense and information aggregation, with the balance controlled by the hypothetical path length constraint  $C_0$ . We vary the allowance  $C_0$ ; for each value  $C_0$ , 100 independent runs are simulated. The numerical results are listed in Table 6. We use the shortest path as

$C_0$	sqrt(MSE)	belief size	# of hops
shortest path	26.71	883.88	9.91
350	22.54	664.21	10.48
450	14.29	270.72	13.07
550	10.18	175.57	15.70
650	8.49	152.20	18.44

**Table 6.** Information-directed routing using RTA\* forward search: results with different hypothetical path length constraints.



**Fig. 11.** Routing query from the proxy to exit node: (a) Shortest path: 10 hops, sqrt(MSE) = 43.18, belief size = 1176; (b) 12-hop path selected by the RTA\* algorithm, sqrt(MSE) = 3.70, belief size = 91; (c) 21-hop path selected by the RTA\* algorithm, sqrt(MSE) = 2.46, belief size = 105;

a benchmark for comparison (the left most point in the figures). The average number of hops increases with the hypothetical allowance  $C_0$ . It is the same order as  $C_0$  divided by the radio range of 50m, but about 35 — 50% larger. The margin accounts for the fact that sensors are not dense enough, and communication over the maximum radio range may not always be feasible. From Table 6 we can see that both the MSE and belief size decreases with path length. Compared to the shortest path, the information-directed routes take a little bit detour, but improve the tracking performance considerably. On average, the square root of MSE is cut to half with a 30% longer path. In sensor network practice, based on the application specifics, one can choose an efficient tradeoff at an affordable communication expense.

Fig. 11 visualizes selected paths with different length. The shortest path is shown in Fig. 11a. It has 10 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. Fig. 11b shows a longer path of 12 hops. Starting from the proxy, the path bends toward the target direction in attempt to accumulate information. The tracking performance is vastly better than the shortest path. Fig. 11c shows a path of 21 hops. The tracking accuracy is further improved, but the improvement is less prominent. The caption of Fig. 11 lists the detailed numerical results for comparison.

## 8. DISCUSSION

We have demonstrated the benefits of information-directed routing that jointly optimizes for maximal information gain and minimal communication cost. In the simulation study, we have shown that the min-hop algorithm, compared to the previous greedy algo-

gorithm, is 4–11 times more likely to succeed in routing a message around sensor holes with a 3-step lookahead, and at the same time produces 2–4 times less error in tracking a signal source. This significant improvement is obtained at the cost of additional computation at each decision node to search the graph of its  $M$ -hop neighborhood. However, the complexity of our algorithm grows only quadratically with the number of nodes in the neighborhood. Choosing an appropriate neighborhood size will allow us to obtain a sufficient amount of information at only a modest cost.

Knowledge about network structure or application plays an important role in assisting the information-directed routing algorithms such as the selection of  $M$ . In a homogeneous network, greedy routing algorithms such as CADR provide satisfactory result at a low computational cost. In the presence of holes in the network, a multi-step lookahead routing algorithms such as the min-Hop algorithm may be necessary. These algorithms search a local neighborhood beyond the immediate one-hop neighbors in the network. Additional knowledge about the network, such as sensor node density in a region, can be used to estimate information gain ahead in the forward search algorithm RTA\*. The parameter or structure of a network, for example node density or network holes, may be discovered and mapped out during the initialization phase. If such knowledge is made available, then online routing algorithms can use either a greedy or a multi-step search and switch as appropriate to minimize the overhead. Likewise, knowing the distribution of physical stimuli can help plan data aggregation and routing accordingly. In general, a priori knowledge about a network or application should be exploited whenever possible to assist routing in a sensor network. If such a priori knowledge is not available, one may resort to an online iterative probing strategy to explore the network inhomogeneity.

While we presented information-directed routing using 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 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 stay the same.

We presented the routing algorithms for scenarios with a single stimulus. The algorithms can be generalized to handle multiple stimuli, using a spanning tree such as Steiner tree [17]. While computing an exact Steiner tree is very expensive, approximate algorithms exist. For example, a greedy Steiner Tree algorithm produces a tree that is no  $\log k$  factor worse than the optimal tree [18, 19], where  $k$  is the number of leaves in the tree. Generalizing the algorithms described in this paper to handle dynamically moving stimuli while maintaining good approximations to the optimal routing tree remains as a future research topic.

## 9. 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 (dsv) 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] 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.
- [13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: John Wiley and Sons, Inc., 1991.
- [14] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.
- [15] 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.
- [16] R. Korf, "Real-time heuristic search," *Artificial Intelligence*, vol. 42, pp. 189–211, 1990.
- [17] 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.
- [18] M. Imase and B. M. Waxman, "Dynamic Steiner tree problem," *SIAM J. Discrete Math.*, vol. 4, pp. 369–384, 1991.
- [19] N. Alon and Y. Azar, "On-line Steiner trees in the Euclidean plane," *Discrete Comput. Geom.*, vol. 10, pp. 113–121, 1993.

## **Author Information**

### **Juan Liu**

Juan Liu received her B.E. degree in electronic engineering from Tsinghua University, China, in 1995, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, USA, in 1998 and 2001, respectively. In September 2001, she joined Palo Alto Research Center as a Research Scientist, working in the Embedded Collaborative Computing Area. Her research interests include signal processing, statistical modeling, detection and estimation, network routing, and their applications to distributed sensor network problems.

### **Feng Zhao**

Feng Zhao is a senior research at Microsoft Research, where he manages the Networked Embedded Computing Group. His current interest is in developing algorithms and software for interconnected devices such as wireless sensor networks. He is well-known for his work in networked embedded systems, distributed algorithms, and artificial intelligence. Dr. Zhao received his Ph.D. in Electrical Engineering and Computer Science from MIT and has taught at Stanford University and Ohio State University. He was a principal scientist at Xerox PARC and directed PARC's sensor network research effort. He is serving as the Editor-In-Chief of ACM Transactions on Sensor Networks, and recently co-authored a book, *Wireless Sensor Networks: An information processing approach*, published by Morgan Kaufmann.

### **Dragan Petrovic**

Dragan Petrovic was born in Nis, Serbia-Montenegro in 1979. He received his B.S. degree in Computer Engineering with highest honors from the University of Illinois at Urbana-Champaign in 1999 and his M.S. degree in Electrical Engineering in 2001 from the University of California, Berkeley. He is currently pursuing a Ph.D. at the University of California Berkeley under the supervision of Profs. Kannan Ramchandran and Jan Rabaey. His research interests include compression, routing, and reliable data transfer in sensor networks as well as error control coding.