nLRTS: Improving Distance Vector Routing in Sensor Networks

Greg Lee and **Vadim Bulitko** and **Ioanis Nikolaidis**
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{greglee|bulitko|yannis}@cs.ualberta.ca

Abstract

Routing in *ad hoc* sensor networks is an important problem with a broad spectrum of applications. Borrowing from the literature on real-time heuristic search, we introduce backtracking and controlled suboptimality to sensor routing, with the intention to improve flexibility by accounting for the restricted capabilities of sensor nodes and the need for prompt delivery of data. The resulting novel algorithm, nLRTS, is compared against a well-known routing method, Distance-Vector Routing (DVR), with respect to several metrics. nLRTS demonstrates improvement in asychronous networks with limited energy reserves.

Introduction

Sensor networks are an important sector in *ad hoc* network research. Sensor nodes generally have limited computational power and energy, thus simple, energy efficient routing algorithms are typically used. Various applications are important, including military (Y.Shang *et al.* 2003), environmental monitoring (Braginsky & Estrin 2002) and reconnaisance missions (Yan, He, & Stankovic 2003). As such, routing in sensor networks is of interest to a large number of people, in many different facets of industry.

Routing in *ad hoc* networks has spawned many different algorithms, each with their own advantages and disadvantages (Royer & Toh April 1999). Two families of routing algorithms have been identified, table-driven (proactive) routing and source-initiated on-demand (reactive) routing. Table-driven methods store routing information before it is required, so that when a need arises, a route has already been computed. Source-initiated methods only generate routes when they are actually demanded in the network. Table-driven methods generally consume much memory, and thus source-initiated methods are preferred in many scenarios.

It is important to note that in sensor networks, it is expensive to maintain routing information proactively (i.e., before it is needed) within an individual sensor's memory, since sensors generally have limited storage and computing abilities. Moroever, even learned routes may need to be "forgotten" to save space since sensor devices usually include tens or a few hundreds of kilobytes of R/W memory in which they accomodate not only routing information but also all pertinent data related to the specific application. Hence, the problem becomes that of *quickly* determining routes in an as–needed fashion that allows them to be easily forgotten/erased yet subsequently quickly recomputed.

Along another line of research, real-time search algorithms have been used extensively in Artificial Intelligence (AI) in situations where there exists a given source and destination, and decisions must be made based on limited information and in limited time (Korf 1990; Bulitko & Lee 2006). As such, these methods are applicable to routing problems in *ad hoc* networks, and we investigate such applicability in this paper.

The contribution of this paper is not in establishing any significant increase in routing performance but in showing that the tools of real–time search algorithms developed by the AI community are equally well–suited for routing in sensor networks, and, in fact, result in quite competitive performance. Moreover, their generalized approach to dealing with path constraints provides a natural setting to explore routing with a variety of resource constraints, not just energy constraints. The focus of this paper is mainly on determining if real-time heuristic search is competitive with respect to energy, and hence network lifetime.

The rest of the paper is organized as follows. In the next section, we outline the problem addressed. This if followed by sections describing related work in real-time heuristic search and extensions to previous algorithms. Next, we describe the simulation used in the paper and then demonstrate how the novel methods improve routing in sensor networks.

Problem Formulation and Challenges

Research in this paper is based upon previous work conducted with real-time heuristic search methods in sensor networks. In (Y.Shang *et al.* 2003), the authors adapt a classic real-time heuristic search algorithm (LRTA*, (Korf 1990)) for routing in *ad hoc* sensor networks. The resulting algorithm, called Constraint-Based Learning Real-Time A* (CB-LRTA*), takes routing and destination constraints into account, and the user controls the main objectives of the network, such as minimizing energy consumption, maintaining confidentiality, and minimizing hops taken to reach the goal. Several other routing algorithms, such as DSR (Johnson & Maltz 1996) and GPSR (Karp & Kung 2000), are mentioned,

but none of them is compared to CB-LRTA* in the paper.

In our work, we build on a much more recent real-time heuristic search algorithm called LRTS (Bulitko & Lee 2006) – a product of 15 years of research that had taken place in the AI community since the original LRTA*. We extend LRTS for routing in *ad hoc* sensor networks and call the new version nLRTS. We focus primarily on the case where sensor nodes are dropped from the sky to monitor territory, landing in a random dispersion, thus necessitating *ad hoc* routing in order to facilitate data transmissions. We are interested in sensor networks of several thousand nodes where the sensing is performed by triggering events that result in a sensor needing to communicate with a far away sensor. For example, a motion detector sensor could send commands to another sensor to turn on the lights in a building. We consider the case of a fixed destination for each transmission, simulating each source node providing information to a central information hub. We experiment with different network topologies in an attempt to learn which algorithms are best for which situations. Also, different numbers of messages and message repetitions are used in an attempt to determine which algorithms are better when more messages are sent, and which are best when the same message is continually repeated. Messages are *repeated* in the sense that a source may not be able to send everything needed in one transmission to a destination, and may need to send multiple messages to deliver the full information. An example of this would be a message that is three times the size of the maximum packet that can be sent by a node, thus requiring three separate messages to be sent from the source to the destination. A by-product of this is a study of how well our algorithm converges (by merit of being iterative in nature) when multiple messages are sent from a given source to a given destination.

Related Work

Learning Real-Time A* (LRTA*) is a heuristic[1] search method that updates its heuristics while it acts in the environment during a search (Korf 1990). It interleaves learning, planning, and execution and, given enough search trials, converges to an optimal solution.

Search proceeds in cycles, each consisting of planning, learning, and execution (Figure 1). On each cycle a particular node is called *current* and serves as a centre of the local search. Such a current node s examines its (immediate) neighboring nodes and for each neighbor s' computes $f(s') = g(s, s') + h(s, s')$, where g is the distance from state s to s' (known precisely), and h is a heuristic estimate of the distance from s' to the closest goal state. The node then *plans* to move the neighbor s'' with the smallest f value. If the node s's current h is smaller than $f(s'')$, the node updates its to $f(s'')$, since the distance to the goal must be at least as large as the distance of going through the neighbor with the minimum f value. This step is called *learning*. Finally, the planned neighbor s'' becomes the next current state (this is called *execution*).

LRTA*'s analogue used in networks, is Distance Vector Routing (DVR) (Royer & Toh April 1999). In DVR, nodes maintain a routing table with (estimated) distances to all other nodes, which they periodically advertise to their neighbours. Thus, for our purposes, LRTA* and DVR are conceptually equivalent. One small difference in classic DVR is that the initial cost to any destination is set to infinity, whereas in classic LRTA* these values are set to zero. In our work, we set the initial distance to any destination to be the minimum distance to the nearest neighbour.

LRTA*

```
1  initialize the heuristic: h ← h_0
2  reset the current state: s ← s_start
3  while s ∉ S_g do
4     generate children one move away from state s
5     find the state s' with the lowest f(s') = g + h
6     update h(s) to f(s') if f(s') is greater
7     execute the action to get to s'
8  end while
```

Figure 1: LRTA* algorithm with a lookahead of one.

A more recent algorithm, called Learning Real-Time Search (LRTS), extended the LRTA* in Figure 1 with three enhancements: deeper lookahead, optimality weighting and backtracking control (Bulitko & Lee 2006). LRTS has demonstrated an impressive performance in combinatorial puzzles and two-dimensional path-finding. In the latter, for instance, LRTS' extensions improved the convergence speed by two orders of magnitude while converging to paths within 3% of optimal.

In this paper, we adapt two of LRTS' extensions to the specifics of routing in *ad hoc* wireless networks. These extensions are described below. First, the distance from the current state to the state on the frontier (i.e., the furthest visible state) is weighted by $\gamma \in (0, 1]$. This allows one to trade-off the quality of the final solution and the convergence travel. This extension of LRTA* is equivalent to scaling the initial heuristic by the constant factor of $1 + \varepsilon = 1/\gamma$ (Shimbo & Ishida 2003). Bulitko (Bulitko & Lee 2006) proved that γ-weighted LRTA* will converge to a solution no worse than $1/\gamma$ of optimal. In practice, much better paths are found (Bulitko & Lee 2006). A similar effect is observed in weighted A*: increasing the weight of h dramatically reduces the number of states generated, at the cost of longer solutions (Korf 1993).

Second, backtracking within LRTA* was first proposed in (Shue & Zamani 1993). Their SLA* algorithm used the lookahead of one and the same update rule as LRTA*. However, upon updating (i.e., increasing) the heuristic value in a state, the agent moved (i.e., backtracked) to its previous state. Backtracking increases travel on the first trial but re-

[1] We use the term *heuristic* with the meaning used by the AI community. That is, a heuristic is a function guiding the search process by which the optimal solution can be derived, regardless of the (possibly non-polynomial) time complexity of the search. This is to distinguish it from heuristics in the systems/networking community, where a heuristic search, by design, does not necessarily reach the optimum solution when it terminates, but typically runs in polynomial time complexity and produces a "reasonably good" solution.

duces the convergence travel (Bulitko & Lee 2006). Note that backtracking does not need to happen after *every* update to the heuristic function. SLA*T, introduced in (Shue, Li, & Zamani 2001), backtracks only after the cumulative amount of updates to the heuristic function made on a trial exceeds the learning quota (T). We will use an adjusted implementation of this idea as presented in (Bulitko & Lee 2006).

Description of Novel Methods Proposed

DVR with two extensions adopted from LRTS is presented in Figure 2 and henceforth referred to as nLRTS (network Learning Real-Time Search). A sensor node s running nLRTS operates as follows. When sent a data message m from another sensor node s_f, the node s first checks whether it is the destination (d) node for this message (line 1, Figure 2). If s is not the destination, it finds the most promising neighbour (line 4). If the node's previously estimated shortest distance to the goal (h_d) is less than the distance through its most promising neighbour($f_{s'_d}$), lines 5 and 6 update h_d to $f_{s'_d}$, assuming the heuristic is admissible (non-overestimating). Node s then broadcasts its updated h_d to all its neighbours so they can update their own heuristic estimates for destination d (line 7). This constitutes the sending of a control message. Node s then checks if the cumulative learning amount u will exceed the learning threshold T (the maximum amount of learning allowed before backtracking occurs) if the learning amount at this juncture $|\Delta f_d|$ is added to it. If so, the learning threshold will have been exceeded, and s bounces the message back to the sender s_f in line 10. Otherwise it updates u and sends the message m to the most promising neighbour (i.e., s') in lines 12, 13. Whenever a message is transmitted in lines 10 and 13, the cumulative learning amount u is sent along.

For all nodes n, the estimated distance h_d to all other nodes n_d is initialized to one hop, since this is the distance from n to its closest neighbour n_c. This ensures the h_d heuristic values are admissible. A neighbour discovery process is run prior to any data messages being sent in order for each node to know its one-hop neighbours and for each node to be able to initialize its heuristics.

nLRTS(γ, T, u, m, s_d)

```
1   if s = s_d then
2     stop
3   else
4     find the neighbour s' with the lowest f_{s'_d} = γ · g_{s'_d} + h_{s'_d}
5     if h_d < f_{s'_d} then
6       update h_d ← f_{s'_d}
7       send updated h_d to all neighbours
8     end if
9     if u + |Δh_d| ≥ T then
10      return m and u to s_f
11    else
12      increase amount of learning u by |Δh_d|
13      send m and u to s'
14    end if
```

Figure 2: The novel nLRTS algorithm proposed in this paper.

Experimental Setup

In order to parallel the experimental setup of previous work (Y.Shang *et al.* 2003), we have developed a simulator in C++.[2] In our experiments, N sensor nodes are randomly distributed in an $X \times Y$ space, with N, X, Y determined by the user. Nodes are ensured not to occupy the same location. Each node is given a transmission radius range of 5 or 10 units (as listed below). Energy consumption is modeled by counting 1.2 energy units for every message reception, and 1.7 units for every message transmission at a node, according to the values provided by (Xu *et al.* 2003). Similarly to (Y.Shang *et al.* 2003), all nodes are always awake (i.e., never go into a sleep mode), until they deplete their energy reserves.

The number of messages sent in the routing process is counted, as a measure of network activity. Any time a transmission of a data or control packet is sent, this counter is incremented. The second measure is the data and control message traffic. Traffic is measured in the same manner as messages, except that data packets are counted as three times a control message. Since we are working with sensor nodes, we assume data transmissions are small, perhaps sending a simple temperature value to another node. Control messages are sent prior to data messages in order to help discover routes for the actual data to be transmitted. Each packet is considered an individual message, with repetitions often necessary to send larger pieces of data. Each message (or repetitions thereof) are sent from a fixed source to a fixed destination.

In order to simulate the concurrent behavior of the real network, we implemented the simulator as a multi-threaded application wherein the threads share the memory where the simulation objects reside. Interference is not modeled in this simulation. We assume scheduling is performed independently of the routing strategy, and that different schedules will not severely affect the performance of any of the routing algorithms used in experiments. Also, since delays and timing are not explicitly modeled, it would be difficult to model interference accurately.

We experiment with both unlimited and fixed battery life for sensors. We considered unlimited energy for nodes in some of our experiments because battery life is application-dependent. We measure the cumulative energy used (as well as the energy at each node), as a method of determining which algorithms will deplete the entire system of energy before others. The source for each transmission is determined randomly, and there is a guaranteed path to the destination.

Empirical Results

Our previous research (Bulitko & Lee 2006) applied LRTS to a special case of sensor networks with synchronous message transmission and unlimited battery life. The messages were sent between a given source and destination pair until convergence. In order to make this paper as self-contained

[2]The original simulator was not available due to export restrictions.

Table 1: Details of the experiments

Network Specifications	
Topologies: (Number of nodes, X and Y dimensions)	{50,20,20}, {100,30,30}, {200,50,50}, {400,50,50}, {400,80,80}, {600,80,80}, {800,80,80}
Range of messages sent:	5 – 1000
Message repetitions:	1 – 1500,
Type of Destination:	Randomly chosen fixed location for all messages
Minimizing Objective:	Hops
nLRTS Parameters	
γ	0.0001, 0.001, 0.05, 0.1, 0.3. 0.5. 0.7, 1
T	0, 10, 100, 1000, ∞

as possible, section replicates these experiments (including figures and tables), and follows up with novel results involving asynchronous messages and pre-convergence experiments. In section , we show results with sensors of limited battery life (again with asynchronous messages and no convergence). Extensive experiments were performed, with the parameter settings listed in Table 1. Throughout this section, we will use the notation $\{X, Y, Z\}$ to denote a network of X nodes of $Y \times Z$ dimensions.

Sensor nodes with unlimited battery life

In order to compare the performance of nLRTS in sensor networks to LRTS performance in traditional real-time search domains, we limited the network to sending one message at a time, and sent the message continuously until there were no heuristic updates made during its delivery. In these experiments, four batches of networks of 50, 200, 400, and 800 nodes each were considered. The nodes were randomly positioned on a square grid of the dimensions 20×20, 30×30, 50×50, and 80×80 respectively. Each batch consisted of 100 randomly generated networks. For each network, 25 convergence runs were performed with the parameter values running $\gamma = \{0.1, 0.3, 0.5, 0.7, 1\}$ and $T = \{0, 10, 100, 1000, 100000\}$. In the following, we present the results from the batch of 800-node networks. The same trends were observed in smaller networks.

Synchronous Messaging: We first experimented with synchronous messages, wherein only one message is sent at a time in the network. We start out by demonstrating the influence of the two control parameters in isolation. Table 2 shows the influence of the heuristic weight γ with no backtracking. The execution convergence cost is the sum of the path lengths for each message sent, until convergence. Smaller values of γ accelerate the convergence, but increase the suboptimality of the final solution. These results are also reported in (Bulitko & Lee 2006).

Table 2: Effects of heuristic weighting in network routing (from (Bulitko & Lee 2006)).

Heuristic weight γ	Convergence cost	Suboptimality
1.0 (DVR)	8188	0%
0.7	8188	0%
0.5	8106	0%
0.3	7972	0.29%
0.1	7957	0.33%

Table 3 demonstrates the influence of the learning quota T when the heuristic weight γ is set to one. Smaller values of T increase the amount of backtracking and speed up convergence cost but lengthen the first trial. This also parallels the results in path-finding in (Bulitko & Lee 2006).

Table 3: Effects of backtracking in network routing (from (Bulitko & Lee 2006)).

Learning quota T	First trial cost	Convergence cost
10^5 (DVR)	549	8188
10^3	1188	8032
10^2	4958	6455
10	5956	6230
0	6117	6138

We will now consider the impact of parameter combination. Figure 3 shows the first trial and convergence execution costs for all combinations of γ and T. Brighter areas indicate higher costs. Figure 4 shows the suboptimality of the final solution and the total traffic on a convergence run, averaged over 100 networks of 800 nodes each. Again, brighter areas indicate higher values.

In summary, by extending the DVR algorithm with the heuristic weight and the backtracking mechanisms of LRTS, reduction of the convergence execution cost and the total network traffic can be achieved. As a result, near-optimal routes are found faster, at a lower energy consumption.

Asynchronous Messaging: We next considered asynchronous messaging (multiple messages sent simultaneously), since in a true sensor network, there are generally many messages being relayed at the same time. The interleaving of discovery messages (control messages) results in an arbitrary sequence of refinements of the paths, and there can be cases where an earlier control message has "helped" a later one. We analyze the effect of γ on several network metrics. In Table 4 we see how the total number of hops taken during the transmission of multiple messages can be reduced by decreasing γ.

Table 5 shows that with more repetitions of the same message, decreasing γ reduces the total energy consumed by all sensors in the network. Table 6 shows that with more repetitions of the same message, decreasing γ reduces the total traffic present in the network. Note that in Tables 4, 5 and 6, the number of hops taken, amount of energy consumed and amount of traffic in the network do not decrease propor-

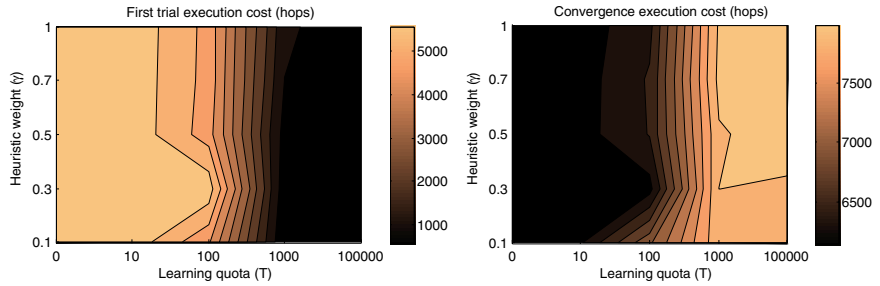

Figure 3: First-trial and convergence execution costs in network routing (from (Bulitko & Lee 2006)).

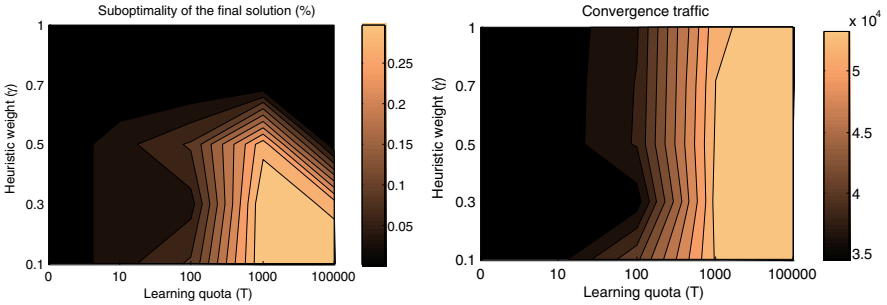

Figure 4: Suboptimality of the final solution and the traffic in network routing (from (Bulitko & Lee 2006)).

Table 4: Total number of hops taken when using various values for γ.

		Total Number of Hops	
T	γ	1000 messages $\times 1$	1000 messages $\times 10$
∞	1	6205	11914
∞	0.5	6143	11832
∞	0.1	6209	11832

Table 5: Total energy consumed in the network when using various values for γ

		Total Energy Consumed	
T	γ	1000 messages $\times 1$	1000 messages $\times 10$
∞	1	12367	64446
∞	0.5	12600	63013
∞	0.1	12618	62574

tionally with the number of repetitions of messages. This is because better paths are learned as messages are repeated.

Table 7 shows how nLRTS can improve sensor network performance with respect to the four different metrics, when compared to DVR. The results show the improvement possible by tuning the parameters for a given topology and number of messages sent.

Finally, we observe that the backtracking parameter T appears to have no significant influence on the network metrics considered. The early cost of backtracking was likely balanced by the savings earned later with the previously learned heuristic values.

Table 6: Total traffic when using various values for γ

		Total Network Traffic	
T	γ	1000 messages $\times 1$	1000 messages $\times 10$
∞	1	25479	46052
∞	0.5	25483	45395
∞	0.1	25014	45243

Sensor nodes with limited battery life

Since real sensor nodes have a limited amount of energy, they can go out of commission during transmission of messages. The remaining nodes are then forced to find alternate routes to the destination. Sometimes enough nodes can be decommissioned so as to make it impossible to deliver a message from source to destination. In the following we describe how nLRTS handles this scenario in an asynchronous network.

Table 8 shows the number of messages lost, that is, not delivered to the destination, for various parameters of nLRTS. It also shows the total number of hops taken in delivering successful messages. Here we supply the nodes with an amount of energy relative to the total number of messages sent (so we give the nodes more energy at the onset if more messages will be sent in the simulation).

With the amount of learning before backtracking takes place T set to 0, messages in the network reach the destination with the same frequency as with $T = \infty$ (DVR), while at the same time successful deliveries take fewer hops. The learning that takes place with the backtracking after every learning step leads to eventual better performance, without costing the network nodes any more energy. Thus, nLRTS is capable of outperforming DVR in this application.

Table 7: nLRTS performance typically increases with more repetitions of messages. The improvement percentages shown are based on the best choice of parameter values for nLRTS for the given network specifications (from among the parameter values considered in Figure 1)

Nodes,Area	Distinct Messages	Repetitions	Improvement with nLRTS Total Hops	Energy	Transmissions	Traffic
50, 20×20	5	1	0%	0%	1%	1%
50, 20×20	5	5	2%	4%	4%	4%
50, 20×20	5	10	5%	7%	7%	6%
50, 20×20	10	1	0%	0%	0%	0%
50, 20×20	10	5	3%	4%	5%	4%
50, 20×20	10	10	3%	5%	6%	5%
50, 20×20	50	1	0%	0%	1%	0%
50, 20×20	50	5	1%	2%	2%	1%
50, 20×20	50	10	2%	3%	4%	3%
50, 20×20	1000	1	0%	0%	0%	0%
50, 20×20	1000	5	0%	0%	0%	0%
50, 20×20	1000	10	0%	0%	0%	0%
100, 30×30	5	1	0%	0%	1%	1%
100, 30×30	5	5	6%	6%	7%	6%
100, 30×30	5	10	3%	3%	4%	3%
100, 30×30	10	1	0%	1%	1%	1%
100, 30×30	10	5	1%	1%	3%	2%
100, 30×30	10	10	4%	5%	5%	5%
100, 30×30	50	1	1%	0%	1%	0%
100, 30×30	50	5	3%	4%	6%	5%
100, 30×30	50	10	1%	2%	2%	1%
100, 30×30	1000	1	0%	0%	0%	0%
100, 30×30	1000	5	0%	0%	1%	0%
100, 30×30	1000	10	0%	0%	0%	0%

Table 8: Number of hops taken and messages not delivered with three different values of T (γ is fixed at 1). With more repetitions a smaller proportion of the messages are missed by each strategy. Setting $T = 0$ (backtracking whenever learning takes place) leads to paths with fewer hops than DVR ($T = \infty$).

	50 messages $\times$ 160		50 messages $\times$ 320	
T	Hops	Missed	Hops	Missed
∞	5140	4.2%	8472	3.6%
100	5210	4.3%	8813	3.6%
0	4951	4.3%	8361	3.5%

Conclusions and Future Work

In this paper, we extend a well-known routing algorithm, Distance Vector Routing, by adding further functionality to this method, namely backtracking and convergence to suboptimal solutions. The resulting algorithm is called nLRTS. These extensions have been previously unified with a learning real-time search algorithm called LRTS. An evaluation of nLRTS in synchronous sensor networks with unlimited energy demonstrated its advantages (Bulitko & Lee 2006). In this paper, we follow up with an application of nLRTS to the more realistic cases of asynchronous networks and nodes with limited battery life. Allowing for more suboptimality and backtracking gives the nodes using the nLRTS algorithm the ability to find better paths from source to destination.

Future work includes narrowing the parameter consideration within nLRTS while gaining a better understanding of which parameter values to use in which situations. Also we would like to add the constraints present in (Y.Shang *et al.* 2003) and observe the performance of nLRTS. Finally we would like to apply nLRTS to routing in mobile sensor networks.

Acknowledgement

We appreciate contributions by Kit Barton and David O'Connell. We are also grateful for the support from the University of Alberta, the Natural Sciences and Engineering Research Council (NSERC), the Informatics Circle of Research (iCORE), and the Alberta Ingenuity Centre for Machine Learning (AICML).

References

Braginsky, D., and Estrin, D. 2002. Rumor routing algorithm for sensor networks. In *International Conference on Distributed Computing Systems (ICDCS-22)*.

Bulitko, V., and Lee, G. 2006. Learning in real time search: A unifying framework. *Journal of Artificial Intelligence Research* 25:119 – 157.

Johnson, D. B., and Maltz, D. A. 1996. Dynamic source routing in ad hoc wireless networks. In Imielinski, and Korth., eds., *Mobile Computing*, volume 353. Kluwer Academic Publishers.

Karp, B., and Kung, H. 2000. Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 243–254.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2-3):189–211.

Korf, R. E. 1993. Linear-space best-first search. *Artificial Intelligence* 62:41–78.

Royer, E., and Toh, C. April 1999. A review of current routing protocols for ad hoc mobile wireless networks. In *IEEE Personal Communications*, volume 6, 46–55.

Shimbo, M., and Ishida, T. 2003. Controlling the learning process of real-time heuristic search. *AIJ* 146(1):1–41.

Shue, L.-Y., and Zamani, R. 1993. An admissible heuristic search algorithm. In *Proceedings of the 7th Int. Symp. on Methodologies for Intel. Systems (ISMIS-93)*, volume 689 of *LNAI*, 69–75.

Shue, L.-Y.; Li, S.-T.; and Zamani, R. 2001. An intelligent heuristic algorithm for project scheduling problems. In *Proceedings of the 32nd Annual Meeting of the Decision Sciences Institute*.

Xu, Y.; Bien, S.; Mori, Y.; Heidemann, J.; and Estrin, D. 2003. Topology control protocols to conserve energy inwireless ad hoc networks. Technical Report 6, University of California, Los Angeles, Center for Embedded Networked Computing. www.isi.edu/˜johnh/PAPERS/Xu03a.html.

Yan, T.; He, T.; and Stankovic, J. A. 2003. Differentiated surveillance for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, 51–62.

Y.Shang; Fromherz, M.; Y.Zhang; and Crawford, L. 2003. Constraint-based routing for ad-hoc networks. In *IEEE International Conference on Information Technology*, 500–505.