A ONE-PASS HEURISTIC FOR COOPERATIVE COMMUNICATION IN MOBILE AD HOC NETWORKS

CLAYTON W. COMMANDER
Air Force Research Laboratory, Munitions Directorate, and
Dept. of Industrial and Systems Engineering,
University of Florida, Gainesville, FL USA.
clayton.commander@eglin.af.mil

CARLOS A.S. OLIVEIRA
School of Industrial Engineering and Management
Oklahoma State University, Stillwater, OK USA.
coliv@okstate.edu

PANOS M. PARDALOS
Center for Applied Optimization,
Dept. of Industrial and Systems Engineering,
University of Florida, Gainesville, FL USA.
pardalos@ufl.edu

MAURICIO G.C. RESENDE
Internet and Network Systems Research Center
AT&T Labs Research, Florham Park, NJ USA.
mgcr@research.att.com

Ad hoc networks have been used in the last few years to provide communications means among agents that need to accomplish common goals. Due to the importance of communication for the success of such missions, we study the problem of maximizing communication among a set of agents. As a practical tool to solve such problems, we introduce a one-pass randomized algorithm that maximizes the total communication, as measured by the proposed objective function. Agents in this problem are routed along the edges of a graph, connecting their individual starting nodes to their respective destination nodes. This problem, known as the *Cooperative Communication Problem in Mobile Ad Hoc Networks*, is known to be NP-hard. We present a new heuristic and motivate the need for more advanced methods for the solution of this problem. In particular, we describe 1) a construction algorithm and 2) a local improvement method for maximizing communication. Computational results for the proposed approach are provided, showing that instances of realistic size can be efficiently solved by the algorithm.

1. Introduction

Advances in wireless communication and networking have lead to the development of new network organizations based on autonomous systems. Among the most important example of such networks systems are mobile ad hoc networks (MANETs). MANETs are composed of a set of loosely coupled mobile agents which communicate using a wireless medium via a shared radio channel. Agents in the network act as both clients and as servers and use various multi-hop protocols to route messages to other users in the system[17,18]. Unlike traditional cellular systems, ad hoc networks have no fixed topology. Moreover, in a MANET the topology changes each time an agent changes its location. Thus, the communication between the agents depends on their physical location and their particular radio devices.

Interest in MANETs has surged in the recent years, due to their numerous civilian and military applications. MANETs can be successfully implemented in situations where communication is necessary, but no fixed telephony system exists. Real applications abound, especially when considering adversarial environments, such as the coordination of unmanned aerial vehicles (UAVs) and combat search and rescue groups. Other examples include the coordination of agents in a hostile environment, sensing, and monitoring. More generally, the study of protocols and algorithms for MANETs is of high importance for the successful deployment of sensor networks – which are themselves composed of a large number of autonomous processors that can coordinate to achieve some higher level task such as sensing and monitoring.

The lack of a central authority in MANETs leads to several problems in the areas of routing and quality assurance[6]. Many of these problems can be viewed as combinatorial in nature, since they involve finding sets of discrete objects satisfying some definite property, such as for example connectedness or minimum cost. Among the challenging problems encountered in MANETs, we can cite routing as one of the most difficult to solve, because of the temporary nature of communication links in such a system. In fact, as nodes move around, they dynamically define topologies for the entire network. In such an environment, it is difficult to determine if two nodes are connected, since any of the intermediate nodes may leave the network at any time.

This scenario makes clear the importance of close coordination among groups of nodes if a definite goal needs to be attained. If at all possible, a plan must be devised such that communication among nodes is maintained for as long as possible. With this objective in mind, we study in this paper algorithms that would allow a set of agents to accomplish a mission, with definite starting and ending positions, but at the same time maximizing the communication time among the agents involved.

This chapter is organized as follows. We first present a graph formulation and discuss the computational complexity of the problem in Section 2. Next, in Section 3 we discuss some of the previous work on areas related to cooperative communication in wireless systems. Potential solution techniques are presented in the following sections. In Section 4, a simple construction algorithm for the maximum communication problem is proposed. In Section 5, a hill-climbing method for improving an initial solution for the problem is presented. Then, in Section 6 a one-pass local search heuristic is presented, combining the ideas discussed on the preceding sections. The numerical results from computational experiments are analyzed on Section 7. Finally, concluding remarks and future research ideas are presented in Section 8.

2. Problem Statement

In MANETs, bandwidth and communication time are usually severely constrained resources. To allow for successful interaction among nodes, we propose the solution of the *cooperative communication problem on ad hoc networks* (CCPM). In this problem, the objective is to determine a set of routes to be followed by mobile agents who must cooperate to accomplish some preassigned tasks. The function we try to maximize represents the total communication time for all agents along the computed trajectories. As described in this section, the *Cooperative Communication Problem* in MANETs can be modeled as a combinatorial optimization problem on graphs.

Consider a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of candidate positions for the wireless agents. Suppose that a node in G is connected only to those nodes that can be reached in one unit of time. Let U represent the set of agents, $S = \{s_1, s_2, \ldots, s_{|U|}\} \subseteq V$ represent the set of initial positions, and $D = \{d_1, d_2, \ldots, d_{|U|}\} \subseteq V$ the set of destination nodes. Let $N(v) \subseteq 2^V$, for $v \in V$, represent the set of neighbors of node v in G. Given a time horizon T, the objective of the problem is to determine a set of routes for the agents in U, such that each agent $u_i \in U$ starts at a source node s_i and finishes at the destination node $d_i \in D$ after at most T units of time.

For each agent $u \in U$, the function $p_t : U \to V$ returns the position of the agent at time $t \in \{1, 2, \dots, T\}$, where T is the time limit by which the agents must reach their destinations. Then at each time instant t, an agent $u \in U$ can either remain in its current location, i.e., $p_{t-1}(u)$, or move to a node in $N(p_{t-1}(u))$.

We can represent a route for an agent $u \in U$ as a path $\mathcal{P} = \{v_1, v_2, \dots, v_k\}$ in G where $v_1 = s_u$, $v_k = d_u$, and, for $i \in \{2, \dots, k\}$, $v_i \in N(v_{i-1}) \cup \{v_i\}$. Finally, if $\{\mathcal{P}_i\}_{i=1}^{|U|}$ is the set of trajectories for the agents, we are given a corresponding vector $\mathcal{L}$ such that $\mathcal{L}_i$ is a threshold on the size of path $\mathcal{P}_i$. This value is typically determined by fuel or battery life constraints on the wireless agents.

We assume that the agents have omnidirectional antennas and that two agents in the network are connected if the distance between them is less than some radius r. More specifically, let $\delta : V \times V \to \mathbb{R}$ represent the Euclidean distance between a pair of nodes in the graph. Then, we can define a function $c : V \times V \to \{0, 1\}$ such that

$$c(p_t(u_i), p_t(u_j)) = \begin{cases} 1, & \text{if } \delta(p_t(u_i), p_t(u_j)) \le r \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

With this, we can define the CCPM as the following optimization problem:

$$\max \sum_{t=1}^{T} \sum_{u,v \in U} c(p_t(u), p_t(v)) \tag{2}$$

$$\text{s.t.} \quad \sum_{j=2}^{n_i} \delta(v_{j-1}, v_j) \le \mathcal{L}_i \quad \forall\, \mathcal{P}_i = \{v_1, v_2, \dots, v_{n_i}\}, \tag{3}$$

$$p_1(u) = s_u \quad \forall\, u \in U, \tag{4}$$

$$p_T(u) = d_u \quad \forall\, u \in U, \tag{5}$$

where constraint (3) ensures that the length of each path $\mathcal{P}_i$ is less than or equal to its maximum allowed length $\mathcal{L}_i$.

Oliveira and Pardalos[17] have shown that CCPM is $\mathcal{NP}$-hard via a reduction to 3SAT[1]. Furthermore, to find an optimal solution at each time $t \in \{1, 2, \dots, T\}$ remains $\mathcal{NP}$-hard. This can be shown by a reduction to MAXIMUM CLIQUE[1], another well-known $\mathcal{NP}$-hard problem. The computational complexity of the problem does not allow for real-world instances to be solved exactly. This motivates the need for efficient heuristics to solve real-world instances within reasonable computing times. In the upcoming sections, we describe one such algorithm and present the framework for a different, non deterministic heuristic.

3. Previous Work

Communication is an important measure of collaboration between entities involved in a mission. It allows different agents to perform the set of tasks that have been planned, and at the same time to implement changes in the case that an unexpected event occurs. Moreover, high communication levels are necessary in order to perform complicated tasks, where several agents must be coordinated. We describe in this section the main concepts found in the literature related to optimizing communication time in ad hoc network systems.

One of the main difficulties concerning the maintenance of communication is an ad hoc network is determining the location of agents at a given moment in time. Several methods have been proposed for improving localization in this situation. Moore et al.[16], for example, presented a linear time algorithm for determining the location of nodes in an ad hoc network in the presence of noise. Other algorithms for the same problem have been suggested by Capkun et al.[7], Doherty et al.[13], and Priyantha et al.[19].

While such algorithms can be useful in determining the correct location of nodes, they are only able to provide information about current positions, and are not meant to optimize locations for a specific objective. Packet routing, on the other hand, has been previously studied with the goal of optimizing some common parameters, such as latency, cost of the resulting route, and energy consumed. For example, Butenko et al.[6] proposed a new algorithm for computing a backbone for wireless networks with minimum size, based on a number of related algorithms for this problem[5,8,15].

Another problem involving the minimization of an objective function over all feasible positions of agents in an ad hoc network is the so-called *location error minimization* problem. In the location error minimization problem, given a set of measurements of node positions (taken from different sources), the goal is to determine a set of locations for wireless nodes such that errors in the given measurements are minimized. This problem has been formulated and solved using mathematical programming techniques, by the use of a relaxation for the general problem into a semi-definite programming model[20,3,4].

In this paper we consider a different optimization problem (CCPM) over the relative position of nodes in a wireless network. The CCPM has the objective of maximizing the *total communication time* of a given set of wireless agents. The problem has been proposed by considering military situations where a set of agents needs to accomplish a mission. It has been proved[17] that the problem is NP-hard, which makes clear the necessity of fast algorithms for the practical solutions of instances with large size. We present a method for constructing and optimizing solutions for the CCPM in the next sections.

4. Construction Heuristic

In this section, we propose a construction heuristic to create high quality solutions for the CCPM. Our objective is to provide a fast way of constructing a set of paths, connecting wireless agents from their initial positions S to the destinations D such that the resulting routes are feasible for the problem. The union of such sequences of nodes will uniquely determine the cost of the solution, which is calculated using equation (2). The algorithm also tries to create solutions that have as large a value as possible for the objective function.

The pseudo-code for the construction heuristic is showed in Figure 1. The algorithm starts initializing the cost of the solution to zero. The incumbent solution, represented by the variable `solution`, is initialized with the empty set.

The next step consists of finding shortest paths connecting each source $s_i \in S$ to a destination $d_i \in D$. Standard minimum cost flow algorithms can be used to calculate these shortest paths. For example, the Floyd-Warshall algorithm[14,22] can be used to compute the shortest path between all pairs of nodes in a graph. The Dijkstra algorithm[12] can also be used to perform this step of the algorithm (with the only difference that, being a single-source shortest path algorithm, it must be run for $|U|$ iterations, one for each of the $|U|$ source-destination pairs).

In the loop from lines 4 to 10, the algorithm performs the assignment of new paths to the solution, using the shortest path algorithm described above. First, a source-destination path s_i-d_i is selected, and based on this a shortest path $\mathcal{P}_i$ corresponding to this pair is generated. Notice that, if the length (number of edges) of the shortest path $\mathcal{P}_i$ is more than T there is not feasible solution for the problem, since the destinations cannot be reached at the end of the requested time horizon. The algorithm checks for this condition on line 6.

If all source-destination pairs are found to be feasible, then a solution is generated by the union of all $\mathcal{P}_i$. Notice that once agent i reaches node d_i it can simply loiter at d_i during all remaining time (until instant T), as shown in line 7. The sequence of nodes found as a result of this process is then added to the solution in line 8 of the algorithm, and the optimum objective value is updated (line 9). Finally, a complete solution is returned on line 11, along with the value of that solution.

```
procedure ShortestPath(solution)
1       c ← 0;
2       solution ← $\emptyset$;
3       Compute all shortest paths for each $(s_i, d_i)$ pair;
4       for i = 1 to $|U|$ do→
5          $\mathcal{P}_i$ ← SP($s_i, d_i$);
6          if length of $\mathcal{P}_i > T$ then return $\emptyset$;
7          let agent $i$ stay in $d_i$ until time $T$ is reached;
8          solution ← solution $\cup \mathcal{P}_i$;
9          c ← c + number of new connections generated by $\mathcal{P}_i$;
10      rof
11      return (c, solution);
end ShortestPath
```

Figure 1.: Pseudocode for the Shortest Path Constructor.

Theorem 4.1. *The construction algorithm presented above finds a feasible solution for the* CCPM *in $\mathcal{O}(|V|^3)$ time.*

Proof. A feasible solution for this problem is given by a sequence of positions starting at s_i and ending at d_i, for each agent $u_i \in U$. Clearly, the union of the shortest paths provide the required connection between each source-destination pair, according to the remarks in the preceding paragraph; therefore the solution is feasible. Suppose that, in line 3, we use the Floyd-Warshall algorithm for all-pairs shortest path[14,22]. This algorithm runs in $\mathcal{O}(|V|^3)$ time. Then, at each step of the `for` loop we need only to refer to the solution calculated by the Floyd-Warshall algorithm and add it to the variable `solution`. This can be done in time $\mathcal{O}(|V|)$, and therefore the `for` loop will run in at most $\mathcal{O}(|V||U|)$ time. Thus, the step with highest time complexity is the one appearing in line 3, which implies that the total complexity of the algorithm is $\mathcal{O}(|V|^3)$. □

5. Local Search Heuristic

A construction algorithm is a good starting point in the process of solving a combinatorial optimization problem. However, due to the NP-hardness of the CCPM, such an algorithm provides no guarantee that a good solution will be found. In fact, it is possible that for some instances the solution found by the construction heuristic is far from the optimum, and not even a local optimal solution.

To guarantee that the solution found is at least locally optimal, we propose a local search algorithm for the CCPM. A local search algorithm receives as input a feasible solution and, given a neighborhood structure for the problem, returns a solution that is guaranteed to be optimal with respect to that neighborhood.

For the CCPM, the neighborhood structure is defined as follows. Given an instance Π of the CCPM, let $\mathcal{S}$ be the set of feasible solutions for that instance. Then, if $s \in \mathcal{S}$ is feasible for Π, the neighborhood $\mathcal{N}(s)$ of s is the set of all solutions $s' \in \mathcal{S}$ that differ from s in exactly one route. Obviously, considering this neighborhood, there are $|U|$ positions where a new path could be inserted; moreover, the number of feasible paths between any source-destination pair is exponential.

Thus, in our algorithm, instead of exhaustively searching the entire neighborhood for each point, we probe only $|U|$ neighbors at each iteration (one for each source-destination pair). Also, because of the exponential size of the neighborhood, we limit the maximum number of iterations performed to a constant `MaxIter`.

We use randomization to select a new route, given a source destination pair. This is done in our proposed implementation using a modified version of the depth-first-search algorithm[11]. A *randomized depth-first-search* is identical to a depth-first-search algorithm, but at each step the node selected to explore is uniformly chosen among the available children of the current node. Using the randomized depth-first-search we are able to find a route that may improve the solution, while avoiding being trapped at a local optimum after only a few iterations.

A description of the local search procedure in form of pseudo-code is given in Figure 2. The algorithm used can be described as follows. Initially, the algorithm receives as input the basic feasible solution generated on phase 1 (the construction phase). A neighborhood for this solution is then defined to be the set of feasible solutions that differ from the current solution by one route, as previously described.

```
procedure HillClimb(solution)
1      Compute cost c of solution;
2      while solution not locally optimal and iter < MaxIter do→
3        for all agent pairs $(s_i, d_i)$ do→
4          Remove $\mathcal{P}_i$ from solution;
5          Find alternate feasible path $\mathcal{P}'_i$
              using the randomized DFS algorithm;
6          compute cost c' of new soluton
7          if new solution is feasible and c' > c then
8              c ← c';
9              iter ← 0;
11         else
12             Restore path $\mathcal{P}_i$;
10         fi
11       rof
12     iter ← iter + 1;
13     elihw
end HillClimb
```

Figure 2.: Pseudocode for the Hill Climbing intensification procedure.

Given the basic feasible solution obtained from the construction subroutine, the neighborhood is explored in the following manner. For each agent $u_i \in U$, we reroute the agent on an alternate feasible path from s_i to d_i (lines 3 to 13). Recall that a path $\mathcal{P}_i$ is feasible if the total length of this path is less than $\mathcal{L}_i$ and the agent reaches its target node by time T. This alternate path is created on line 5 using a modified depth-first-search algorithm[2]. The modification to the DFS is a randomization which selects the child node uniformly during each iteration. This procedure is efficient in that it can be implemented in polynomial time, as shown bellow.

Theorem 5.1. *The time complexity of the algorithm above is $\mathcal{O}(kTu^2m)$, where $k =$ `MaxIter`, T is the time horizon, $u = |U|$ and $m = |E|$.*

Proof. Notice the the most time consuming step of the algorithm is the construction of a new path (line 5). However, using a randomized depth-first-search procedure this can be done in $\mathcal{O}(m)$ time[2]. Each iteration of the `while` loop (lines 2 to 13) will perform local improvements in the solution using the re-routing procedure to improve the objective function. An upper bound on the best solution for an instance of this problem is $Tu(u-1)/2$

(the time horizon multiplied by maximum number of connections). Each improvement can require at most `MaxIter` iterations to be achieved. Therefore, in the worst case this heuristic will end after $\mathcal{O}(kTu^2m)$ time. $\square$

6. Combining Algorithms into a One-Pass Heuristic

The two algorithms described in Sections 4 and 5 can be combined into a single one-pass heuristic for the CCPM. The pseudo-code for the complete algorithm used can be seen in Figure 3. The new algorithm now behaves as a single-start, diversification and intensification heuristic for the CCPM.

The total time complexity of this heuristic can be determined from Theorems 1 and 2. Taking the maximum of the two time complexities determined previously, we have a total time of $O(\max\{n^3, kTu^2m\})$, where T is the time horizon, $u = |U|$, $n = |V|$, $m = |E|$, and $k =$ `MaxIter` is the maximum number of iterations allowed on the local search phase.

```
procedure OnePass(Instance)
1    Input: Instance of the CCPM, with $n$ nodes, $m$ edges, and a set $U$ of agents;
2    solution ← ConstructionHeuristic(Instance);
3    solution ← HillClimbHeuristic(solution);
4    return solution ;
end OnePass
```

Figure 3.: Pseudocode for the One-Pass Heuristic

7. Computational Results

The algorithm proposed above was tested to verify the quality of the solutions produced, as well as the efficiency of the resulting method. The test instances employed in the experiments were composed of 60 random unit graphs, distributed into groups of 20, each group having graphs with 50, 75, and 100 nodes. The communication radius of the wireless agents was allowed to vary from 20 to 50 units. This has provided us with a greater base for comparison, resulting in random graphs and wireless units that more closely resemble real-world instances.

The graphs used in the experiment were created with the algorithm proposed by Butenko et. al.[10,9] in the context of the BROADCAST SCHEDULING problem. The routines were coded in FORTRAN. Random numbers were generated using Schrage's algorithm[21]. In all experiments, the random number generator was started with the seed value 270001.

Instance	Nodes	Radius	Agents	OnePass	SP Soln	Agents	OnePass	SP Soln	Agents	OnePass	SP Soln
1	50	20	10	63.6	52.4	15	152	120.8	25	414.66	353.6
2	50	30	10	83.8	58.4	15	182.2	124.4	25	516.2	415.6
3	50	40	10	95.4	67.4	15	228.6	171.8	25	695	474.8
4	50	50	10	115.4	64.4	15	275.8	167.4	25	797.4	485.4
5	75	20	10	76.8	59	20	270.2	228.6	30	575.2	464
6	75	30	10	85.8	56	20	299.6	241.2	30	725.4	554
7	75	40	10	96.4	64.4	20	386	261	30	862.6	595.4
8	75	50	10	125	67.8	20	403.2	246.8	30	1082.4	670.8
9	100	20	15	113.6	100.4	25	333.4	269.4	50	1523.2	1258.8
10	100	30	15	166.2	124.4	25	511.2	365	50	1901.4	1515.8
11	100	40	15	203.4	141	25	600.6	389.8	50	2539.2	1749.4
12	100	50	15	255.8	151.8	25	756.8	479.6	50	3271.2	2050.6

Table 1.: Comparative results between shortest path solutions and heuristic solutions.

Results obtained in our preliminary experiments are reported in Table 1. In this table, the results of the one-pass algorithm (OnePass column) are compared to a simple routing scheme where only the construction phase is explored (the SP Soln column). The solutions shown in the table represent the average of the objective function values from the 5 instances in each class.

The numerical results provided in the table demonstrate the effectiveness of the proposed heuristic when the improvement phase is added to the procedure. The proposed heuristic increased the objective value of the shortest path solutions by an average of 38%. One reason for this is the fact that, when agents are routed solely according to a shortest path, they are not taking advantage of the remaining time they are allotted (i.e., the time horizon T) and the values from the distance limit given by $\mathcal{L}$.

Our heuristic, on the other hand, allows wireless agents to take full advantage of these bounds. The algorithm can do this by adjusting the paths to include those nodes within the range of other agents. In addition, at any given time an agent is allowed to loiter in its current position, possibly waiting for other agents to come into its range. This cannot occur in the phase 1 algorithm because, according to the shortest path routing protocol, loitering is forbidden.

We notice that that, in fact, our method provides solutions that are better than the shortest path protocol. The time spent on the algorithm has always been less than a few seconds, therefore the computational time is small enough for the problem sizes explored in our experiments. We believe, however, that the quality of the solutions and computational time can be further improved using a better implementation, and more sophisticated data structures to handle the information stored during the algorithm.

8. Conclusions and Future Research

In this paper we presented a heuristic approach to solve the *cooperative communication problem on ad hoc networks*. This problem, known to be NP-hard, is of importance in the planning of operations involving high levels of collaboration among team members. The proposed algorithm creates a high quality solution for the problem using two phases: 1) a construction heuristic, which uses shortest path algorithms to create a feasible solution, and 2) a local search algorithm, which improves the solution previously found in order to guarantee local optimality.

This paper reflects the current stage of our research in this problem. We plan to extend the algorithmic methods presented in this paper using more efficient optimization strategies. We will use the two phase algorithm described as the starting point for a greedy randomized procedure (GRASP metaheuristic). This will allow us to escape local minima inherent to the approach used in this paper, and find solutions closer to the desired global optimum.

Acknowledgments

The first and third authors have been partially supported by NSF and U.S. Air Force grants. The second author has been partially supported by a grant from the Measurement & Control Engineering Center/NSF.

References

1. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
2. R.K. Ahuja and T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.
3. P. Biswas and Y Ye. A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization. Technical report, Dept of Management Science and Engineering, Stanford University, 2003.

4. P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 46–54. ACM Press, 2004.
5. S. Butenko, X. Cheng, D.-Z. Du, and P. M. Pardalos. On the construction of virtual backbone for ad hoc wireless network. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Cooperative Control: Models, Applications and Algorithms*, pages 43–54. Kluwer Academic Publishers, 2002.
6. S.I. Butenko, X. Cheng, C.A.S. Oliveira, and P.M. Pardalos. A new algorithm for connected dominating sets in ad hoc networks. In S. Butenko, R. Murphey, and P. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, pages 61–73. Kluwer Academic Publishers, 2003.
7. S. Capkun, M. Hamdi, and J. Hubaux. Gps-free positioning in mobile ad-hoc networks. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9*, page 9008, Washington, DC, USA, 2001. IEEE Computer Society.
8. X. Cheng, X. Huang, D. Li, W. Wu, and D.Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.
9. C.W. Commander, S.I. Butenko, and P.M. Pardalos. On the performance of heuristics for broadcast scheduling. In D. Grundel, R. Murphey, and P. Pardalos, editors, *Theory and Algorithms for Cooperative Systems*, pages 63–80. World Scientific, 2004.
10. C.W. Commander, S.I. Butenko, P.M. Pardalos, and C.A.S. Oliveira. Reactive grasp with path relinking for the broadcast scheduling problem. In *Proceedings of the 40th Annual International Telemetry Conference*, pages 792–800, 2004.
11. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1992.
12. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.
13. L. Doherty, K. S. J. Pister, and Ghaoui L. E. Convex position estimation in wireless sensor networks. In *Proc. IEEE INFOCOM*, Anchorage, AK, 2001.
14. R.W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345, 1962.
15. M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, and D.J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.
16. David Moore, John Leonard, Daniela Rus, and Seth Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM Press.
17. C.A.S. Oliveira and P.M. Pardalos. An optimization approach for cooperative communication in ad hoc networks. Technical report, School of Industrial Engineering and Management, Oklahoma State University, 2005.
18. C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. Optimization problems in multicast tree construction. In *Handbook of Optimization in Telecommunications*, pages 701–733. Kluwer, Dordrecht, 2005.

19. N.B. Priyantha, H. Balakrishnan, E.D. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 172–183, 2005.
20. C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, 2002.
21. L. Schrage. A more portable FORTRAN random number generator. *ACM Transactions on Mathematical Software*, 5:132–138, 1979.
22. S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.