Simultaneous Placement and Assignment for Exploration in Mobile Backbone Networks

Emily M. Craparo, Jonathan P. How, and Eytan Modiano

Abstract— **This paper presents new algorithms for conducting cooperative sensing using a mobile backbone network. This hierarchical sensing approach combines backbone nodes, which have superior mobility and communication capability, with regular nodes, which are constrained in mobility and communication capability but which can sense the environment. In the framework of a cooperative exploration problem, a technique is developed for simultaneous placement and assignment of regular and mobile backbone nodes. This method, a generalization of existing techniques that only consider stationary regular nodes, optimally solves the simultaneous placement and assignment problem in computationally tractable time for problems of moderate size. For large-scale instances of this problem, a polynomial-time approximation algorithm is developed. This algorithm carries the benefit of a theoretical performance guarantee and also performs well in practice. Finally, the simultaneous placement and assignment technique is incorporated into a cooperative exploration algorithm, and its performance is shown to compare favorably with that of a benchmark based on existing assignment algorithms for mobile backbone networks.**

I. Introduction

The motivation for analyzing mobile backbone networks is discussed in detail in Refs. [6], [8], [9]. Srinivas et al. [8] define two types of nodes, which may be thought of as representing robotic agents: regular nodes, which have limited mobility and communication capability, and mobile backbone nodes, which have greater communication capability than regular nodes and which can be placed at arbitrary locations in order to provide communication support for the regular nodes. Each regular node is assigned to at most one mobile backbone node and communicates only with this node. The mobile backbone nodes, in turn, communicate with each other in order to provide end-to-end communication capability for the network. Thus, the goal of mobile backbone network optimization is to place these nodes and assign regular nodes to mobile backbone nodes such that the overall quality of the network is optimized. Quality is defined, for example, in terms of the number of regular nodes that achieve a desired minimum throughput, or the lowest throughput achieved by any regular node in the network.

This paper uses a model of communication described by Srinivas and Modiano [7]. In this model, the throughput (or data rate) that can be achieved for transmissions from a regular node to its assigned mobile backbone node is modeled as a decreasing function of both the distance between the two nodes and the number of other regular nodes that are also communicating with that particular mobile backbone node and thus causing interference. For example, the throughput τ between regular node i and mobile backbone node j when using a Slotted Aloha communication protocol can be approximated by [7]

$$\tau(i,j) \approx \frac{1}{e \cdot |A(j)| \cdot d(i,j)^{\alpha}} \tag{1}$$

where $|A(j)|$ represents the number of regular nodes assigned to mobile backbone node j, $d(i,j)$ represents the distance between regular node i and mobile backbone node j, α represents the path loss exponent, and e is the base of the natural logarithm. An implicit assumption is made that regular nodes assigned to one mobile backbone node encounter no interference from regular nodes assigned to other mobile backbone nodes (for example, because each "cluster" consisting of a mobile backbone node and its assigned regular nodes operates at a different frequency than other clusters).

A key insight noted in [7] and utilized in this work is that although the mobile backbone nodes can be placed anywhere in the plane, only a limited number of possible locations for mobile backbone nodes need to be considered in order to obtain an optimal solution to the problems described in the first paragraph of this section. Specifically, each mobile backbone node can be placed at the 1-center of its set of assigned regular nodes in an optimal solution. The 1-center location for a set of regular nodes is simply the location that minimizes the maximum distance to any regular node, and it is easily computable [1]. Fortunately, although there are $O(2^N)$ possible subsets of N regular nodes, there are only $O(N^3)$ distinct 1-center locations [5]. This means that although the mobile backbone nodes can theoretically occupy any of an infinite number of possible locations, only a polynomial number of locations need actually be considered, and these locations can be enumerated efficiently. Srinivas et al. take advantage of this fact to develop a search-based algorithm for solving the *maximum fair placement and assignment* problem (MFPA) in networks comprised of stationary regular nodes and mobile backbone nodes [7]. In the MFPA problem, optimality is defined in terms of the minimum throughput achieved by a regular node when all regular nodes are assigned to mobile backbone nodes. The running time of the algorithm presented in [7] is polynomial

E. Craparo is a Research Assistant in the Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, emilyc@mit.edu

J. How is a Professor in the Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, jhow@mit.edu

E. Modiano is an Associate Professor in Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, modiano@mit.edu

in the number of regular nodes but exponential in the number of mobile backbone nodes.

Our previous work has described an improved optimal solution technique for this problem, as well as for the related problem of maximizing the number of regular nodes that achieve a desired minimum throughput [2]. Our work also described the first known polynomial time approximation algorithm for maximizing the number of regular nodes that achieve a desired minimum throughput for the case of stationary regular nodes [2].

This paper formulates an extension to the mobile backbone network problem that allows modeling of regular node motion and derives exact and approximate solution techniques for this problem. These techniques are then applied to a cooperative exploration problem.

II. Joint Placement of Regular and Mobile Backbone Nodes

A. Previous work and problem statement

Previous problem formulations in mobile backbone networks have assumed that the locations of regular nodes are fixed *a priori* and that only the locations of mobile backbone nodes are variable [2] [7] [8]. This assumption is reasonable for some applications, such as scenarios that involve mobile agents extracting data from a fixed sensor network. However, in many applications the locations of both regular nodes and mobile backbone nodes can be controlled. For example, consider a cooperative exploration mission being executed by a heterogeneous team of air and ground vehicles. The ground vehicles can move and can accurately sense phenomena at ground level, but the air vehicles are more mobile and are better equipped to communicate over long distances.

This paper develops a modeling framework and solution technique that are appropriate for problems of this nature. We assume that L candidate regular node locations are available *a priori*, perhaps selected by heuristic means or due to logistical constraints. Each of N regular nodes ($N \leq L$) must occupy one of these locations, and no two regular nodes can be assigned to the same location. Given an initial location and a mobility constraint, each regular node is capable of reaching a subset of the other locations. There are K mobile backbone nodes ($K \leq N$) that can be placed anywhere, a throughput function τ is specified, and a desired minimum throughput τ_{min} is given.

Given these assumptions, the goal of this section is to *place* both the regular nodes and mobile backbone nodes while simultaneously *assigning* regular nodes to mobile backbone nodes in order to maximize the number of regular nodes that are successfully assigned and achieve the desired minimum throughput level τ_{min}, under the given throughput function τ.

B. Network design formulation

Optimal simultaneous placement and assignment of regular nodes and mobile backbone nodes is achieved through the solution of a *network design* problem. In network design problems, a given network (represented by a directed graph)

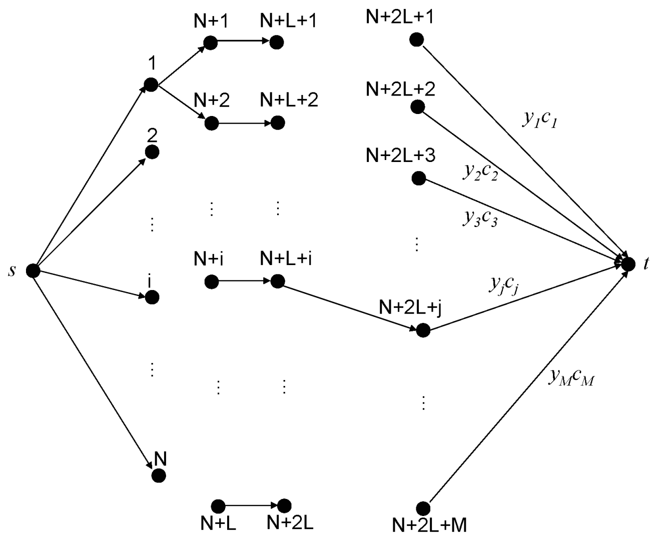


Fig. 1. The network design problem corresponding to the joint placement and assignment problem for mobile backbone networks, with regular node mobility. Unlabeled arc capacities are equal to one. For clarity, not all arcs and nodes are shown.

can be augmented with additional arcs for a given cost, and the goal is to optimize some desired flow characteristic at a minimum cost by intelligently "purchasing" a subset of these arcs, subject to a budget constraint. In this case, flow through the network is used to represent regular node movement as well as assignment of regular nodes to mobile backbone nodes, while arc "purchases" represent mobile backbone node placement.

The network graph over which this optimization takes place is schematically represented in Fig. 1. This graph is constructed as follows: the source node, s, is connected via an arc of unit capacity to each of a set of nodes N$= \{1, \ldots, N\}$, which represent the initial locations of the N regular nodes. Each of these nodes, in turn, is connected via an arc of unit capacity to a subset nodes in L$= \{N+1, \ldots, N+L\}$. Node i is connected to node $N+j$ iff regular node i can reach sensing location j under its mobility constraint. Next, each of the nodes in L is connected to a copy of itself in set L$'= \{N+L+1, \ldots, N+2L\}$, and again these arcs are of capacity one. This duplication is done in order to enforce the constraint that only one regular node can occupy each sensing location. The portion of the graph described thus far models regular node placement.

The remainder of the graph models mobile backbone node placement, as well as assignment of regular nodes to mobile backbone nodes. Each of the nodes in L$'$ is connected via an arc of unit capacity to a subset of the nodes in M$= \{N+2L+1, \ldots, N+2L+M\}$, which represent possible mobile backbone node locations. (Recall that, although mobile backbone nodes can be placed at arbitrary locations, only M locations need to be considered, where M is $O(L^3)$.) Node $N+L+i$ is connected to node $N+2L+j$ iff sensing location i is within the radius of the 1-center associated with j. (Recall that each 1-center location has both a center and a radius associated with it.)

Finally, each node in M is connected to the sink t. The capacity of the arc from node $N+2L+i$ to t is the product of a binary variable y_i, which represents the decision of whether to "purchase" this arc, and a constant c_i, which is the floor of the inverse with respect to cluster size of the throughput function, evaluated at the desired minimum throughput level, i.e. the maximum number of regular nodes that can be assigned to a mobile backbone node at location i and achieve the desired throughput level. For example, for the approximate Slotted Aloha throughput function described by Eq. 1,

$$c_i = \left\lfloor \frac{1}{e \cdot \tau_{min} \cdot r_i^{\alpha}} \right\rfloor$$

where τ_{min} is the desired minimum throughput and r_i is the radius associated with 1-center location i.

Note that any feasible solution to this network design problem represents a feasible placement and assignment of regular nodes and mobile backbone nodes (recall that all flows in an optimal solution to an integer network flow problem are integer); likewise, any feasible placement and assignment of regular nodes and mobile backbone nodes also determines a feasible flow in the graph. Therefore, an optimal solution to this network design problem yields an optimal solution to the simultaneous placement and assignment problem.

Denote the set of nodes in the network design graph by $\mathscr{N}$ and the set of arcs by $\mathscr{A}$. If K mobile backbone nodes are available and a minimum throughput level is specified, the goal of the network design problem is to select K arcs from $\{N+2L+1,\dots,N+2L+M\}$ to t and a feasible flow x_{ij}, $(i,j) \in \mathscr{A}$ such that the $s-t$ flow is maximized. This problem can be solved via the following mixed-integer linear program (MILP):

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{N} x_{si} \tag{2a}$$

$$\text{subject to } \sum_{i=1}^{M} y_i \le K \tag{2b}$$

$$\sum_{j:(i,j)\in\mathscr{A}} x_{ij} = \sum_{l:(l,i)\in\mathscr{A}} x_{li} \qquad i \in \mathscr{N} \setminus \{s,t\} \tag{2c}$$

$$x_{ij} \ge 0 \qquad \forall\ (i,j) \in \mathscr{A} \tag{2d}$$

$$x_{ij} \le 1 \qquad \forall\ (i,j) \in \mathscr{A} : j \in \mathscr{N} \setminus \{t\} \tag{2e}$$

$$x_{(N+2L+i)t} \le y_i c_i, \qquad i \in \{1,\dots,M\} \tag{2f}$$

$$y_i \in \{0,1\} \qquad i \in \{1,\dots,M\} \tag{2g}$$

where the constraints state that at most K arcs (mobile backbone node locations) can be selected (2b), flow through all internal nodes must be conserved (2c), arc capacities must be observed (2d- 2f), and y_i is binary for all i (2g).

Fig. 2 shows an example of a solution to the simultaneous placement and assignment problem with regular node movement. The regular nodes, initially in positions indicated by ●, are able to move to other locations (○) within their radii of motion, indicated by shaded pink circles. This initial configuration is shown in Fig. 2(a). In an optimal solution to this problem, shown in Fig. 2(b), the regular nodes have moved such that they are grouped into compact clusters for which the mobile backbone nodes can provide an effective communication infrastructure. The clusters are relatively balanced, in that the clusters with larger radii tend to have fewer regular nodes, while the more compact clusters can accommodate more regular nodes and still achieve the desired minimum throughput. In this example, all regular nodes have been successfully assigned to mobile backbone nodes.

TABLE I

AVERAGE COMPUTATION TIMES FOR VARIOUS VALUES OF N, K AND L, FOR THE MFPA PROBLEM.

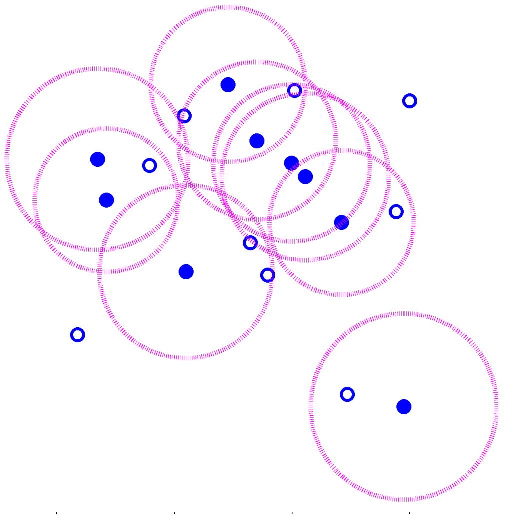
N	K	L	MILP Algorithm with Regular Node Movement
4	2	10	10 sec
6	2	15	16.5 sec
8	3	20	80 sec

We make the following remarks about this algorithm:

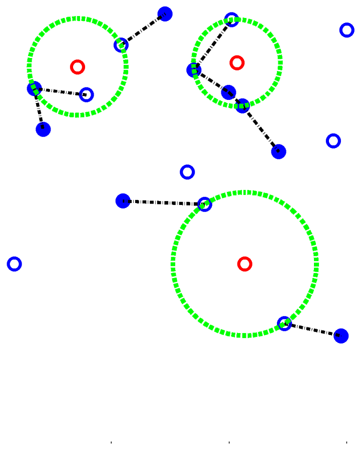
Remark 1: This algorithm is designed maximize the number of regular nodes that are assigned at throughput level τ_{min}. If, instead, the goal is to achieve the best possible minimum throughput such that *all* regular nodes are assigned to a mobile backbone node (i.e. to solve the MFPA problem), it is necessary to solve the MILP problem in Eq. 2 $O(\log(NL^3))$ times for different throughput values (which result in different values for the c_i's in the network design problem).

Remark 2: If arbitrarily many mobile backbone nodes are available and the goal is to achieve a desired minimum throughput while utilizing a minimal number of mobile backbone nodes, then a MILP problem similar to the one in Eq. 2 needs only to be solved once for the values of c_i corresponding to the desired throughput. The problem must be modified so that the number of mobile backbone nodes used is minimized, subject to the constraint that the flow through the graph is equal to the number of regular nodes.

Remark 3: It should be noted that the worst-case complexity of mixed-integer linear programming is exponential in the number of binary variables. However, this approach performs well in practice, and simulation results indicate that it compares very favorably with the search-based approach developed in Ref. [7] (see [2]). Table I shows the computation time of the MILP algorithm when applied to the MFPA problem described in Remark 1. Note that this problem requires repeated solution of the MILP; for problems that do not require repeated solution of the MILP, the algorithm is therefore faster. As the table indicates, this method is appropriate for problems of moderate scale.



(a) Initial regular node placement, with radius of motion for each regular node.



(b) An optimal placement of regular and mobile backbone nodes.

Fig. 2. A small example of mobile backbone network optimization with limited regular node movement. Open blue circles represent possible regular node locations, and filled blue circles are the initial locations of the regular nodes. Shaded pink circles in the left figure indicate the possible radius of motion of each regular node. In the right figure, mobile backbone nodes, shown in red, are placed such that they provide communication support for the regular nodes. Each regular node is assigned to at most one mobile backbone node. Dotted lines indicate regular node motion in this optimal solution. Dashed circles indicate the radius of each cluster of nodes. In this example, all regular nodes have been successfully assigned to mobile backbone nodes.

III. Approximation Algorithm

While the MILP-based algorithm described in the previous section is computationally feasible for problems of moderate scale, its worst-case computation time is exponential in the number of binary variables. Therefore, this section develops an approximation algorithm for this problem that is appropriate for problems of larger scale.

This approximation algorithm is based on the insight that the number of regular nodes that can be placed and assigned is a *submodular* function of the set of mobile backbone node locations that are selected. The submodularity condition for a set function f is typically stated as

$$f(S \cup \{i,j\}) - f(S \cup \{i\}) \leq f(S \cup \{j\}) - f(S)$$

where S is a set, and i and j are individual elements of the ground set such that $i,j \notin S$, $i \neq j$.

Submodular functions in discrete optimization are analogous to convex functions in continuous optimization [3]. Both can be efficiently minimized; however, maximization is more difficult. Fortunately, it has been shown that for maximization of a nondecreasing submodular set function f, where $f(\emptyset) = 0$, greedy selection of elements carries a performance guarantee of $1-(1-\frac{1}{R})^R > 1-\frac{1}{e}$, where R is the number of elements to be selected and e is the base of the natural logarithm [4]. This means that if an exact algorithm selects R elements from the ground set and produces a solution of value OPT, a greedy selection of R elements (i.e. selection via a process in which element i is selected if it is the element that maximizes $f(S \cup i)$, where S is the set of elements already selected) produces a solution of value at least $(1-(1-\frac{1}{R})^R) \cdot OPT$.

This observation motivates consideration of a greedy algorithm (Algorithm 1) for the problem of maximizing the number of regular nodes that achieve throughput level τ_{min}. Given a network design graph G, K mobile backbone nodes and M possible mobile backbone node locations, and denoting by f the maximum flow through G as a function of the set of mobile backbone node locations selected, this greedy algorithm is:

Algorithm 1

```
$S \leftarrow \emptyset$
$maxflow \leftarrow 0$
for $k$=1 to $K$ do
    for $m$=1 to $M$ do
        if $f(S \cup \{m\}) \geq maxflow$ then
            $maxflow \leftarrow f(S \cup \{m\})$
            $m^* \leftarrow m$
        end if
    end for
    $S \leftarrow S \cup \{m^*\}$
end for
return $S$
```

The following theorem describes the performance of Algorithm 1:

Theorem 1: Algorithm 1 returns a solution S such that $f(S) \geq \lceil (1-\frac{1}{e}) \cdot f(S^*) \rceil$, where S^* is the optimal solution to the network design problem on G.

Proof: This follows from the observation that all maximum flows through G are integer, and from the following Lemma, the proof of which appears in the Appendix:

Lemma 1: The maximum flow that can be routed through G is a submodular function of S, the set of arcs that are selected.

∎

Thus, Algorithm 1 is an approximation algorithm with approximation guarantee $1 - \frac{1}{e}$. Additionally, because each round of greedy selection consists of solving a polynomial number of maximum flow problems, and there are K rounds of selection, the running time of Algorithm 1 is polynomial in the number of regular nodes, the number of locations, and the number of mobile backbone nodes.

IV. Application to Cooperative Exploration

This section applies the techniques developed in the previous sections to a cooperative exploration problem. Consider a situation in which a set of L locations are to be visited and sensed by regular nodes, and the sensor data taken by the regular nodes is to be transmitted to the mobile backbone nodes. A location is successfully visited at time t if the following conditions are met:

- The location is occupied by a regular node n_i at time t.
- Regular node n_i is assigned to a mobile backbone node at time t.

Once a location has been visited, it remains visited for all future time. Our goal is to minimize the time required to visit all locations.

This problem can be written as a MILP; however, even for small numbers of locations and regular nodes, the problem rapidly becomes computationally intractable. Therefore, we turn our attention to heuristic and approximate algorithms.

First, consider a greedy algorithm (Algorithm A) based on a slight modification of the MILP technique described in Section II. At each time step, the algorithm positions both regular nodes and mobile backbone nodes in order to maximize the number of unvisited locations that are visited. This is accomplished using the MILP described in Section II. In the case that no regular node is able to reach an unvisited location in a particular iteration, a simple greedy algorithm may become "stuck" and make no further progress because no regular node has any incentive to move. Therefore, if a subset of the regular nodes is unable to reach any unvisited locations, they simply move to the locations that minimize the sum of their distances to the remaining unvisited locations. This modification guarantees that all locations will be visited in finite time.

For comparison, a second greedy algorithm is also considered. This algorithm (Algorithm B) is based on existing techniques that cannot accommodate regular node motion. In this algorithm, regular nodes are greedily positioned on unvisited locations, and mobile backbone nodes are then optimally placed in order to service the regular nodes occupying unvisited locations. Again, regular nodes that cannot reach unvisited locations are moved to the locations that minimize the sum of their distances to the remaining unvisited locations.

The key difference between these two algorithms is that Algorithm A optimizes over both the *placement* of regular and mobile backbone nodes as well as the *assignment* of regular nodes simultaneously, while Algorithm B must treat regular node placement and assignment sequentially, resulting in degraded performance.

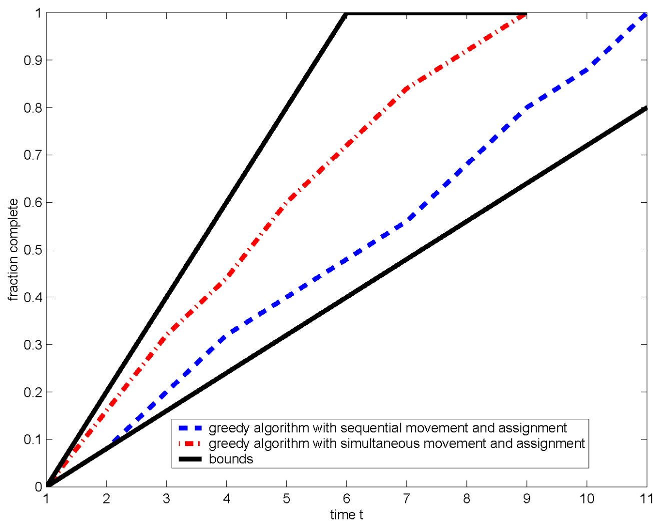


Fig. 3. Performance of two greedy algorithms (Algorithm A and Algorithm B) for the exploration problem, in terms of the fraction of locations visited as a function of time.

A. Experimental Performance Analysis

Fig. 3 illustrates the typical performance of these greedy algorithms on a particular example problem. A set of 25 locations were randomly generated in a finite 2-dimensional space according to a uniform distribution, and five regular nodes were randomly assigned to initial locations. Two mobile backbone nodes were available to collect data from the regular nodes. The red (dash-dot) line represents the percentage of the locations that were visited as a function of time when Algorithm A was used. The blue (dashed) line represents the same quantity when Algorithm B was used.

The upper black line in Fig. 3 is a theoretical upper bound on performance: this line depicts the fraction of locations visited if every regular node is successfully placed at an unvisited location and assigned to a mobile backbone node at every time step. In many cases this level of performance and is not achievable by any algorithm; this upper bound is considered due to the intractability of solving the problem to optimality. The lower black line represents the level of performance if every mobile backbone node covers only one regular node on an unvisited location at each time step. This is a strict lower bound on performance if the regular nodes are unconstrained in their movement (i.e. a regular node can reach any location from any other location in a single time step); otherwise, it is not a strict lower bound, but it is an interesting point of comparison by which to judge algorithms.

As shown in Fig. 3, simultaneous placement and assignment of regular nodes and mobile backbone nodes tends to significantly outperform sequential placement of these nodes in terms of total time required to visit all locations, as well as in the percentage of locations that have been visited at times prior to the completion time.

To verify that these trends hold over many problem instances, the performance of the two greedy algorithms was

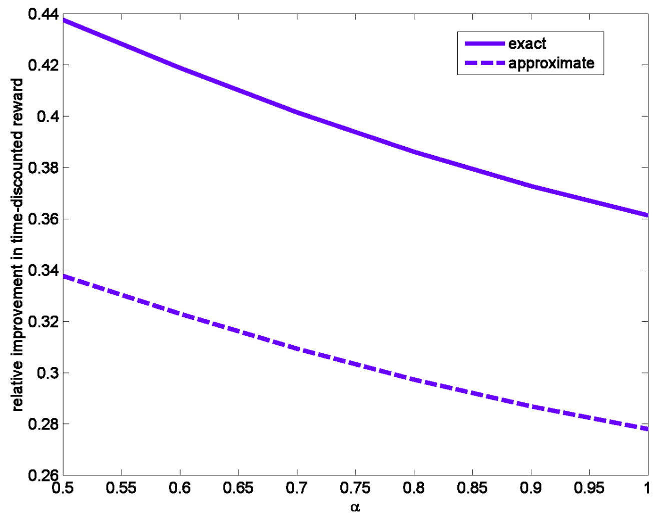


Fig. 4. Average improvement in time-discounted reward of the exact and approximate versions of Algorithm A, relative to the time-discounted reward of Algorithm B.

examined for 100 randomly-generated sets of initial conditions. On average, Algorithm A significantly outperformed Algorithm B, both in terms of total time to visit all locations and in terms of the percentage of locations that were visited at any particular time. At the theoretical minimum time at which exploration might have been completed by an exact algorithm ($t = \lceil \frac{L}{N} \rceil$), Algorithm A had visited an average of 72% of the locations, while Algorithm B had only visited 57% of the locations. An approximate version of Algorithm A in which the MILP optimization was replaced with the polynomial-time approximation algorithm developed in Section III had visited 67% of the locations.

It is also of interest to examine the time-discounted performance of both algorithms, since information gathered from uncertain environments is generally more useful when it is received earlier rather than later. The average time-discounted reward earned by both greedy algorithms was calculated for the randomly-generated instances described in the previous paragraph, where the reward at time t is simply the total number of locations that have been visited at time t, discounted by a factor of α^t, where $\alpha \leq 1$. Fig. 4 shows the relative improvement in total discounted reward obtained by the exact and approximate versions of Algorithm A over Algorithm B, evaluated at $t = \lceil \frac{L}{N} \rceil$, for various values of α. As the graph indicates, Algorithm A achieved a discounted reward that was $35-45\%$ greater than that of Algorithm B for values of $\alpha \geq 0.5$, and the approximate version of Algorithm A achieved a discounted reward $25-35\%$ greater than that of Algorithm B.

B. Theoretical Performance Analysis

As described previously, there are two complicating aspects of the exploration problem under communication constraints. One is the issue of motion planning, which is a difficult problem even when communication constraints are neglected. The other is the impact of communication constraints, as considered in this paper. To isolate the effect of communication constraints on the efficiency of exploration, we assume for purposes of analysis that the regular nodes are unrestricted in their movement, i.e. a regular node can reach any location from any other location in a single time step. In this case, a trivial upper bound on the time required to visit all locations is $T \leq \lceil \frac{L}{K} \rceil$, but a tighter upper bound can be found.

First, note that the total number of locations visited is a submodular function of the set of configurations of regular nodes and mobile backbone nodes that have been realized, where a configuration of regular nodes and mobile backbone nodes includes both their locations and the assignment of regular nodes to mobile backbone nodes. This is easy to see: if a measurement is taken from configuration j, this measurement cannot increase the total number of locations visited by a greater quantity if measurements have already been taken from configurations $S \cup \{i\}$ than if measurements have been taken from configurations S, since the measurement taken from configuration j may involve locations that are also measured in configuration i.

Using this insight, one can derive a performance bound on the time required to explore all locations using a greedy approach such as Algorithm A. Let T^* denote the time required to visit all locations using an exact algorithm. Because of the submodularity property, at time $t = T^*$, a greedy algorithm will have visited at least $\lceil (1-(1-\frac{1}{T^*})^{T^*})L \rceil \leq \lceil (1-\frac{1}{e})L \rceil$ locations. Furthermore, at each time $t > T^*$, the greedy algorithm can visit at least K new locations (assuming that K locations remain to be visited). So, the time required to visit the remaining locations is at most

$$\begin{aligned} \left\lceil \frac{L - \lceil (1-\frac{1}{e})L \rceil}{K} \right\rceil &= \left\lceil \frac{L + \lfloor (\frac{1}{e}-1)L \rfloor}{K} \right\rceil \\ &= \left\lceil \frac{\lfloor \frac{L}{e} \rfloor}{K} \right\rceil. \end{aligned}$$

This yields an overall bound for the time T required to visit all locations of

$$T \leq \min \left\{ \left\lceil \frac{L}{K} \right\rceil, \; T^* + \left\lceil \frac{\lfloor \frac{L}{e} \rfloor}{K} \right\rceil \right\},$$

which means that although even an exact algorithm may take up to $\lceil \frac{L}{K} \rceil$ time steps to completely explore all locations, a greedy algorithm is guaranteed to take no longer than $\left\lceil \frac{\lfloor \frac{L}{e} \rfloor}{K} \right\rceil$ more time steps than an exact algorithm, up to a maximum of $\lceil \frac{L}{K} \rceil$ total time steps.

V. Conclusions

This paper presented a generalization of existing mobile backbone network problems that models the motion of both regular nodes and mobile backbone nodes. A MILP-based exact solution to this problem was shown to perform well in practice for problems of moderate size, and a polynomial-time approximation algorithm was given for larger problems.

The performance of greedy algorithms based these techniques was given for a cooperative exploration problem, and a performance bound was established for a special case of this exploration problem. Given the intractability of solving the exploration problem exactly, our results indicate that a greedy approach is an excellent alternative.

VI. Acknowledgments

This work was funded in part by NSF grant CCR-0325401, AFOSR grant FA9550-04-1-0458, and an NSF Graduate Fellowship.

References

[1] P. Agarwal and M. Sharir, "Efficient Algorithms for Geometric Optimization," ACM Comput. Surveys, 30, pp. 412-458, 1998.

[2] E. Craparo, J. How and E. Modiano, "Optimization of Mobile Backbone Networks: Improved Algorithms and Approximation," *Proceedings of the American Control Conference*, June 2008.

[3] s. Fujishige, *Submodular Functions and Optimization (Second Edition)*, Annals of Discrete Mathematics, Vol. 58, 2005.

[4] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, 1978, pp 265-294.

[5] F. Preparta and M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[6] I. Rubin, A. Behzadm R. Zhang, H. Luo, and E. Caballero, "TBONE: a Mobile-Backbone Protocol for Ad Hoc Wireless Networks," *Proc. IEEE Aerospace Conference*, 6, 2002.

[7] A. Srinivas and E. Modiano, "Joint node placement and assignment for throughput optimization in mobile backbone networks," To appear in *Proc. IEEE INFOCOM'08*, Apr. 2008.

[8] A. Srinivas, G. Zussmanm and E. Modiano, "Mobile Backbone Networks: Construction and Maintenance," *ACM MOBIHOC 2006*, May 2006.

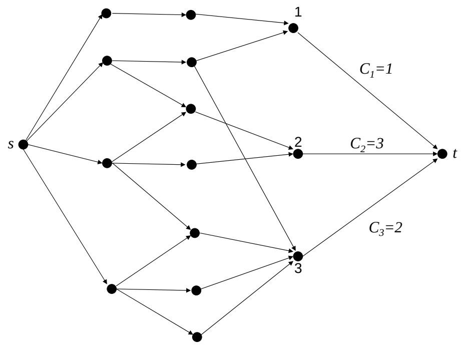
[9] K. Xu, X. Hong, and M. Gerla, "Landmark Routing in Ad Hoc Networks with Mobile Backbones," *Journal of Parallel and Distributed Computing*, 63, 2, pp 110-122, 2003.

Appendix

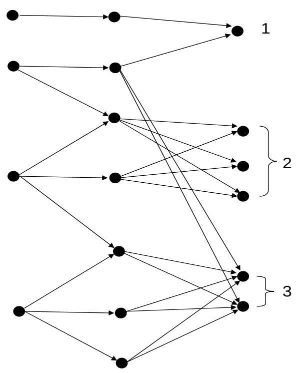
This section gives a proof of Lemma 1: The maximum flow that can be routed through G is a submodular function of S, the set of arcs that are selected.

For purposes of proving the submodularity of the network design objective function, the maximum flow problem of Section II will be reformulated as a set-to-set node disjoint path problem in a modified version of the maximum flow graph. A set-to-set node disjoint path problem specification consists of a directed graph, H, and designations of subsets of the nodes of H as the source set and the destination set. The goal of a set-to-set node disjoint path problem is to find the maximum number of paths originating in the source set and terminating in the destination set, such that no node in H is traversed by more than one path.

The modification of the maximum flow graph to the graph induced by the corresponding set-to-set node disjoint path problem is accomplished as follows: the s and t nodes are removed, and node set N remains unchanged. Node sets L and L$'$ are compressed into a single set L; since the problem under consideration is a node disjoint path problem, there is no need to enforce the node capacity constraint using a duplicate set of location nodes, as in the maximum flow problem. Set M is modified in the following way: if a node $m \in$M in the maximum flow problem has outgoing capacity c, then node set M in the modified graph itself contains a *set* of nodes m consisting of c copies of this node, each of which is connected to the same nodes in L as the original node m. An example of this reformulation is shown in Fig. 5.



(a) Graph induced by a maximum flow problem. For clarity, node sets L and L$'$ present in Fig. 1 have been replaced with a single node set, with the restriction that at most one unit of flow may traverse each of these nodes.



(b) Graph induced by a set-to-set node disjoint path problem.

Fig. 5. An example of conversion from a maximum flow problem to an equivalent set-to-set node disjoint path problem.

The source set in this problem is N, and the destination set is M. Note that any configuration of set-to-set node disjoint paths in this modified graph has a corresponding feasible flow in the maximum flow problem. Likewise, any feasible flow in the maximum flow problem defines a set of node disjoint paths in the modified problem. Therefore, the maximum flow in the original problem is equal to the maximum number of node disjoint paths in the modified problem.

To show that the maximum flow through G is a submodular function of the set of arcs that are selected, we will prove that the maximum number of node disjoint paths in H is a submodular function of the set of destination nodes. A restatement of the submodularity condition is:

$$f(S\cup\{i,j\})+f(S)\leq f(S\cup\{i\})+f(S\cup\{j\}).$$

The relevant maximum flow graphs for this relation are shown at the top of Fig. 6: the sum of the maximum flows through the left two graphs must be less than or equal to the sum of the maximum flows through the right two graphs.

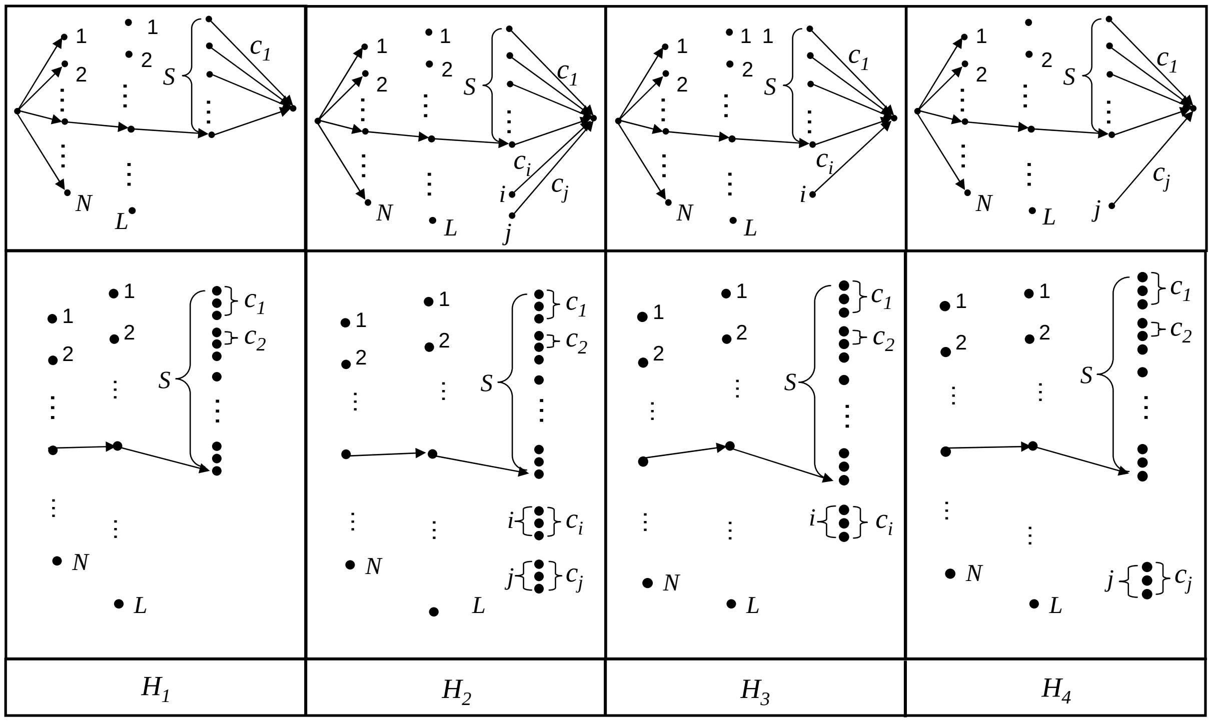


Fig. 6. Schematic representation of the graphs involved in the proof of Lemma 1. The top four graphs are for the original maximum flow problem, while the bottom four graphs are their equivalent reformulations in the node disjoint path problem. For clarity, not all arcs are shown.

Converting these maximum flow problems into their equivalent node disjoint path problems yields the graphs shown at the bottom of Fig. 6. The submodularity condition states that the maximum number of node disjoint paths in the left two graphs is at most the maximum number of node disjoint paths in the right two graphs. Denote these graphs from left to right by H_1, H_2, H_3 and H_4.

We will also make use of the following fact, which we state without proof: If H is a three-layered graph of the form shown in Fig. 5(b), with source set N and destination set M, and P is a set of node disjoint paths in H that covers a subset D of the destination nodes, there exists a maximum set of node disjoint paths P' in H that also covers node set D.

Lemma 1 can now be proved.

Proof: Making use of the reformulation of the maximum flow problem as a node disjoint path problem, the claim of the lemma can be restated as follows: if P_i denotes a maximum set of node disjoint paths in graph H_i from Fig. 6 for $i = 1,\ldots,4$, and $|P_i|$ denotes the cardinality of P_i (i.e. the number of elements from the source or destination sets covered by P_i), then $|P_1| + |P_2| \leq |P_3| + |P_4|$.

Consider a maximum set of node disjoint paths P_1 in graph H_1, and denote its cardinality by N_s. Note that P_1 is a feasible set of node disjoint paths for graph H_2 as well.

Because P_1 is feasible in graph H_2, there is a maximum set of node disjoint paths in H_2 that covers the same set of destination nodes in S as H_1. Call this optimal solution P_2. Denote the number of nodes covered by P_2 in node sets i and j by N_i and N_j, respectively. Then, the total number of node disjoint paths in P_1 and P_2 is equal to $2N_s + N_i + N_j$.

Now consider the set of node disjoint paths obtained by removing the paths ending in node set j from P_2. Note that this set of node disjoint paths is feasible for graph H_3, and its cardinality is $N_s + N_i$. Likewise, the set of node disjoint paths obtained by removing the paths ending in node set i from P_2 is feasible for graph H_4, and its cardinality is $N_s + N_j$. Since these sets of node disjoint paths are feasible (but not necessarily optimal) for H_3 and H_4, the sum of the cardinalities of maximum node disjoint paths for these graphs must be at least $2N_s + N_i + N_j$.

This establishes the submodularity property for the node disjoint path problem under consideration, and by extension for the maximum flow problem. ■