2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Reconfiguration Algorithms for Mobile Robotic Networks

Nilanjan Chakraborty
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA-15213
Email: nilanjan@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA-15213
Email: katia@cs.cmu.edu

***Abstract*—For a deployed mobile robotic network to function usefully, the robots should have the capability to adjust their positions, while maintaining the network connectivity. In this paper, we present algorithms that allows a robot to decide when it is feasible for it to move to a desired point by adjusting its own positions (and the positions of some other robots in the network), while maintaining all the network connectivity constraints. Under the assumption of a disc model of communication, we show that the problem can be formulated as a convex optimization (or feasibility) problem (actually a second order cone program). Thus, the problem can be solved in polynomial time by centralized interior point algorithms. However, this requires the robot to have knowledge of the position of all the nodes in the network. Our main contribution is the development of an incremental algorithm, that solves the feasibility problem (of whether the robot can move to its desired goal) by obtaining the information about the position of the robots and their immediate neighbors only if they are required to move. We present simulation results comparing the performance of the centralized algorithm with the incremental algorithm for randomly generated networks. From simulation results, we observe that the time required by the incremental algorithm to solve the feasibility problem is relatively independent of the size of the network.**

***Index Terms*—robotic networks, network reconfiguration, second order cone program (socp), convex optimization.**

I. Introduction

Mobile robotic networks with communication capabilities have received wide attention recently because of their potential applications in environmental monitoring, search and rescue operations, extraterrestrial exploration. In a deployed mobile robotic network, it is desirable for the robots to have the capability of adjusting their positions according to the demands of the situation. It is also necessary to ensure that the communication network formed by the robots stay connected after the adjustments have been made. An example scenario is presented in Figure 1 where the robots, in order to be robust to any communication link failure, wants to form a 2-connected network (i.e., at least two edges in the graph need to be removed for it to become disconnected). One way to do this is to move the robot 1 within the communication range of robot 10 without breaking any of the communication edges already present in the network.

We note that our problem is different from the formation control problem of moving the whole robotic network while maintaining connectivity (as studied in [1], [2]). The distinction is in terms of the scale of the distance to be moved. To get an intuitive understanding of the distinction, let us consider the convex hull of all the robots positions and let the diameter of the convex hull polytope be its characteristic length. In our problem, the distance to be moved by a robot is much smaller than the characteristic length of the convex hull while in [1], [2] it is much larger. Thus, we call the problems that we are studying network reconfiguration (or repositioning) problems.

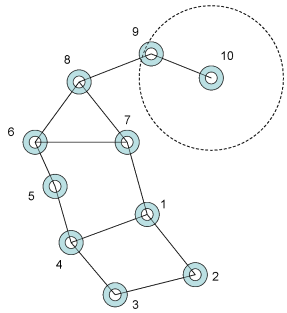

Fig. 1. An example network of 10 robots. The robot 1 wants to move within the communication range of robot 10 to form a 2-connected network.

In this paper, we consider two related abstract network reconfiguration problems: (a) design algorithms to decide whether it is possible to reconfigure the network (i.e., adjust the position of the robots while satisfying the constraints of each robot) to bring a specified robot within a given distance of a known point, (b) design algorithms to decide whether it is possible to reconfigure the network to bring at least one robot within a given distance of a known point. Note that if we can solve the problem (a), we can solve the problem (b) by repeating the algorithm for solving (a) for every robot in the network. Thus, the basic problem that we address in this paper is problem (a) and it can be written as follows: *Let* $P = \{\mathbf{q}_j = (x_j, y_j)\}, j = 1, 2, \ldots, n$, *be the set of positions of* n *mobile robots (or nodes). Let* C_j *be a convex set specifying the communication area of robot* j *and* S_j *be a convex set specifying the processing*[1] *area of robot* j. *Let* $P' \subset P$ *be the set of robots that has to stay within the*

[1]We use the term processing to denote any generic operation that may be physically interactive like sample collection or non-physical like sensing.

processing area of their current positions. Is it possible for the robots to adjust their positions (while maintaining all the previous network constraints) such that a given point $\mathbf{q}_0$ falls within the processing zone of a given robot i? We assume the communication area and processing area of each robot to be discs with possibly different radii.

We show that the above problem is equivalent to checking the feasibility of a second order cone program (SOCP) and can thus be solved in polynomial time by a centralized interior point algorithm. Moreover, within the same framework, we can also optimize the total distance moved by the robots (so that the energy expended in repositioning can be minimized). For the centralized algorithm, the robot i (or the centralized processor) needs to know the coordinates of all the robots in the network. Thus, the centralized algorithm may be useful when (a) the number of robots in the network is small or (b) the architecture of the deployed mobile robotic system is such that there is a base station that keeps the updated information about the position of all the robots in the network. However, in general, the centralized algorithm is not scalable. Therefore, we present an incremental algorithm, that solves the feasibility problem by obtaining the information about the position of the robots and their immediate neighbors only if they are required to move. For network reconfiguration problems, usually only a small fraction of the total number of nodes need to be moved. Therefore, as verified by our simulation results on randomly generated networks, the computation time for obtaining the feasible solution is relatively independent of network size. This scalable incremental algorithm is our main contribution.

The paper is organized as follows. After a discussion of related work in Section II, we review the relevant mathematical background in Section III. We present the formulation of the network reconfiguration problems in Section IV and describe the centralized and incremental algorithms for solving the problems in Section V. We present our implementation results in Section VI, discuss the inclusion of obstacles in our framework in Section VII, and conclude with a discussion of future work in Section VIII.

II. Related Work

The network reconfiguration problem for improving sensing quality has been studied by Kansal et al. [3] in the context of a network of camera sensors. In their problem the cameras had limited tilt, pan and zoom capability and they wanted to reconfigure the camera network utilizing this mobility to improve the resolution or quality of the image obtained. However, since the cameras were statically located, there was no issue of loss of connectivity. An alternative body of recent literature concentrates on maintaining connectivity of a network of mobile robots while they are moving [4], [5], [1], [6]. In [4], [6], the authors present control laws for maintaining the connectivity of the network. The main goal of the control laws are to move the robots so as to maintain the connectivity and the objective of any robot having a desired goal is a secondary objective. In our work, the desired goal (if achievable) is the primary objective and we want to ensure that the connectivity constraints are maintained at the final positions of the robots. In that sense this work is complementary to the control theoretic approaches.

Convex optimization techniques have been used in the literature to solve network localization problems in sensor networks [7], [8] and shape formation problems [9], [10]. In [7] the authors formulated the sensor network localization problem as a semi-definite program that is solved using a centralized algorithm. Thereafter, much effort has gone into formulating the problem as a SOCP and developing distributed algorithms to solve the resulting SOCP (see [8] and references therein). The shape formation problem was proposed as a SOCP in [9] and a centralized solution was given to the problem. Thereafter, distributed algorithms were proposed to solve the problem [10]. Our problem is distinct from that in [9] because we do not know the positions the robots should move to, whereas, in the shape changing problem, the final shape, i.e., final coordinates of the robots are given.

III. Mathematical Preliminaries

Undirected graph: An undirected graph G is an ordered pair, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of n nodes, and $E \subseteq V \times V$ is a set of edges. Two nodes v_i and v_j are called *neighbors* of each other if $(v_i, v_j) \in E$. The set $\mathcal{N}_i = \{v_j | (v_i, v_j) \in E\}$ is the set of v_i's neighbors, and $|\mathcal{N}_i|$ is defined as the *degree* of node v_i.

Connected graph: A path between two nodes in a graph, $G = (V, E)$, is said to exist if there is a sequence of edges from E joining the two nodes. An undirected graph is called a connected graph, if there is a path between any two pairs of nodes in the graph.

Convex Set: A set $U \subseteq \mathcal{R}^n$ is called a convex set if for any two points $\mathbf{u}_1$, $\mathbf{u}_2 \in U$ and any λ with $0 \leq \lambda \leq 1$, we have

$$\lambda \mathbf{u}_1 + (1 - \lambda)\mathbf{u}_2 \in U.$$

Convex Function: A function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ is convex if the domain of f (*dom* f) is a convex set and for all $\mathbf{u}_1$, $\mathbf{u}_2 \in$ *dom* f and any λ with $0 \leq \lambda \leq 1$, we have

$$f(\lambda \mathbf{u}_1 + (1 - \lambda)\mathbf{u}_2) \leq \lambda f(\mathbf{u}_1) + (1 - \lambda) f(\mathbf{u}_2).$$

Second Order Cone: A set U is called a cone, if $\mathbf{u} \in U$ implies $\lambda \mathbf{u} \in U$ for all $\lambda > 0$. It is called a convex cone, if U is also a convex set. A second order cone is a convex cone defined by the Euclidean norm.

$$U = \{(\mathbf{u}, t) \in \mathbb{R}^{n+1} | \ \|\mathbf{u}\| <= t\} \tag{1}$$

Second Order Cone Program (SOCP): A second order cone program is a optimization problem where the objective function is linear and the constraints are either linear or second order cones. From the definition, it follows that a SOCP is a convex optimization problem.

IV. Network Reconfiguration Problems

The basic problem addressed in this paper can be formulated in graph theoretic terms. Let $G = (V, E)$ be the communication graph of the given set of robots, where

each node in V represents a robot and an edge $e \in E$ exists between two nodes (robots) if they are within the communication set of each other (for simplicity assume the communication model is a disc model with r_i being the communication radius of robot i). We assume that G is a connected graph, i.e., any two robots can communicate with each other. The basic problem that we are considering in this paper can be stated as follows:

Problem 1 ($P1$): Consider a set of robots in $\mathbb{R}^2$, whose communication graph[2], $G = (V, E)$, is a connected graph. Let $v_i \in V$ be a given robot with position $\mathbf{q}_i$, and Q (with coordinates $\mathbf{q}_0 \in \mathbb{R}^2$) be a fixed point such that the robot i needs to come within a distance ρ_i of Q. Also, let $V' \subset V$ be a set of nodes that has to be within the processing zone of their original position. Determine, if it is feasible for the robot i to get within ρ_i of point Q while maintaining all the previous network connections and satisfying the other processing constraints.

If the above problem is feasible, we want to find the robots that should be moved and their final positions. We may also want to minimize other objective functions like the total distance moved by the robots. Note that if the set V' is empty, there always exists a feasible solution to the problem. The trivial feasible solution is to translate each robot along the vector from $\mathbf{q}_i$ toward $\mathbf{q}_0$. The magnitude of translation is equal to the difference of their current distance and ρ_i. However, there may be other feasible solutions for which the distance traveled (and hence energy consumed) in reconfiguring may be much lesser, so finding the feasible solution such that the total distance traveled by the robots is minimized is desirable.

We can write the problem $P1$ as an optimization (or feasibility) problem and the resulting problem is a convex optimization problem. Let n be the total number of robots, i.e., $|V| = n$, and m be the total number of edges in the graph, i.e., $|E| = m$. Let $\mathcal{I}$ be the index set of the robots in the set V'. Let $\mathbf{q}_j \in \mathbb{R}^2$ be the coordinates of node v_j after repositioning and let $\tilde{\mathbf{q}}_j \in \mathbb{R}^2$ be the coordinates of the nodes before moving. The coordinates of each robot, $\tilde{\mathbf{q}}_j$, is known in a global reference frame, i.e., the robots are assumed to be localized and the coordinates of the point Q, i.e., $\mathbf{q}_0$, is known in the same global frame. Let $\mathbf{q}, \tilde{\mathbf{q}} \in \mathbb{R}^{2n}$ be the vectors formed by concatenating the vectors $\mathbf{q}_j$ and $\tilde{\mathbf{q}}_j$ respectively. The problem $P1$ can be then written as:

$$\begin{aligned} \min \quad & f_0(\mathbf{q}) \\ \text{s.t.} \quad & \|\mathbf{q}_j - \mathbf{q}_k\|^2 \leq d_{jk}, \ \forall (v_j, v_k) \in E \\ & \|\mathbf{q}_j - \tilde{\mathbf{q}}_j\|^2 \leq \rho_j, \quad j \in \mathcal{I} \\ & \|\mathbf{q}_i - \mathbf{q}_0\|^2 \leq \rho_i^2 \end{aligned} \tag{2}$$

where d_{jk} is the square of the maximum distance between node j and k for which connectivity can be maintained, and ${\rho_j}^3$, $j \in \mathcal{I}$ is the square of the processing radii each node in V'. The first set of constraints in Equation 2 represents the communication constraints and the second set of constraints represent the processing constraints of the robots. The last constraint represents that robot i should be within a distance of ρ_i of the given point $\mathbf{q}_0$. The objective function $f_0 : \mathbb{R}^{2n} \rightarrow \mathbb{R}$ is assumed to be a convex function. For the feasibility problem, the objective function is 0, and for the minimum motion problem, the objective function is

$$\sum_{i=1}^{n} \|\mathbf{q}_j - \tilde{\mathbf{q}}_j\|^2 \tag{3}$$

which is a convex function. It is easy to verify that the quadratic constraints in Equation 4 are convex quadratic functions. Since both the objective function and the constraints are convex quadratic functions the optimization problem in equation 2 is a convex optimization problem or a quadratically constrained quadratic program (that can be written as a SOCP).

We note that we can also consider other functions apart from minimizing the total distance traveled by the robots. An alternative objective function could be minimizing the maximum distance traveled by any robot. This function is also a convex function, hence this problem is also a convex optimization problem. In other words we can solve all variations of the problem $P1$ that keeps the problem as a convex optimization (or feasibility) problem. Since the problem in Equation 2 is a convex optimization problem the problem $P1$ can be solved in polynomial time using interior point methods.

As mentioned earlier, another problem of interest is reconfiguring the network such that a given point is within a certain distance of at least one of the robots in the network. This problem can be written more formally as:

Problem 2 ($P2$): Given a connected graph $G = (V, E)$, a set $V' \subset V$ in which the robots should remain within the processing radii around their original position, and a fixed point Q, is it possible to bring any robot in the network within a given distance of the point Q while maintaining the original connectivity of the network and satisfying the other proximity constraints.

As in the problem $P1$, here also we are interested in finding the robots to be moved, and their final positions, if the problem is feasible. An objective of minimizing the total distance or the maximum distance traveled by a robot can also be used. We can also write the problem $P2$ as an optimization problem as follows:

$$\begin{aligned} \min \quad & f_0(\mathbf{q}) \\ \text{s.t.} \quad & \|\mathbf{q}_j - \mathbf{q}_k\|^2 \leq d_{jk}, \ \forall (v_j, v_k) \in E \\ & \|\mathbf{q}_j - \tilde{\mathbf{q}}_j\|^2 \leq \rho_j, \quad j \in \mathcal{I} \\ & \min_{j=1,\ldots,n} \{\|\mathbf{q}_j - \mathbf{q}_0\|^2 - \rho_j\} \leq 0 \end{aligned} \tag{4}$$

where the notation is same as before. Note that the difference between the problem defined by Equation 2 and the problem defined by Equation 4 is in the last constraint. The last

[2]For clarity of exposition, we are implicitly assuming that the communication graph is undirected. All of the discussion in the paper goes through for directed graphs, in which case we assume G to be a strongly connected graph.

[3]ρ_j may also be 0; for example in a sensor network with static and mobile nodes, $\rho_j = 0$ for the static nodes.

constraint in Equation 4 is not convex and hence the problem $P2$ as written in Equation 4 is not a convex optimization problem. However, the problem in Equation 4 can be solved if we solve n versions of the problem in Equation 2, one for each robot, and take the solution in which the optimal value of the objective function is minimum. Since the problem in Equation 2 can be solved in polynomial time in the number of robots, the problem in Equation 4 can also be solved in polynomial time in n (an additional factor of n will be present in the expression of time complexity).

V. Solution Algorithms

In this section, we discuss the solution algorithms for the network reconfiguration problems described above. We will present our solutions in the context of problem $P1$. The solution of problem $P2$ involves repeated solution of problem $P1$. Since the problem given by Equation 2 is a convex optimization problem, we can solve it in polynomial time using a centralized interior point algorithm [11]. For implementation of the centralized algorithm, a network node needs to maintain and update the information about all other robots in the network via message passing. In addition, the computational cost also increases at least cubically with the number of robots; hence the centralized algorithms may not be scalable. Therefore, we present an incremental algorithm, which solves Equation 2 by solving a series of optimization problems, collecting information about positions of other robots in the network only when needed.

A. Centralized Solution

Theoretically, solving the optimization version of Equation 2 has the same computational complexity as solving the feasibility problem. Moreover the optimal solution using Equation 3 as the objective function is also a solution to the feasibility problem. Therefore, for the rest of this section we present the discussion in the context of the optimization version of Equation 2 with the objective function given by Equation 3.

We can rewrite Equation 2 as a SOCP as shown below:

$$\begin{aligned} \min \quad & \sum_{j=1,\ldots,n} s_j \\ \text{s.t.} \quad & \|\mathbf{q}_j - \mathbf{q}_k\|^2 \leq d_{jk}, \ \forall (v_j, v_k) \in E \\ & \|\mathbf{q}_j - \tilde{\mathbf{q}}_j\|^2 \leq s_j, \quad j = 1, \ldots, n \\ & \|\mathbf{q}_i - \mathbf{q}_0\|^2 \leq \rho_i^2 \\ & s_j \leq \rho_j, \quad j \in \mathcal{I} \end{aligned} \tag{5}$$

The objective function above is linear, the first three sets of constraints are cone constraints and the last set of constraints is linear; hence the optimization problem in Equation 5 is a second order cone program. The optimization variable is a $3n \times 1$ vector, $\mathbf{x} = [\mathbf{s}; \mathbf{q}]$, where $\mathbf{s}$ is a $n \times 1$ concatenated column vector of $s_j, j = 1, \ldots, n$. The number of constraints is $(1+m+n+|V'|)$. So, the theoretical complexity of solving this problem is $O(m+n)^{3.5}$ [11].

For a networked system of mobile robots the utility of the centralized algorithm depends on the architecture of the deployed system and the number of robots in the network. If the architecture of the deployed system is such that there is one robot that keeps the information about the positions of all the robots, then a centralized algorithm can be used for solving the problem. However, the centralized algorithm scales poorly with the size of the network.

B. Incremental Algorithm

In this section, we present the incremental algorithm. We will present our discussion in the context of the feasibility problem version of Equation 2. We first consider the related optimization problem of minimizing the distance between the robot i and the point Q subject to all the constraints. The problem can be written as

$$\begin{aligned} \min \quad & \|\mathbf{q}_i - \mathbf{q}_0\|^2 \\ \text{s.t.} \quad & \|\mathbf{q}_j - \mathbf{q}_k\|^2 \leq d_{jk}, \ \forall (v_j, v_k) \in E, \\ & \|\mathbf{q}_j - \tilde{\mathbf{q}}_j\|^2 \leq \rho_j, \quad j \in \mathcal{I} \end{aligned} \tag{6}$$

The difference between the feasibility version of Equation 2 and Equation 6 is that the last constraint in Equation 2 is the objective function in Equation 6. The feasible set for Equation 6 is non-empty and so Equation 6 always has a solution (the robots being at the positions where they started is a trivial feasible solution). When the optimal value of the objective in Equation 6 is less than ρ_i, the optimal solution for Equation 6 is a feasible solution for Equation 2.

In our incremental algorithm, we solve the optimization problem given by Equation 6, where we do not put all the constraints at once but put in the constraints as required. Note that the robot i is constrained by its neighbors only, each of which is constrained by their neighbors and so on. If it is possible for robot i to come within ρ_i of Q with its neighbors staying at their previous positions, then robot i should do so. Therefore, we start by putting in the constraints for robot i only and this results in the following problem:

$$\begin{aligned} \min \quad & \|\mathbf{q}_i - \mathbf{q}_0\|^2 \\ \text{s.t.} \quad & \|\mathbf{q}_i - \tilde{\mathbf{q}}_j\|^2 \leq d_{ij}, \ \forall (v_i, v_j) \in E \end{aligned} \tag{7}$$

By not putting in the constraints for the other robots, we are essentially saying that they cannot move, i.e., $\mathbf{q_j} = \tilde{\mathbf{q}}_j, j = 1, \ldots, n, j \neq i$. Let the value of the optimal solution to Equation 7 be d^*_{i0}. If $d^*_{i0} \leq \rho_i$ we have a feasible solution with the new position of robot i being $\mathbf{q}_i$ and other robots remaining in their original position. If $d^*_{i0} > \rho_i$, we can find the set of constraints in the solution of Equation 7 that are active at the optimal solution, i.e., the j's for which $\|\mathbf{q}_i - \tilde{\mathbf{q}}_j\|^2 = d_{ij}$. For ease of exposition, let us assume that there is only one active constraint and let this constraint correspond to the kth robot. The fact that there will be at least one active constraint in the final optimal solution is a fundamental fact of convex optimization [12]. Obviously, the robot i cannot progress further towards the point Q, if these active constraints cannot be relaxed, i.e., if robot k

cannot be moved. So, in the next step, we allow the robots constraining the motion of robot i to move, i.e., we remove the set of constraints $\|\mathbf{q}_i - \tilde{\mathbf{q}}_k\|^2 \leq d_{ik}, k \in K$ from Equation 7 and add the set of constraints $\|\mathbf{q}_i - \mathbf{q}_k\|^2 \leq d_{ik}$, $\|\mathbf{q}_k - \tilde{\mathbf{q}}_\ell\|^2 \leq d_{k\ell}, (v_k, v_\ell) \in E, k \in K$ to Equation 7. The set K consists of all the robots that were preventing the motion of robot i at the previous step. Again we solve the optimization problem, check the value of the optimal solution, identify the active constraints and repeat the procedure until $d^*_{i0} \leq \rho_i$. In the optimal solution of the problem in Equation 7, the Lagrange multipliers (or dual variables) corresponding to the active constraints is positive and so the active constraints can easily be identified by checking the dual variables.

To illustrate the above mathematical description of the incremental algorithm, let us consider the example that we introduced in Figure 1. We consider a network of 10 robots, where the robot 1 wants to move within the communication range of robot 10. Now, if we allow only robot 1 to move towards robot 10, robot 2 and 4 will constrain its motion if they are fixed. This corresponds to the first step of the incremental algorithm where we will obtain the active constraints as $1-2$ and $1-4$. So, in the second step, we will allow 1, 2, and 4 to move. Now the position of 2 and 4 will be constrained by robot 3. Robot 1 will then be constrained by either 2 or 4 or both. The new active constraints that will be obtained from the second step of the incremental algorithm is $2-3$ and $4-3$. Therefore, in the third step, we will allow robots $1, 2, 3$, and 4 to move. In this step the robot 1 can reach its goal, hence we obtain the final reconfigured network.

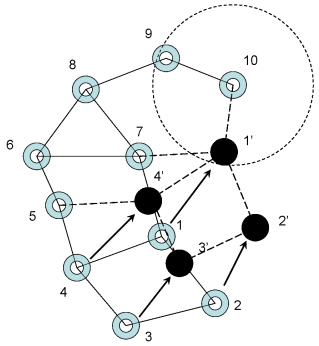

Fig. 2. An example network of 10 robots. The robot 1 wants to move within the communication range of robot 10 such that the network becomes 2-connected. Only the robots $1, 2, 3, 4$ need to move to their new position $1', 2', 3', 4'$ for the network to be 2-connected. The dashed lines shows the new communication edges after the network has reconfigured.

Once we obtain the feasible solution, we can also obtain the total distance moved by the robots by adding the distances moved by the robots that are allowed to move. However, this is an upper bound on the minimum distance that the robots need to move since the d^*_{i0} may be strictly less than ρ_i. If we are interested in finding the solution to the minimum distance problem we can now solve Equation 2 with only the constraints present in the optimal solution of Equation 7.

Technically, the incremental algorithm described above is also a centralized algorithm, because the computation is done by one robot only (the robot i that wants to move). In terms of computational complexity, the worst case performance of the incremental algorithm is similar to the centralized algorithm. However, unlike the centralized algorithm, the node i does not need the information about the whole network at once, but can collect the information as required. At each step, the robot i collects the position information about only the neighbors of the robots that need to be moved. Thus, in practical problems, where the network adjustment can be done by moving only a few nodes, the incremental algorithm may result in substantial savings in computational and communication effort. For a node which is more than 1 hop away from the robot i, a path is known for requesting the required position information, because within the active set there will be a sequence of active constraints starting from robot i to that particular robot.

C. Solution Algorithms for $P2$

For problem $P2$, it can be easily seen that by solving problem $P1$ for all the n robots and finding the solution with minimum objective value among them we can solve $P2$. However, in practice, because of the geometric structure of the network, it may not be required to solve $P1$ for all possible robots but only for robots that are within a certain distance of Q. We are exploring this further.

VI. Simulation Results

In this section we present some preliminary simulation results showing the performance of the centralized and incremental algorithms on randomly generated networks. We present simulation results for problem P1 only. We used YALMIP [13] for modeling the optimization problem and used Sedumi [14], for solving second order cone programs. All the coding was done in MATLAB. For simplicity, we assumed that the communication radii of the robots are identical and equal to 1. Furthermore, in the simulations we considered only the communication constraints, i.e., the set V' was assumed to be empty. We generated the networks randomly (ensuring that it is connected at the beginning). For each generated network, the robot i to be moved was chosen randomly and the point Q was also chosen randomly within a distance of 7-hops from the robot i. We ran both the centralized algorithm and the incremental algorithm for each problem instance and the average run time over 50 instances for each value of n is shown in Table I. Note that the choice of 7-hops is arbitrary and is there to ensure that the distance from i to point Q is much smaller than the diameter of the convex hull of the robot positions. As can be observed from table I, the computation time for the incremental algorithm is relatively insensitive to the size of the network. Moreover, in all cases the solution was obtained with less than 30 robots moving to new positions.

VII. Discussion on Obstacle Avoidance

In all of the above discussion, we have assumed that a robot can move to any point in the environment, i.e., there are no obstacles in the environment. Presence of obstacles in the environment can (a) make candidate final positions in

	Number of nodes	Centralized Algorithm Run Time (s)	Incremental Algorithm Run Time (s)
1	10	1.4	1.2
2	50	2	1.4
3	100	2.7	1.8
4	200	4.8	3.4
5	500	19	7.6
6	1000	130	8.1

TABLE I

SAMPLE RUN TIMES, IN SECONDS, FOR THE CENTRALIZED ALGORITHM AND INCREMENTAL ALGORITHM SOLVED USING SEDUMI IN MATLAB. THE RUN TIMES WERE COMPUTED BY AVERAGING OVER 50 RANDOMLY GENERATED NETWORKS. ALL DATA WAS OBTAINED ON A 2.53 GHZ INTEL(R) P87000 DUAL CORE CPU WITH 2 GB OF RAM.

the environment infeasible and (b) change the feasible path for moving from initial position to final position. Since our goal in this paper is to obtain the final feasible positions (and not on computing the paths to move to the final position) we will consider the problem (a) above in this section. We note here that the convex optimization framework and algorithms presented in this paper can be extended to take into consideration obstacles. We assume that the obstacles can be defined as a union of convex sets where each convex set is a polytope. Since any obstacle O is a convex polytope, it can be represented as an intersection of half planes, i.e., $O = \{\mathbf{q} \in \mathbb{R}^2 | \mathbf{a}_j^T \mathbf{q} + b_j \leq 0, j = 1, \ldots, p\}$. Assuming the robots to be point robots, for the robot to avoid being positioned inside the obstacle, the distance between the robot and the obstacle should be greater than 0. An alternate way of writing the collision avoidance constraint for robot i and obstacle O is as follows:

$$\max_{j=1,\ldots,p} \{\mathbf{a}_j^T \mathbf{q}_i + b_j\} \geq 0 \tag{8}$$

Since the constraint above is a pointwise maximum of a finite number of linear functions in $\mathbf{q}_i$, it is a convex function in $\mathbf{q}_i$ [11]. Thus, we can write each collsion constraint at the final position as a convex constraint (actually as a set of linear inequality constraints). Therefore incorporating collision avoidance constraints would involve adding a collection of linear inequalities to Equation 2 and the problem remains a convex optimization problem.

The incremental algorithm for the feasibility problem can thus be implemented by taking into considerationthe collision avoidance constraints for all the robots that can move. However, the solution obtained from the feasibility problem will not necessarily lead to the solution for minimum distance moved.

VIII. CONCLUSION

In this paper we have provided algorithms for solving the network reconfiguration problem of moving a robot in the network within a specified distance of a given point. We showed that the above problem is a second order cone program (SOCP) and can thus be solved in polynomial time by a centralized interior point algorithm. However, the centralized algorithm is not scalable. Hence, we presented an incremental algorithm, that solves the feasibility problem by obtaining the information about the position of the robots and their immediate neighbors only if they are required to move. From our simulation results on randomly generated networks, we observed that the computation time for obtaining the feasible solution is relatively independent of the size of the network.

Future Work: We note that both the algorithms that we have presented are in essence centralized, because only one robot is doing the computation. Therefore, for problems, where a large fraction of the robots have to move the incremental algorithm will perform similar to the centralized algorithm. The optimization problem given by Equation 2 has a separable objective function. Hence it may be possible to obtain a distributed algorithm for this problem using dual decomposition techniques (analogous to what is done in [2]). The distributed algorithms will give rise to issues concerning synchronization. Moreover, if only a small percentage of the nodes are to be moved, most of the computation of the nodes may be wasteful. The tradeoff's associated with the various algorithms is an interesting question to pursue.

ACKNOWLEDGEMENTS

This research was partially supported by MURI grants FA9550-08-1-0356 and N00014-08-1-1186. Thanks to S. Okamoto and R.Zivan for early discussions on a different version of the problem.

REFERENCES

[1] M. Ji and M. Egerstedt, "Distributed formation control while preserving connectedness," in *IEEE Conference on Decision and Control*, San Diego, CA, December 2006, pp. 5962–5967.

[2] R. L. Raffard, C. Tomlin, and S. P. Boyd, "Distributed optimization for cooperative agents: Application to formation flight," in *IEEE Conference on Decision and Control*, Bahamas, December 2004, pp. 2453–2459.

[3] A. Kansal, W. Kaiser, G. Pottie, M. Srivastava, and G. Sukhatme, "Revonfiguration methods for mobile sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 4, pp. 22:1–22:28, October 2007.

[4] M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, December 2008.

[5] M. C. DeGennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *IEEE Conference on Decision and Control*, San Diego, CA, December 2006, pp. 3628–3633.

[6] A. Muhammad and M. Egerstedt, "Network configuration control via connectivity graph processes," in *ACC*, June 2006.

[7] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks," in *Proceedings of IEEE Infocom 2001)*, vol. 3, 2001, pp. 1665–1663.

[8] S. Srirangarajan, A. H. Tewfik, and Z.-Q. Luo, "Distributed sensor network localization using socp relaxation," *IEEE Trans. on Wireless Communications*, vol. 7, no. 12, pp. 4886–4895, December 2008.

[9] J. Spletzer and R. Fierro, "Optimal positioning strategies for shape changes in robot teams," in *IEEE International Conference on Robotics and Automation*, vol. 1, Barcelona, Spain, April 2005, pp. 754–759.

[10] J. Derenick and J. Spletzer, "Concex optimization strategies for coordinating large scale robot formations," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1252–1259, December 2007.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[12] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York, USA: John Wiley, 1993.

[13] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in MATLAB," in *Proc. of the CACSD conference*, Taipei, Taiwan, 2004.

[14] J. F. Sturm, "Using sedumi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11-12, pp. 625–653, 1999.