Efficient Motion Planning Strategies for Large-scale Sensor Networks

Jason C. Derenick, Christopher R. Mansley, and John R. Spletzer

Lehigh University, Computer Science and Engineering, Bethlehem, PA, 18015, USA
{jcd6,crm5,josa}@lehigh.edu

Summary. In this paper, we develop a suite of motion planning strategies suitable for large-scale sensor networks. These solve the problem of reconfiguring the network to a new shape while minimizing either the total distance traveled by the nodes or the maximum distance traveled by any node. Three network paradigms are investigated: centralized, computationally distributed, and decentralized. For the centralized case, optimal solutions are obtained in $O(m \log m)$ time in practice using a logarithmic-barrier method. Key to this complexity is transforming the associated Karush-Kuhn-Tucker (KKT) matrix to a mono-banded system solvable in $O(m)$ time. These results are then extended to a distributed approach that allows the computation to be evenly partitioned across the m nodes in exchange for $O(m \log m)$ messages in the overlay network. Finally, we offer a decentralized, hierarchical approach whereby follower nodes are able to solve for their objective positions in $O(1)$ time from observing the headings of a small number (2-4) of leader nodes. This is akin to biological systems (*e.g.* schools of fish, flocks of birds, *etc.*) capable of complex formation changes using only local sensor feedback. We expect these results will prove useful in extending the mission life of large-scale mobile sensor networks.

1 Introduction

Consider the initial deployment of a wireless sensor network (WSN). Ideally, the WSN is fully connected with a topology to facilitate coverage, sensing, localization, and data routing. Unfortunately, since deployment methods can vary from aerial to manual, the initial configuration could be far from ideal. As a result, the WSN may be congested, disconnected, and incapable of localizing itself in the environment. Node failures in established networks could have similar effects. Such limitations in static networks have lead to an increased research interest into improving network efficiency via nodes that support at least limited mobility [3].

Also of fundamental importance to WSN research is resource management, and perhaps most importantly power managment. Energy consumption is the most limiting factor in the use of wireless sensor networks, as service life is limited by onboard battery capacity. This constraint has driven research into power sensitive routing protocols, sleeping protocols, and even network architectures for minimizing data traffic

[1, 2, 12]. It would only seem natural then to develop motion planning strategies with similar performance objectives.

In this vein, we propose a set of *motion planning strategies* that allow a mobile network to reconfigure to a new geometry while minimizing the total distance the nodes must travel, or the maximum distance that any node must travel. We believe a suite of strategies is critical due to the proliferation of non-standard sensor network architectures which are often implementation specific. As such, we provide centralized, computationally distributed, and decentralized approaches suitable for use with large-scale sensor network architectures. Each is computationally efficient, and without onerous communication overhead. We expect these results will be useful for extending the mission life of large-scale mobile networks.

2 Related Work

Changes to the environment, mission objectives, and node failures are all factors that can contribute to need for reconfiguring a sensor network. However, tolopology changes can also be driven by performance objectives. For example, Cortes *et al* applied optimization based techniques to motion planning for improving network coverage [8]. Similarly, Zhang and Sukhatme investigated using motion to control node density [23]. The work of Hidaka *et al* investigated deployment strategies for optimizing localization performance [18], while the work of Butler and Rus was motivated by event monitoring using constrained resources [7].

There has also been extensive research in "collective" robot motion within the control community. Feddema *et al*. applied decentralized optimization based control to solve a variety of multi-robot problems [13]. Also worth noting is work in the areas of formation control [24], conflict resolution [19], and cooperatice control [4]. A recent survey/tutorial outlining additional relevant work within each of these areas can be found in [14].

In contrast to these efforts, the focus of our work is efficient motion planning strategies suitable for large-scale networks. Given initial and objective network geometries, we determine how to optimally reposition each node in order to achieve the objective configuration while minimizing the distances that the nodes must travel. The objective positions can then be fed to appropriate controllers to drive the nodes to their desired destinations. When servo/actuator costs dominate the power budget, such approaches can dramatically improve the network mission life. We also emphasize applicability to large-scale systems. Our methods scale well in terms of both computational and message complexity to ensure that advantages gained through efficient motion planning are not compromised by excessive computation or routing requirements. Finally, we provide centralized, computationally distributed, and decentralized models to support the diverse array of WSN architectures.

3 The Motion Planning Problem

In developing our motion planning strategies, we leverage results from our previous work [22] where it was shown that the *shape* of a robot formation could be expressed as the set of linear equality constraints

$$\begin{array}{ll} \parallel s_2 - s_1 \parallel (q_i^x - q_1^x) - (s_i^x, s_i^y)^T (q_2 - q_1) = 0, & i = 3 \ldots m \\ \parallel s_2 - s_1 \parallel (q_i^y - q_1^y) - (s_i^y, s_i^x)^T (q_2 - q_1) = 0, & i = 3 \ldots m \end{array} \tag{1}$$

where $Q = [q_1, \ldots, q_m]^T \in \mathbb{R}^{m \times 2}$ denote the concatenated coordinates of the objective formation shape, $S = [s_1, \ldots, s_m]^T \in \mathbb{R}^{m \times 2}$ represents an instance or an *icon* of our objective shape, and the (x, y) superscripts denote the specific Euclidean coordinate.

These constraints define the equivalence class of the full set of similarity transformations of the formation. Thus, if an objective shape Q and an icon S satisfy these constraints, the two shapes are equivalent under the Euclidean similarity transformations of translation, rotation and scaling. This is a traditional definition of shape employed in statistical shape analysis [11]. So, given an initial formation position $P = [p_1, \ldots, p_m]^T \in \mathbb{R}^{m \times 2}$, and an objective shape icon S, the problem becomes finding the set of objective positions $Q \sim S$ such that

1. $\max \parallel q_i - p_i \parallel$ is minimized for $i = 1, \ldots, m$ OR
2. $\sum_{i=1}^{k} \parallel q_i - p_i \parallel$ is minimized.

In other words, if the network were given an objective icon S, it must determine the objective positions for each node that minimize the maximum distance or the total distance the nodes must travel, and where the final shape Q is equivalent to S.

Since the constraints are linear in Q, the problems can be modeled as the respective second-order cone programs (SOCPs)

$$\begin{array}{ll} \min\limits_{q,t_1} t_1 & \min\limits_{q,t} \sum\limits_{i=1}^{k} t_i \\ \text{s.t.} \parallel q_i - p_i \parallel_2 \leq t_1, & \text{s.t.} \parallel q_i - p_i \parallel_2 \leq t_i \\ \quad Aq = 0 & \quad Aq = 0 \end{array} \tag{2}$$

for $i = 1, \ldots, m$ and where the A matrix corresponds to the constraint set defined by (1). Since the SOCPs are convex, a local minimum corresponds to a global minimum. This allows optimal solutions to be obtained through a variety of methods such as descent techniques or (more efficiently) by interior point methods (IPMs). While primal-dual IPMs represent perhaps the most efficient algorithms, we employ a simpler barrier IPM. It provides good computational complexity in practice, and as we shall see lends itself to a computationally distributed implementation.

Finally, we should note that in the interest of brevity the results presented here focus on the SOCP for the *mini-max* distance metric as defined in Equation 2 (left). Similar results for the total distance metric can be found in the extended technical report version of this paper [10].

4 A Centralized Approach

Centralized approaches are appropriate for hierarchical network architecures such as the TENET [12]. In this paradigm, more powerful "master" nodes execute all computationally complex algorithms, while the small-form-factor (SFF) motes merely provide data input. Thus for the motion planning problem, master nodes acting as cluster heads would calculate the objective positions for the cluster and communicate these to the SFF nodes. While simple in design, the hierarchy requires that algorithms scale scale well computationally with the size of the network.

To address this, we solve the motion planning problem by adapting the logarithmic penalty-barrier approach [5]. Like other IPMs, the complexity is largely defined by solving a linear system of equations. In this case, Equality-constrained Newton's method (ENM) is used for internal minimization and the linear system is in KKT form. We show that by reformulating the SOCP, we can band the KKT matrix to solve the system in $O(m)$ time via algorithms that exploit knowledge of matrix bandwidth. Furthermore, we show empirically that the total number of iterations required to reduce the duality gap to a desired tolerance is $O(\log m)$. The result is a simple IPM that in practice solves the shape problems in order $O(m \log m)$ time.

4.1 Reformulating the Shape Problems

The original shape problem can be restated in a relaxed form suitable for solving via the barrier approach. Conversion requires augmenting the objective function given in (2) with the log-barrier terms corresponding to the problem's conic constraints. The problem is restated in its equivalent form as follows:

$$\begin{aligned} &\min_{q,t_1} \tau t_1 - \sum_{i=1}^{m} \log\left(t_1^2 - u_i^T u_i\right) \\ &\text{s. t. } Aq = 0 \\ &\qquad u_i = q_i - p_i \end{aligned} \tag{3}$$

For the sake of clarity, the linear constraint $u_i = q_i - p_i$ is included; however, it is assumed that it will be eliminated by substitution upon implementation.

4.2 Banding the KKT System

Noting that the KKT system for the shape problem is symmetric indefinite, we employ Gaussian elimination with non-symmetric partial pivoting. The performance of Gaussian elimination suffers significantly due to fill-in when the linear system in question features dense rows and/or columns [21]. Observing the non-zero dot-plot of the nominal KKT formulation given in Figure 1 (left), it is evident that (3) does not facilitate efficient solving due to the dense features. As our objective is to make the KKT system banded so that it can be solved in $O(m)$ [20], we restate the problem in the following equivalent form:

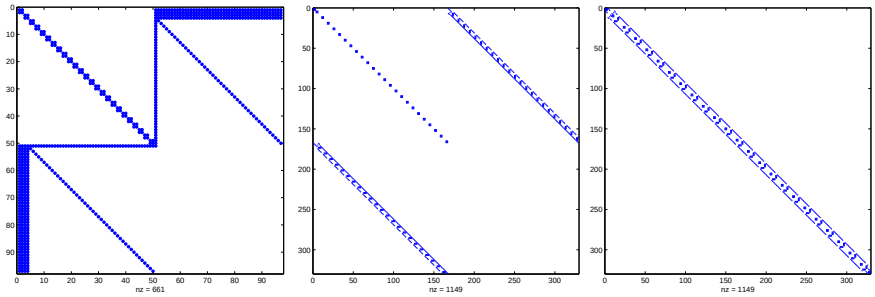

Fig. 1. The dot plots of the KKT systems for the unregulated orientation/scale shape problem. The pattern corresponds to a system of 25 nodes in $\mathbb{R}^2$. (Left) KKT system sparsity structure for the maximum distance metric problem. (Center) Augmented KKT system sparsity structure. (Right) The banded KKT system with a lower and upper bandwidth of 8.

$$\begin{aligned}
&\min_{q,t} \ \frac{\tau}{m}\sum_{i=1}^{m} t_i - \sum_{i=1}^{m}\log\left(t_i^2 - u_i^T u_i\right) \\
&\text{s. t. } Aq = 0 \\
&\qquad t_{i+1} = t_i, && i = 1,\ldots,m-1 \\
&\qquad d_{2i+1} = d_{2i-1}, && i = 1,\ldots,m-3 \\
&\qquad d_{2(i+1)} = d_{2i}, && i = 1,\ldots,m-3 \\
&\qquad u_i = q_i - p_i, && i = 1,\ldots,m \\
&\qquad d_1 = q_1 \\
&\qquad d_2 = q_2
\end{aligned} \tag{4}$$

Observe that the objective has changed from (3); however, we see that both forms are equivalent since:

$$\frac{\tau}{m}\sum_{i=1}^{m} t_i = \frac{\tau}{m}\sum_{i=1}^{m} t_1 = \left(\frac{\tau}{m}\right) m t_1 = \tau t_1 \tag{5}$$

The first equality holds due to the equality constraints placed on t_i.

Given this formulation, our claim is that the system can be made banded. In order to show the validity of this statement, we begin by defining the nominal solution vector for the KKT matrix. It is given as follows:

$$\left[\delta\eta_1^T, \delta\eta_2^T, \delta\kappa_1^T, \ldots, \delta\kappa_{(m-2)}^T, \mu^T\right]^T \tag{6}$$

$$\delta\eta_i = \begin{bmatrix}\delta q_i \\ \delta t_i\end{bmatrix} \quad \delta\kappa_i = \begin{bmatrix}\delta d_{2(i-1)+1} \\ \delta d_{2(i-1)+2} \\ \delta\eta_{(i+2)}\end{bmatrix} \quad \mu = \begin{bmatrix} w_1 \\ \vdots \\ w_{7m-13}\end{bmatrix}$$

and the δ variables correspond to the primal Newton step components associated with each of the respective system variables.

Given the new objective function and assuming the shape problem's solution vector permutation corresponds with the permutation yielding (6), the Hessian for our problem is now:

$$H = \begin{bmatrix} \psi_1 & \dots & 0 \\ & \psi_2 & \\ \vdots & \psi_3 & \vdots \\ & & \ddots \\ 0 & \dots & \psi_m \end{bmatrix} \tag{7}$$

$$\psi_i = \nabla^2\phi(u_i, t_i),\ i \in \{1, 2\}$$
$$\psi_i = \begin{bmatrix} 0_{4\times4} & 0_{4\times3} \\ 0_{3\times4} & \nabla^2\phi(u_i, t_i) \end{bmatrix},\ i \in \{3, \dots, m\}$$

where $\nabla^2\phi(u_i, t_i)$ is defined as in [16] with $u_i = q_i - p_i$. Notice that the Hessian is block-diagonal and separable. This differs from the nominal Hessian form, which features a dense row and column corresponding to t_1. This is evident by observing the upper left quadrant (defined by H) of the KKT system presented in Figure 1 (left).

Similarly, we can eliminate the dense columns and rows in A (and A^T) by introducing $2(m-2)$, d_j variables along with their associated $4(m-3)$ equality constraints. Doing so allows us to rewrite (1) as

$$q_i^x - d_j^x = \frac{s_i^x}{\| s_2 \|}\left(d_{j+1}^x - d_j^x\right) - \frac{s_i^y}{\| s_2 \|}\left(d_{j+1}^y - d_j^y\right) \tag{8}$$

$$q_i^y - d_j^y = \frac{s_i^x}{\| s_2 \|}\left(d_{j+1}^y - d_j^y\right) + \frac{s_i^y}{\| s_2 \|}\left(d_{j+1}^x - d_j^x\right) \tag{9}$$

for $i = 3, \dots, m$, and $j = 2(i-3)+1$. By reformulating the linear shape constraints in this fashion, we are now able to construct A as a *pseudo-banded* system. We say pseudo-banded because the matrix is non-square but features a band-like structure.

We now define the nominal form of the linear constraint matrix, A. We begin by defining the constraints associated with q_1 and q_2 as follows:

$$\begin{aligned} \varrho_1 &\triangleq q_1^x = d_1^x \\ \varrho_2 &\triangleq q_1^y = d_1^y \\ \varrho_3 &\triangleq t_1 = t_2 \\ \varrho_4 &\triangleq q_2^x = d_2^x \\ \varrho_5 &\triangleq q_2^y = d_2^y \end{aligned}$$

Similarly, for $3 \leq i \leq (m-1)$, we define the constraints associated with q_i as:

$$\begin{aligned}
\varphi_{i_1} &\triangleq q_i^x - d_j^x = \beta_1 \left(d_{j+1}^x - d_j^x\right) - \beta_2 \left(d_{j+1}^y - d_j^y\right) \\
\varphi_{i_2} &\triangleq q_i^y - d_j^y = \beta_1 \left(d_{j+1}^y - d_j^y\right) + \beta_2 \left(d_{j+1}^x - d_j^x\right) \\
\varphi_{i_3} &\triangleq t_i = t_{i-1} \\
\varphi_{i_4} &\triangleq d_{j+2}^x = d_j^x \\
\varphi_{i_5} &\triangleq d_{j+2}^y = d_j^y \\
\varphi_{i_6} &\triangleq d_{j+3}^x = d_{j+1}^x \\
\varphi_{i_7} &\triangleq d_{j+3}^y = d_{j+1}^y
\end{aligned}$$

where $\beta_1 = \frac{s_i^x}{\|s_2\|}$, $\beta_2 = \frac{s_i^y}{\|s_2\|}$, and j is as previously defined.

Finally, we associate q_m with the remaining three constraints:

$$\begin{aligned}
\varphi_{m_1} &\triangleq q_m^x - d_j^x = \beta_1 \left(d_{j+1}^x - d_j^x\right) - \beta_2 \left(d_{j+1}^y - d_j^y\right) \\
\varphi_{m_2} &\triangleq q_m^y - d_j^y = \beta_1 \left(d_{j+1}^y - d_j^y\right) + \beta_2 \left(d_{j+1}^x - d_j^x\right) \\
\varphi_{m_3} &\triangleq t_m = t_{m-1}
\end{aligned}$$

where $\beta_1 = \frac{s_m^x}{\|s_2\|}$, $\beta_2 = \frac{s_m^y}{\|s_2\|}$, and $j = 2(m-3)+1$.

Given these definitions, we define the following nominal row permutation for the linear constraint (coefficient) matrix A as:

$$\left[\vartheta^T, \varkappa_1^T, \ldots, \varkappa_{(m-1)}^T, \varsigma^T\right]^T \tag{10}$$

$$\vartheta = \begin{bmatrix} \varrho_1 \\ \varrho_2 \\ \varrho_3 \\ \varrho_4 \\ \varrho_5 \end{bmatrix} \quad \varkappa_i = \begin{bmatrix} \varphi_{(i+2)_1} \\ \varphi_{(i+2)_2} \\ \varphi_{(i+2)_3} \\ \varphi_{(i+2)_4} \\ \varphi_{(i+2)_5} \\ \varphi_{(i+2)_6} \\ \varphi_{(i+2)_7} \end{bmatrix} \quad \varsigma = \begin{bmatrix} \varphi_{m_1} \\ \varphi_{m_2} \\ \varphi_{m_3} \end{bmatrix}$$

Notice that all of the primal constraints defined in (4) have been included in A.

Given the definitions of A and H, the banded KKT system can now be constructed. Symmetrically applying the permutation that yields the following KKT system solution vector ordering:

$$\left[\lambda^T, \xi_1^T, \ldots, \xi_{(m-3)}^T, \chi^T\right]^T \tag{11}$$

$$\lambda = \begin{bmatrix} \delta q_1 \\ \delta t_1 \\ w_1 \\ \vdots \\ w_5 \\ \delta q_2 \\ \delta t_2 \end{bmatrix} \quad \xi_i = \begin{bmatrix} \delta d_{2(i-1)+1} \\ \delta d_{2(i-1)+2} \\ w_{6+7(i-1)} \\ \vdots \\ w_{12+7(i-1)} \\ \delta q_{(i+2)} \\ \delta t_{(i+2)} \end{bmatrix} \quad \chi = \begin{bmatrix} \delta d_{2m-5} \\ \delta d_{2m-4} \\ w_{7m-15} \\ w_{7m-14} \\ w_{7m-13} \\ \delta q_m \\ \delta t_m \end{bmatrix}$$

produces a mono-banded system with total bandwidth $\sigma \leq 17$. Notice that permuting the KKT system in this way preserves symmetry. As such, its upper bandwidth, b_u, is equal to its lower bandwidth, b_l (both having a value of 8).

In Figure 1 (center), we show the standard form KKT system constructed from the Hessian given by (7) and the linear constraints given in (4). The constraints (appearing in the lower-left quadrant) are permuted to correspond with (10). Notice that A (and A^T), in its nominal form, resembles a banded system. Taking the KKT system in this form and symmetrically permuting its rows and columns according to (11) yields the mono-banded system appearing in Figure 1 (right). The system corresponds to a team of 25 agents dispersed in $SE(2)$. It can now be efficiently solved in $O(m)$ using a band-diagonal LU-based solver [20].

4.3 Complexity of the Shape Problem in Practice

In the previous section, we proved that the KKT system for our motion planning problem can be solved in $O(m)$ operations. We now show empirically that this enables the SOCP to be solved in $O(m \log m)$ operations in practice.

Experimental Setup: In order to quantify the performance of this approach, 10,000 instances of the motion planning problem were solved using an implementation of the barrier algorithm for SOCPs outlined in [5]. Values of m were considered between 10 and 1000 using a step size of 10. For each value of m, a total of 100 random shape SOCPs were generated for solving. This yielded a total of 10,000 random problem instances. Of those instances, one problem could not be solved due to an ill-conditioned KKT system. The validity of our implementation was established by comparing obtained results against those of the Mosek industrial solver [17].

Outer Iteration Complexity: In order to characterize the algorithm's complexity, we consider the total number of Newton iterations required to reach optimality. In [5], it is shown that the total number of barrier iterations grows with $\log m$, and the number of Newton iterations per barrier iteration grows with m. As such, a conservative bound of $O(m \log m)$ can be placed on the total number of Newton iterations. However, our empirical results show that the number of Newton iterations required per barrier iteration remains constant, resulting in a total number of iterations that grows as $O(\log m)$ in practice.

Figure 2 (left) shows the mean number of Newton iterations required per barrier iteration. We see that the number essentially remains constant for $m \gtrapprox 50$. The tightness of the distributions suggests the number of iterations will be ≈ 4, regardless

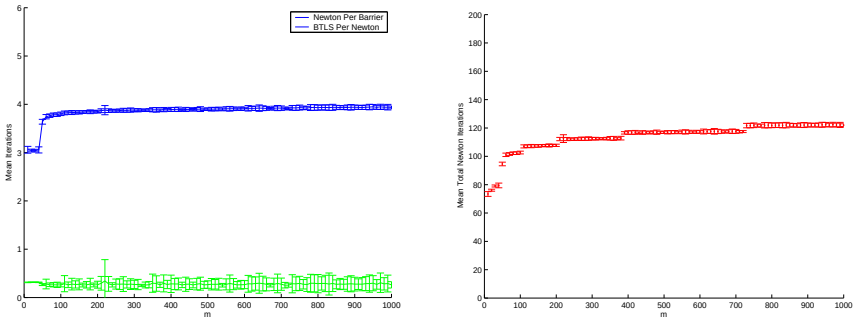

Fig. 2. (Left) The mean number of Newton iterations per barrier iteration and BTLS iterations per Newton iteration as a function of m. Error bars signify a single standard deviation for each sample set. (Right) The mean total number of Newton iterations required to reduce the duality gap to within 10^{-4} of optimality. These indicate a linear per-iteration complexity, with an iteration complexity of $O(\log m)$ for a total expected complexity of $O(m \log m)$.

of the value of m. Given this result and noting that the barrier algorithm terminates after $O(\log m)$ steps [5], we see that a better estimate for the total number of Newton iterations required to solve the motion planning problem is $O(\log m)$ in practice. This is corroborated by our empirical results shown in Figure 2 (right).

Per-Iteration Complexity: Since Equality-constrained Newton's Method (ENM) is employed, three main steps define the per-iteration complexity: 1) Solving the KKT system, 2) Computing the Newton Decrement and 3) Performing a Back-tracking Line Search (BTLS). We have shown previously that the KKT system can be solved in $O(m)$ operations. For computing the Newton decrement, recall that the relaxed problem yields a block-diagonal, separable Hessian. Given knowledge of the block-width, it is trivial to exploit this information in order to yield an $O(m)$ matrix-vector multiplication routine.

The final step is determining the complexity associated with using BTLS to compute the desired step length. From the definition of BTLS (see [5]), the number of iterations per Newton iteration is not obvious (as a function of m) since it largely depends on the objective function definition. However, each BTLS iteration corresponds to vector addition and evaluating the gradient, which can trivially be done in $O(m)$ time. Furthermore, Figure 2 (left) indicates that the mean number of BTLS iterations executed per Newton iteration is *less than 1 for all considered values of* m. This implies that a unit step size typically provides a sufficient decrease that ultimately satisfies Wolfe's condition. Considering the distributions (indicated via the single standard deviation bars), we can expect each Newton iteration to require only one or two BTLS iterations.

These empirical results, in conjunction with our previous analysis, suggests that the per-iteration complexity grows linearly with m. Thus, solving the SOCP for our centralized motion planning strategy will typically require only $O(m \log m)$ basic operations in practice.

5 A Computationally Distributed Approach

Our centralized solution features both a band-diagonal linear system as well as a separable objective function. We shall leverage these characteristics to distribute the computational workload evenly across the network. The resulting $O(\log m)$ expected per-node workload should enable our approach to be employed by a significantly less sophisticated class of processors, or to significantly larger-scale networks. We now define a hierarchical, cluster-based architecture for achieving this objective.

5.1 Architectural Overview

Our paradigm solves convex optimization problems in the context of a hierarchical cluster-based network paradigm under the direction of some *root* node(s). The *root* is responsible for orchestrating the solve process; thus, it maintains a global state reflecting the status of the distributed computation. It is responsible for performing such tasks as initializing the network and determining when the solve process is complete. Although the *root* maintains a global perspective per se, its data view is primarily limited to that which affects the computation of its associated decision variables. The only exception is when it requests data from the network to determine the next state in the IPM solve process. For instance, when the $root$ issues a request for Newton decrement data.

At the *root*'s disposal are the remaining nodes in the network, which we term the *secondary* peers. These nodes are considered *secondary*, because they serve only as a distributed memory pool and a computational engine for the *root* during the solve process; individually, they lack a global view of the solver and only manage data relevant to their computations. They wait until a data request is received that originates from the *root* or some neighboring clusterhead before transitioning into a state of distributed computation.

To reduce the communication overhead experienced at any node, we define the architecture to have a hierarchical scheme based upon network clusters. The role of clusterheads is to ensure that each request of the *root* is satisfied at the lowest level. Sub-nodes treat their clusterhead as a local accumulator and forward the requested information to that node where it is aggregated before being passed up the hierarchy, ultimately to the *root*. The result is that the $root$ (and all clusterheads) only need to send a constant number of messages with each data request.

5.2 Distributing and Solving the KKT System

Given the objective function and Hessian are separable, implementing a distributed Newton decrement or BTLS computation reduces to having each node pass its contribution to the greater value up the cluster hierarchy at request. For this reason, along with the fact that the per-iteration complexity of ENM is largely defined by solving the KKT system, we focus our discussion on distributing the LU solver. As will be seen, we can effectively distribute the process while providing per-node message, computation, and storage complexities of $O(1)$.

To properly distribute the KKT system, $K \in \mathbb{R}^{y \times y}$, among m nodes in a WSN, we make the assumption that the system is band-diagonal with respective upper and lower bandwidths of b_u and b_l. Additionally, we assume the matrix is represented in its equivalent compact form, K_c, where $K_c \in \mathbb{R}^{y \times (b_l + b_u + 1)}$ [20]. We denote the solution vector or right-hand-side vector of the KKT system as b, where $b \in \mathbb{R}^y$.

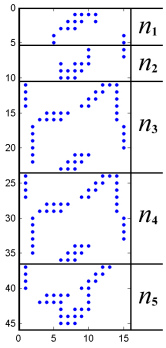

Fig. 3. A non-zero dot-plot illustrating the decomposition of the compact KKT (i.e. K_c) system for a configuration of 5 nodes in $SE(2)$ solving the total distance metric. For this problem, $b_l = b_u = 7$. Notice that the middle $(m-3)$ nodes (i.e. n_3 and n_4) are assigned sub-blocks with identical structure.

Adopting this representation for K, we adapt the LU-based solver with partial pivoting outlined in [20]. Distributing this algorithm, we begin by assigning the i^{th} node, n_i, a sub-block, K_c^i, of K_c. Each n_i also manages a sub-vector, b_i, of b. The sub-vector contains the values corresponding to the equations contained in K_c^i. To illustrate this decomposition, we provide Figure 3, which shows the distribution of K_c for a team of 5 nodes in $SE(2)$. Given the dependencies between the equations in the linear system, devising a completely concurrent solution is not feasible. Thus, we assume the decomposition and subsequent solves are done one node at a time in a *pass-the-bucket* fashion, where node n_i decomposes K_c^i and then hands the process off to node $n_{(i+1)}$. This process continues iteratively until decomposition is complete. Both the forward substitution and backward substitution phases are conducted in a similar manner.

Decomposition: During the decomposition phase, the algorithm employs partial pivoting by searching at most b_l sub-diagonal elements in order to identify one with greater magnitude. This implies that a node in our WSN that is performing its respective decomposition may only need information pertaining to at most b_l rows, which may be buffered at one or more peers. In the worst case scenario, where each node only manages a single row, node n_i may have to query up to b_l of its peers (particularly, it may have to contact nodes $n_{(i+1)}, \ldots, n_{(i+b_l)}$). With this result in mind, and defining $\psi(i)$ as the number of peers node n_i has to contact, we offer the following theorem:

Theorem 1. *Let $i \in \{1,\ldots,m\}$ and let $K_c^j \in \mathbb{R}^{u_j \times (bl+bu+1)}, u_j > 0$ for $j = 1,\ldots,m$. Define $\psi(i)$ as a mapping from the node index/id i to the number of nodes that have to be contacted by n_i during the decomposition of sub-block K_c^i. Given this definition and these assumptions, the following holds:*

$$\psi(i) \leq \phi(b_l, u_1, \ldots, u_m) = \left\lceil \frac{b_l}{\left(\min_{i \in \{1,\ldots,m\}} u_i\right)} \right\rceil$$

Proof. By contradiction.
Assume $\psi(i) > \phi(b_l, u_1, \ldots, u_m)$. Choosing $u_i = 1, \forall i \in \{1,\ldots,m\}$, we see:

$$\psi(i) > \left\lceil \frac{b_l}{1} \right\rceil = b_l$$

However, it must hold that $\psi(i) \leq b_l$, since n_i will only ever require data about b_l rows during the decomposition of K_c^i. $\perp$

To illustrate this result, consider Figure 3 again. Based on the assignment for our shape problem, we see that all nodes in the network except for n_1 (which will only require communicating with two other nodes) will require communicating with at most a single node in order to successfully complete decomposition.

As decomposition progresses, node n_i iteratively constructs a permutation vector, p_i. Observing the algorithm, it is evident that the permutation value assigned to the j^{th} position of p_i will be no more than $(j + b_l)$. This fact becomes important when forward substitution is started, because it implies that the solving node will have to communicate with at most ϕ peers to resolve the values of b_i during this phase.

Once n_i has decomposed its sub-block, it notifies each of the $\psi(i)$ nodes (i.e. nodes $n_{(i+1)}, \ldots, n_{(i+\psi(i))}$) from which it acquired row information, before handing off the decomposition process. The content of each message is the modified row(s) and the adjusted permutation vectors corresponding to the changes n_i made with respect to row data each peer provided. Each peer updates the rows of its sub-block as well as the corresponding elements in p_i, before the process is handed off to $n_{(i+1)}$.

Forward Substitution: Similar to the decomposition phase, the forward substitution step is done in an iterative manner. In order to successfully solve its sub-block, n_i requires information from each of its supporting $\psi(i)$ peers. These nodes must provide the corresponding rows that may be required by n_i's forward substitution as well as any relevant components of their respective b sub-vectors. Upon completion, n_i sends a message to each of the peers with updated values for their respective b sub-vectors. It concludes by handing off the process to $n_{(i+1)}$.

Backward Substitution: The backward substitution phase begins when n_m completes the forward solve on its sub-block, K_c^m. Unlike the forward substitution phase, this phase requires a node to communicate with at most 2ϕ peers. The additional messaging is introduced via the upper triangular factor having a bandwidth constrained

no longer by b_u. As a result of row permutations, its bandwidth is instead bounded by $(b_u + b_l + 1)$ [15]. To complete the backward substitution phase, n_i simply must acquire the row information and b vector data from said peers (all of which have already completed their respective backward substitutions). The process completes when the node managing K_c^1 solves its sub-block.

LU-solver Complexity: For simplicity, the assumption is made whenever n_i requests row or vector data from any of its peers, the data is received in a single message. This assumption is reasonable, because the amount of row and vector data that has to be shared between any two nodes is a function of b_u and b_l, which are both independent of configuration size. As such, the number of messages required to transmit said data is also constant. Noting that data is delivered upon request, we see that the total number of messages sent by n_i is:

$$O(2\psi(i) + 2\psi(i) + 4\psi(i)) \equiv O(8\phi) \equiv O(1) \tag{12}$$

Since all m nodes send $O(1)$ messages during the solve, the total message complexity for the distributed LU process is $O(m)$. Furthermore, since n_i manages some K_c^i (along with b_i and p_i) and row data received by as many as $\psi(i)$ peers, storage is also bounded by a constant that is a function of b_l and b_u. With $O(\log m)$ expected iterations, this translates to a total of $O(m \log m)$ messages in the overlay network.

5.3 Experimental Results

To demonstrate our approach, we implemented the distributed framework on a team of six Sony Aibos and charged the team with transitioning to a delta formation. Each Aibo was outfitted with a unique butterfly pattern [6] that was tracked via an overhead camera system serving as an indoor "GPS". Figure 4 (left) shows the initial

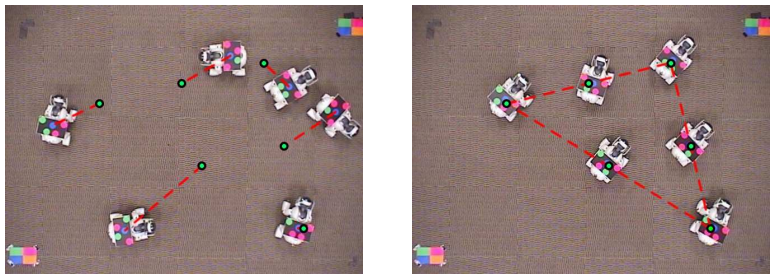

Fig. 4. (Left) An initial dispersion of 6 Aibos, along with overlaid lines/points mapping each to its computed optimal position. (Right) The Aibos after reconfiguring to the desired delta shape formation. All computations were done in a distributed fashion, with each dog being responsible for computing its optimal position.

configuration, along with lines mapping each to its computed optimal position. The objective was to minimize the total distance traveled by the team. Figure 4 (right)

shows the Aibos after transitioning to the optimal shape configuration. All computations were done in a distributed fashion, with each dog being responsible for computing its objective position and local control inputs.

6 A Decentralized Hierarchical Approach

For our decentralized approach, we assume a hierarchical network design whereby a small number of *leader* nodes acting as exemplars solve the motion planning problem. This allows the remaining *follower* nodes to infer their objective positions through local observations. Such a model is attractive to not only hierarchical network architectures [12], but also models where minimizing data communication is a primary objective [1, 2].

For our decentralized approach, we make the following assumptions.

1. Each node knows the objective shape icon S for the network.
2. Leader nodes (individually or collectively) know the current network shape.
3. Follower nodes have *no knowledge* of the current network shape.
4. Follower nodes can identify their neighbors and measure their *relative* position.
5. Follower nodes can observe the *relative* heading of their immediate neigbors.

6.1 An $O(1)$ Decentralized Solution

Key to this approach is the realization that although the optimization problem includes $2m$ decision variables (corresponding to the m robot positions), the feasible set is constrained to the equivalence class of the full set of similarity transformations for the objective formation shape. More concisely, there are only 4 degrees of freedom in determining a node's objective position on the plane which correspond to the translation, rotation, and scale of the objective shape.

As the leader nodes have knowledge of the current and objective shapes, they can solve for their objective positions using either of the approaches outlined in Sections 4-5. Follower nodes have more constrained knowlege, and as a result are incapable of estimating their objective positions. However, an observation of the heading ω_l of leader l introduces an additional constraint on the objective shape of the form $(q_l - p_l)^T(\sin\omega_l - \cos\omega)_l = 0$ where all measurements are relative to the follower's coordinate frame $\mathcal{F}$. If the headings of 4 leader nodes can be observed, the motion planning problem becomes fully constrained via the equality constraints. Perhaps more significant is that the problem can now be solved by the followe node in decentralized fashion, and in $O(1)$ time regardless of formation size.

To see this, recall that in addition to this heading constraint, each robot imposes two additional equality constraints on the objective network shape as shown in Equation 1. With 4 leader nodes and 1 follower node, this corresponds to a total of 4 bearing and 10 shape constraints over 14 decision variables. However, noting that the shape index (*not* coordinate) assignments are arbitrary, the follower node can designate itself as the first index corresponding to the 3-tuple $\{p_1, q_1, s_1\}$ and associate

one of the observed leaders with $\{p_2, q_2, s_2\}$. This eliminates the associated shape constraints for these two nodes, and reduces the set to

$$\begin{array}{rl} (q_l - p_l)^T(\sin\omega_l, -\cos\omega_l) = 0, & l = 2\ldots5 \\ \parallel s_2 - s_1 \parallel (q_l^x - q_1^x) - (s_l^x, s_l^y)^T(q_2 - q_1) = 0, & l = 3\ldots5 \\ \parallel s_2 - s_1 \parallel (q_l^y - q_1^y) - (s_l^y, s_l^x)^T(q_2 - q_1) = 0, & l = 3\ldots5 \end{array} \quad (13)$$

where $l \in \{2\ldots5\}$ now corresponds to the set of observed leaders. The constraint set is linear in q, and can be written in the form $A\hat{q} = b$, where the solution vector $\hat{q} \subset q$ is the objective positions of follower and 4 observed leader nodes. It is a linear system of 10 equations in 10 unknowns, and is readily solvable via Gaussian elimination techniques.

Thus, each follower node can solve for its objective position (as well as its neighbors) so long as the *relative* position and headings of 4 neighbors can be observed. This is akin to biological systems (*e.g.* schools of fish, flocks of birds, *etc.*) capable of complex formation changes using only local sensor feedback. Furthermore, the solution is obtained from solving an $O(1)$ sized (10×10) linear system of equations - regardless of the number of nodes in the network. The assumption of knowledge of the objective shape does however require $O(m)$ storage for each node.

It should also be noted that after solving for its objective position, each follower is "promoted" to leader status. As it migrates to its objective position, its heading can be observed by other follower nodes to solve their own decentralized problem. So, while in practice the actual number of leader nodes will be a function of the sensor network topology, in theory only 4 are *necessary*. This is illustrated below.

6.2 Simulation Results

Figure 5 models the intial deployment of a sensor network. The objective configuration was a {4,4} tessellation on the plane with a tiling size of 10 meters. Unfortunately, positional errors introduced during deployment - modeled as Gaussian noise $\sim N(0, \sigma_x = \sigma_y = 7.5)$ - result in a significantly different geometry (Figure 5a). To compensate for these errors, four leader nodes (red circles) solve the motion planning problem, and begin migrating to their objective positions. Relative sensor measurements allow the remaining follower nodes (blue triangles) to solve for their objective positions in decentralized fashion. The propagation of decentralized solutions through the network is reflected in Figure 5b. The decentralized trajectories that minimize the maximum distance that any node must travel, and the optimal network configuration achieving the desired shape are shown in Figures 5c-d. It was assumed that the sensing range of each node was 25 meters.

Note that in this case, the orientation of the shape was not constrained. If a fixed orientation was desired (*e.g.*, orthogonal to the $x-y$ axes), the number of degrees of freedom would be reduced to 3 - as would the number of observations required to solve the decentralized problem. Fixing the scale would simplify the problem even further, requiring only 2 observations for each decentralized node solution. We should also emphasize that although in this example the decentralized solution was able to propagate through the entire network using the minimum number of leader

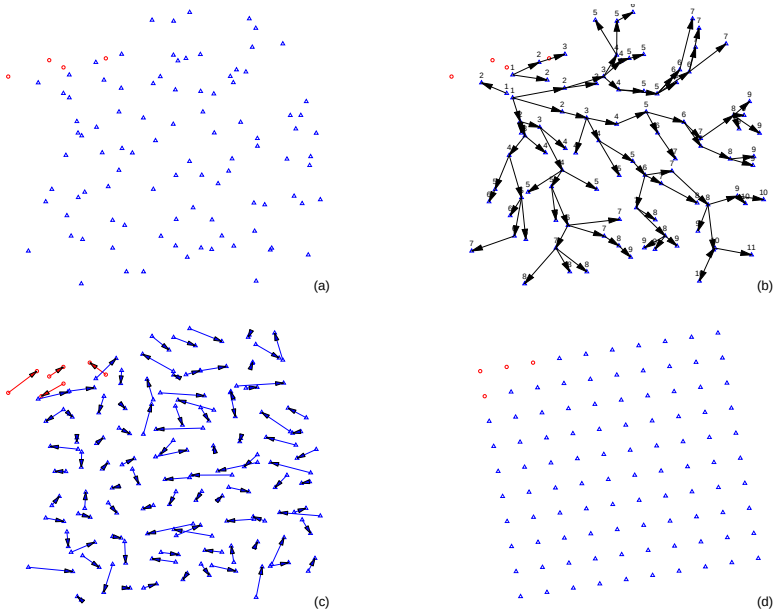

Fig. 5. Decentralized Motion Planning: (a) The initial network configuration with leader (red circle) and follower (blue triangle) nodes. (b) Evolution of the decentralized solution. (c) Node trajectories (d) Final network configuration achieving the desired {4,4} tessellation.

nodes, this will *not* typically be the case. More than likely, a small number of leader nodes will be associated with disjoint clusters in the network.

7 Discussion

In this paper, we developed a set of motion planning strategies suitable for large-scale sensor networks. These solve the problem of reconfiguring the network to a new shape while minimizing either the total distance traveled by the nodes or the maximum distance traveled by any node. The centralized approach runs in $O(m \log m)$ time in practice through banding the KKT system. The distributed approach reduces the expected per node workload to $O(\log m)$ in exchange for $O(\log m)$ messages per node in the overlay network. Finally, we derived a decentralized, hierarchical approach whereby follower nodes are able to solve for their objective positions in $O(1)$ time from observing the headings of a small number of leader nodes.

We are currently extending these results to a more general motion planning framework. To achieve this, issues such as collision/obstacle avoidance will have to be addressed. The latter is a particularly challenging task, as the presence of obstacles introduces concave constraints on the feasible set, and the resulting problem is no longer solvable as a SOCP. We hope that randomization and convex restriction techniques [9] will still allow the problem to be solved for real-time applications.

References

1. Compass: Collaborative multiscale processing and architecture for sensornetworks. http://compass.cs.rice.edu/.
2. Wavescope: An adaptive wireless sensor network system for high data-rate applications. http://wavescope.csail.mit.edu/.
3. Networking technology and systems (NeTS). NSF Solicitation 06-516, Dec 2005.
4. R. Bachmayer and N. E. Leonard. Vehicle networks for gradient descent in a sampled environment. In *Proc. IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec 2002.
5. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Unviersity Press, 2004.
6. J. Bruce and M. Veloso. Fast and accurate vision-based pattern detection and identification. In *IEEE International Conference on Robotics and Automation*, May 2003.
7. Z. Butler and D. Rus. Event-based control for mobile sensor networks. *IEEE Pervasive Computin*, 2(4):10–18, 2003.
8. J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. on Robotics and Automation*, 20(2):243–255, April 2004.
9. A. d'Aspremont and S. Boyd. Relaxations and randomized methods for nonconvex qcqps. Stanford University, 2003.
10. J. Derenick and J. Spletzer. TR LU-CSE-05-029: Optimal shape changes for robot teams. Technical report, Lehigh University, 2005.
11. I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.
12. R. G. et al. Tenet: An architecture for tiered embedded networks, November 10 2005.
13. J. T. Feddema, R. D. Robinett, and R. H. Byrne. An optimization approach to distributed controls of multiple robot vehicles. In *Workshop on Control and Cooperation of Intelligent Miniature Robots, IEEE/RSJ IROS*, Las Vegas, Nevada, October 31 2003.
14. A. Ganguli, S. Susca, S. Martínez, F. Bullo, and J. Cortés. On collective motion in sensor networks: sample problems and distributed algorithms. In *Proc. IEEE Conf. on Decision and Control*, pages 4239–4244, Seville, Spain, Dec. 2005.
15. G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
16. M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and Applications, Special Issue on Linear Algebra in Control, Signals and Image Processing*, 1998.
17. MOSEK ApS. *The MOSEK Optimization Tools Version 3.2 (Revision 8) User's Manual and Reference*. http://www.mosek.com.
18. A. Mourikis and S. Roumeliotis. Optimal sensing strategies for mobile robot formations: Resource constrained localization. In *Robotics: Science and Systems Conference*, pages 281–288, June 2005.
19. P. Ögren and N. Leonard. A tractable convergent dynamic window approach to obstacle avoidance. In *IEEE/RSJ IROS*, volume 1, Lausanne, Switzerland, October 2002.
20. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, 1992.
21. Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.
22. J. Spletzer and R. Fierro. Optimal positioning strategies for shape changes in robot teams. In *Proc. IEEE Int. Conf. Robot. Automat.*, 2005.
23. B. Zhang and G. S. Sukhatme. Controlling sensor density using mobility. In *The Second IEEE Workshop on Embedded Networked Sensors*, pages 141 – 149, May 2005.
24. F. Zhang, M. Goldgeier, and P. S. Krishnaprasad. Control of small formations using shape coordinates. In *Proc. IEEE Int. Conf. Robot. Automat.*, volume 2, Taipei, Sep 2003.