Efficient Motion Planning Strategies for Large-scale Sensor Networks

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

Department of Computer Science and Engineering, Lehigh University,
{jcd6,crm5,josa}@lehigh.edu

Abstract: 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 $O(m)$ time in practice using a logarithmic-barrier method. Key to this complexity is transforming the Karush-Kuhn-Tucker (KKT) matrix associated with the Newton step sub-problem into 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 $O(m)$ 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 lives 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 [2].

Also of fundamental importance to WSN research is resource management, and (perhaps most importantly) power management. 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, 13]. It would seem only natural 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.

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, topology changes can also be driven by performance objectives. For example, Cortes *et al* applied optimization based techniques to motion planning for improving network coverage [7]. Similarly, Zhang and Sukhatme investigated using motion to control node density [20]. The work of Hidaka *et al* investigated deployment strategies for optimizing localization performance [16], while the work of Butler and Rus was motivated by event monitoring using constrained resources [6]. Also worth noting is work in the areas of formation control [21], conflict resolution [17], and cooperative control [3]. A recent survey/tutorial outlining additional relevant work within each of these areas can be found in [11].

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 consider the problem of having a multi-agent team transition to a new shape formation while minimizing either the total distance or maximum distance metric. For our purposes, we adopt the traditional definition of shape that is often employed in statistical shape analysis [10]:

Definition 1. *The shape of a formation is the geometrical information that remains when location, scale, and rotational effects are removed.*

Thus, formation shape is invariant under the Euclidean similarity transformations of translation, rotation and scale [10].

For brevity, in this paper we only consider operations in $SE(2)$ and refer the reader to [9] for details on obtaining optimal solutions in $\mathbb{R}^3$. Letting $Q = [q_1, \ldots, q_m]^T \in \mathbb{R}^{m \times 2}$ denote the concatenated coordinates of the objective shape formation with respect to some world frame $\mathcal{W}$ and letting $S = [s_1, \ldots, s_m]^T \in \mathbb{R}^{m \times 2}$ denote an instance (or an *icon*) of our objective shape with respect to some local frame $\mathcal{F}$, the shape of a robot formation can be represented as the set of equality constraints:

$$
\begin{aligned}
q_i^x - q_1^x &= \alpha \left(s_i^x \cos \theta - s_i^y \sin \theta \right) \\
q_i^y - q_1^y &= \alpha \left(s_i^x \sin \theta + s_i^y \cos \theta \right)
\end{aligned}
\tag{1}
$$

for $i = 2, \ldots, m$. In this formulation, $\alpha \in \mathbb{R}_+$ and θ respectively denote the scale and orientation of the formation, while the (x, y) superscripts denote the specific Euclidean coordinate.

Without loss of generality, we can define the objective formation scale and orientation respectively as:

$$
\alpha = \frac{\| q_2 - q_1 \|}{\| s_2 - s_1 \|} = \frac{\| q_2 - q_1 \|}{\| s_2 \|} \qquad \theta = \arctan \frac{q_2^y - q_1^y}{q_2^x - q_1^x}
\tag{2}
$$

The former equalities hold as we choose $s_1 \stackrel{\triangle}{=} O_{\mathcal{F}}$. From the latter, we obtain:

$$
\cos \theta = \frac{q_2^x - q_1^x}{\| q_2 - q_1 \|} \qquad \sin \theta = \frac{q_2^y - q_1^y}{\| q_2 - q_1 \|}
\tag{3}
$$

Given these definitions, the non-convex constraints in (1) can be restated as the following set of linear equalities:

$$
\begin{aligned}
\| s_2 \| (q_i^x - q_1^x) - (s_i^x, s_i^y)^T (q_2 - q_1) &= 0, & i = 3, \ldots, m \\
\| s_2 \| (q_i^y - q_1^y) - (s_i^y, s_i^x)^T (q_2 - q_1) &= 0, & i = 3, \ldots, m
\end{aligned}
\tag{4}
$$

These constraints are now convex, and they 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. 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. $\max \| q_i - p_i \|$ is minimized for $i = 1, \ldots, m$ OR
2. $\sum_{i=1}^{k} \| q_i - p_i \|$ is minimized.

In other words, if the network were given an objective icon S, it must determine the objective positions for each node that minimizes the chosen metric, while ensuring the final shape formation $Q \subset \mathcal{W}$ is equivalent to S.

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

$$
\begin{array}{ll}
\min_{q,t_1} t_1 & \min_{q,t} \sum_{i=1}^{m} t_i \\
\text{s.t.} \ \| q_i - p_i \|_2 \leq t_1 & \text{s.t.} \ \| q_i - p_i \|_2 \leq t_i \\
\quad\quad Aq = 0 & \quad\quad Aq = 0
\end{array}
\tag{5}
$$

for $i = 1, \ldots, m$. 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, in the interest of brevity, the results presented in this paper largely focus on the *mini-max* distance problem as defined in Equation 5 (left). It should be noted that similar results have been obtained for the total distance variation [9].

4 A Centralized Approach

Centralized approaches are appropriate for hierarchical network architectures such as the TENET [13]. For the motion planning problem, "master" nodes acting as cluster-heads would calculate the objective positions for the cluster and communicate these to supporting nodes in the network. While simple in design, the hierarchy requires that algorithms 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 outlined in [4]. 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. As solving this system provides a solution to the Newton step sub-problem, we accordingly refer to it as the "Newton KKT system." We show that by reformulating the SOCP, we can band the coefficient 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 $O(1)$. The result is a simple IPM that in practice solves the motion planning (and similar) problems in $O(m)$ time.

4.1 Reformulating the Motion Planning Problem

The original *mini-max* motion planning problem can be restated in a relaxed form suitable for solving via the barrier approach. Conversion requires augmenting the objective function given in (5) with log-barrier terms corresponding to the problem's conic constraints as follows:

$$\min_{q,t_1} \tau_k t_1 - \sum_{i=1}^{m} \log\left(t_1^2 - (q_i - p_i)^T(q_i - p_i)\right)$$
$$\text{s. t. } Aq = 0 \tag{6}$$

where τ_k is the inverse log-barrier scaler for the kth iteration. Essentially, solving our SOCPs reduces to solving a sequence of convex optimization problems of this form, where after each iteration τ_{k+1} is chosen such that $\tau_{k+1} > \tau_k$.

4.2 Banding the Newton KKT System

During each iteration of the log-barrier approach, we aim to minimize the second-order Taylor approximation of our objective function as a function of the Newton step, δx, subject to $A\delta x = 0$. As a result, obtaining δx is equivalent to analytically solving the KKT conditions associated with this equality-constrained sub-problem. In other words, we must solve the following linear system of equations [4]:

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ w \end{bmatrix} = \begin{bmatrix} -g \\ 0 \end{bmatrix} \tag{7}$$

where H and g respectively denote the evaluated Hessian and gradient of the objective function given in (6) at x, w is the corresponding dual variable for δx, and A is as previously defined. Solving (7) is the bottleneck of the algorithm; however, we will show that it can be solved very efficiently (*i.e.* in $O(m)$ time) by simply reposing the problem given in (6).

Noting that the coefficient matrix of (7) is symmetric indefinite, we employ Gaussian elimination with non-symmetric partial pivoting. The performance of this technique suffers significantly when the linear system in question features dense rows and/or columns due to fill-in [19]. In particular, the algorithm could yield a worst-case performance of $O(m^3)$ when solving an instance of (7) associated with the nominal problem formulation given in (6). To illustrate this point, we include Figure 1 (left) which shows the corresponding non-zero sparsity structure (*a.k.a.* the dot-plot) of the Newton KKT system. As the rows of system are permuted during reduction, the dense rows and columns respectively located in the upper-right and lower-left quadrants of (7) could introduce a solid sub-block of order $m \times m$, which itself would require $O(m^3)$ basic operations to reduce. Such a workload is highly impractical, especially when considering large-scale configurations that inherently feature 1000's of decision variables.

To address this issue, we present the following auxiliary formulation of (6) that facilitates transforming the Newton KKT system into a mono-banded form:

$$\min_{q,t} \frac{\tau_k}{m} \sum_{i=1}^{m} t_i - \sum_{i=1}^{m} \log\left(t_i^2 - (q_i - p_i)^T(q_i - p_i)\right)$$
$$\text{s. t. } Aq = 0$$
$$t_{i+1} = t_i, \quad i = 1, \ldots, m-1$$
$$d_{2i+1} = d_{2i-1}, \quad i = 1, \ldots, m-3 \tag{8}$$
$$d_{2(i+1)} = d_{2i}, \quad i = 1, \ldots, m-3$$
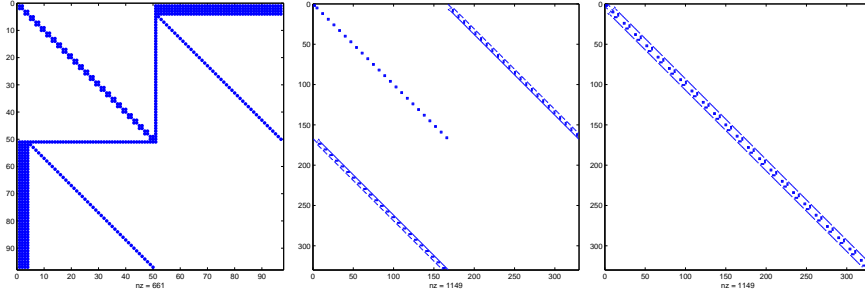$$d_i = q_i, \quad i \in \{1, 2\}$$

Fig. 1. (left) The nominal Newton KKT system sparsity structure for the *mini-max* motion planning problem in $SE(2)$. (center) Augmented Newton KKT system sparsity structure. (right) The banded system with lower and upper bandwidths of 8.

Notice that the objective has changed from (6); however, we see that both forms are equivalent since:

$$\frac{\tau_k}{m} \sum_{i=1}^{m} t_i = \frac{\tau_k}{m} \sum_{i=1}^{m} t_1 = \left(\frac{\tau_k}{m}\right) m t_1 = \tau_k t_1 \tag{9}$$

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

Given this augmented formulation, our claim is that the system can be made mono-banded. To show this, we begin by defining the nominal solution vector for the coefficient structure of (7) as follows:

$$\left[\delta\eta_1^T, \delta\eta_2^T, \delta\kappa_1^T, \ldots, \delta\kappa_{(m-2)}^T, \mu^T\right]^T \tag{10}$$

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

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

Given the new objective, (9), and assuming the shape problem's solution vector permutation corresponds with (10), the Hessian for our problem is now

$$H = \begin{bmatrix} \psi_1 & \cdots & \cdots & 0 \\ \vdots & \psi_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \psi_m \end{bmatrix} \quad \begin{aligned} & \psi_i = \nabla^2\phi(u_i, t_i), \ i \in \{1, 2\} \\ & \psi_i = \begin{bmatrix} 0_{4\times 4} & 0_{4\times 3} \\ 0_{3\times 4} & \nabla^2\phi(u_i, t_i) \end{bmatrix}, \ i \in \{3, \ldots, m\} \end{aligned} \tag{11}$$

where $\nabla^2\phi(u_i, t_i)$ is defined as in [14] with $u_i = q_i - p_i$. Notice that this Hessian is block-diagonal and separable. This differs from the its nominal form, which features a dense row and column corresponding to the variable, t_1. This is evident by

observing the upper-left quadrant (defined by H) of the KKT matrix given in Figure 1 (left).

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

$$
\begin{aligned}
\| s_2 \| (q_i^x - d_j^x) - (s_i^x, s_i^y)^T(d_{j+1} - d_j) = 0 \\
\| s_2 \| (q_i^y - d_j^y) - (s_i^y, s_i^x)^T(d_{j+1} - d_j) = 0
\end{aligned}
\tag{12}
$$

for $i = 3, \ldots, m$ 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 and exhibits a band-like structure.

To show this, we begin by stating the constraint/row permutation that yields A in *pseudo-banded* form. We define the constraints associated with q_1 and q_2 as:

$$
\begin{aligned}
\varrho_1 &\triangleq q_1^x = d_1^x & \varrho_4 &\triangleq q_2^x = d_2^x \\
\varrho_2 &\triangleq q_1^y = d_1^y & \varrho_5 &\triangleq q_2^y = d_2^y \\
\varrho_3 &\triangleq t_1 = t_2
\end{aligned}
\tag{13}
$$

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

$$
\begin{aligned}
\varphi_{i_1} &\triangleq \| s_2 \| (q_i^x - d_j^x) = (s_i^x, s_i^y)^T(d_{j+1} - d_j) & \varphi_{i_4} &\triangleq d_{j+2}^x = d_j^x \\
\varphi_{i_2} &\triangleq \| s_2 \| (q_i^y - d_j^y) = (s_i^y, s_i^x)^T(d_{j+1} - d_j) & \varphi_{i_5} &\triangleq d_{j+2}^y = d_j^y \\
\varphi_{i_3} &\triangleq t_i = t_{i-1} & \varphi_{i_6} &\triangleq d_{j+3}^x = d_{j+1}^x \\
& & \varphi_{i_7} &\triangleq d_{j+3}^y = d_{j+1}^y
\end{aligned}
\tag{14}
$$

where j is as previously defined.

With q_m, we associate the remaining three constraints:

$$
\begin{aligned}
\varphi_{m_1} &\triangleq \| s_2 \| (q_m^x - d_j^x) = (s_m^x, s_m^y)^T(d_{j+1} - d_j) \\
\varphi_{m_2} &\triangleq \| s_2 \| (q_m^y - d_j^y) = (s_m^y, s_m^x)^T(d_{j+1} - d_j) \\
\varphi_{m_3} &\triangleq t_m = t_{m-1}
\end{aligned}
\tag{15}
$$

where j is as previously stated with $i = m$.

Given these definitions, we provide the following row permutation for A, which yields the *pseudo-banded* form that appears in the lower-left (and upper-right) quadrant of Figure 1 (center):

$$
\left[\vartheta^T, \varkappa_1^T, \ldots, \varkappa_{(m-1)}^T, \varsigma^T \right]^T
\tag{16}
$$

$$
\vartheta = \begin{bmatrix} \varrho_1 \\ \varrho_2 \\ \vdots \\ \varrho_5 \end{bmatrix} \quad \varkappa_i = \begin{bmatrix} \varphi_{(i+2)_1} \\ \varphi_{(i+2)_2} \\ \vdots \\ \varphi_{(i+2)_7} \end{bmatrix} \quad \varsigma = \begin{bmatrix} \varphi_{m_1} \\ \varphi_{m_2} \\ \varphi_{m_3} \end{bmatrix}
$$

Notice that all of the primal constraints defined in (8) have been included.

Given the definitions of A and H, the mono-banded form of (7) can now be constructed. Symmetrically applying the permutation that yields the following Newton KKT system solution vector ordering:

$$\left[\lambda^T, \xi_1^T, \ldots, \xi_{(m-3)}^T, \chi^T\right]^T \tag{17}$$

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

produces a mono-banded coefficient structure having a total bandwidth of 17. Simply using the standard Reverse Cuthill-McKee (RCM) reordering algorithm [8] would yield a bandwidth 47% larger than that obtained with our approach.

In Figure 1 (center), we show the "augmented" Newton KKT system constructed from the Hessian given by (11) and the linear constraint set given in (8). The latter is permuted according to (16). Taking the coefficient structure of (7) in this form and symmetrically permuting its rows and columns according to (17) 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 solved in $O(m)$ using a band-diagonal LU-based solver [18].

4.3 Total Complexity

Assuming a fixed duality gap reduction, the iteration complexity of the barrier approach grows as $O(\sqrt{m})$ [4]. Noting that the per-iteration complexity is defined by solving the mono-banded Newton KKT system as well as computing banded matrix-vector products, the total number of basic operations required to achieve optimality grows only as $O(m^{1.5})$. The generality of this result should not go overlooked as the bound applies to any SOCP that yields a mono-banded Newton KKT system. This includes regulated cases of the motion planning problem in $\mathbb{R}^3$ [9].

4.4 Performance in Practice

To gauge the performance of the framework in practice, we assumed both a fixed duality gap reduction and barrier parameter μ as suggested in [4]. A total of 5,000 random instances of the motion planning problem were solved using an implementation of the barrier algorithm. The objective was to minimize the total distance traveled by the team. Values of m were considered between 10 and 1000 at intervals of 10, where m denotes configuration size. Our implementation was validated by comparing obtained solutions against those of the MOSEK industrial solver [15]. All problems
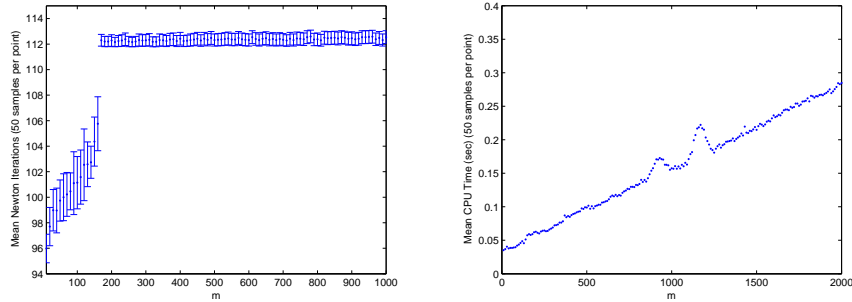
Fig. 2. (left) The mean number of Newton iterations required to solve the motion planning problem (total distance metric) as a function of configuration size, m. For $m \gtrsim 170$ the number of iterations appears constant using the log-barrier approach. Error bars signify a single standard deviation. (right) The mean CPU-utilization time required to solve the motion planning problem using the MOSEK solver. The trend is strongly linear, with $r^2 = 0.9869$.

were solved using a standard desktop PC having a 3.0 GHz Pentium 4 processor and 2.0 GB of RAM.

Figure 2 (left) shows the results of these trials. Each data point corresponds to the mean of 50 samples with the error bars corresponding to a single standard deviation. The trend indicates that the total number of Newton iterations remains constant (for $m \gtrsim 170$). This result in tandem with the linear per-iteration complexity established earlier shows that in practice the motion planning problem is solvable in $O(m)$ time.

These results are associated with the simple barrier method outlined in [4]; however, it should be noted that empirical results show that similar performance can be achieved using more sophisticated solvers. Figure 2 (right) shows the CPU time required by the MOSEK industrial solver for configuration sizes having up to 2000 nodes. Each point corresponds to the mean obtained from solving 50 randomly generated motion planning SOCPs (total distance metric). This data shows that for a configuration of 2000 nodes in $SE(2)$ an optimal solution can be obtained in only 0.28 seconds. Furthermore, the CPU time clearly scales as $O(m)$ with linear regression analysis revealing $r^2 = 0.9869$, where r is the associated correlation coefficient. Results obtained using the solver also indicate that an optimal solution can typically be found in less than 12 iterations - regardless of configuration size.

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 $O(1)$ expected per-node workload will 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 cluster-based network architecture 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 is complete. Although the *root* maintains a "global perspective", 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 needed to manage the IPM solve process. For instance, when it requests 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 originating from the *root* before transitioning into a new state of computation.

To reduce the communication overhead, 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 Newton KKT System

Given the objective function and Hessian are separable, implementing a distributed Newton decrement and line search computation (see [4]) 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 is defined by solving the Newton 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 computation, storage, and overlay message complexities of $O(1)$.

To distribute the Newton KKT system, $K \in \mathbb{R}^{y \times y}$, among m nodes, 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 $K_c \in \mathbb{R}^{y \times (b_l + b_u + 1)}$ [18]. We respectively denote the corresponding right-hand-side and permutation vectors as $b \in \mathbb{R}^y$ and $p \in \mathbb{Z}_+^y$.

Adopting this representation for K, we adapt the LU-based solver with partial pivoting outlined in [18]. Distributing this algorithm, we begin by assigning the ith node, n_i, a sub-block $K_c^i \subset K_c$. Additionally, each n_i manages the corresponding sub-vectors $b_i \subset b$ and $p_i \subset p$. To illustrate the decomposition, we provide Figure 3, which shows the distribution of K_c for a team of 5 nodes in $SE(2)$ solving the total distance problem. Given the dependencies between the equations in the linear
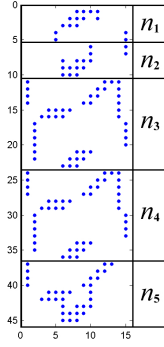
Fig. 3. A non-zero dot-plot illustrating the decomposition of the compact Newton KKT system (i.e. K_c) for a configuration of 5 nodes in $SE(2)$ minimizing the total distance metric. For this problem, $b_l = b_u = 7$. Notice that the middle $(m - 3)$ nodes (i.e. n_3 and n_4) are assigned sub-blocks with identical structure.

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.

Decomposition

During decomposition, 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 can 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. With this result in mind, we offer the following theorem:

Theorem 1. *Let* $i \in \mathcal{I} = \{1, \ldots, m\}$ *and let* $K_c^j \in \mathbb{R}^{u_j \times (b_l + b_u + 1)}, u_j \in \mathbb{Z}_+$ *for* $j = 1, \ldots, m$. *Define* $\psi(i) : \mathcal{I} \to \mathbb{Z}_+$ *as a mapping to the number of nodes that have to be contacted by* n_i *during the decomposition of* K_c^i. *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. $\to\leftarrow$

Using the data it acquired from its $\psi(i) \leq \phi$ supporting peers (in particular, $n_{i+1}, \ldots, n_{i+\psi(i)}$), n_i performs standard LU decomposition on K_c^i. Upon completion, it sends an update to each of the supporting nodes. The update contains the modified row(s) information and the adjusted permutation vectors corresponding to the changes it made with respect to the data the recipient provided. This allows each supporting node to update its cache before the process is handed off to n_{i+1}.

Forward and Backward Substitution

Similar to decomposition, both forward and backward substitution are done in an iterative manner. In both cases, the active node, n_i, will have to communicate with a small (bounded) number of its peers. During forward substitution, it will have to acquire information from each of the $\psi(i)$ nodes that provided it with data during the decomposition. This differs from the backward substitution phase, which may require n_i to communicate with up to 2ϕ nodes. The additional messaging is introduced via the upper triangular factor, U, having a bandwidth now constrained by $(b_u + b_l + 1)$, which is latent to the use of partial pivoting [12]. Since n_m is the last node to perform both decomposition and forward substitution, it is responsible for signaling the start of a new phase in the LU-solver process.

Message and Storage Complexity

For simplicity, the assumption is made that whenever n_i requests information from any node, the data is received in a single message. This assumption is reasonable, because the amount of information (including row 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 information is delivered upon request, the total number of messages sent by n_i is:

$$O(2\psi(i) + 2\psi(i) + 4\psi(i) + \gamma(i)) \equiv O(8\phi + 3) \equiv O(1) \tag{18}$$

where $\gamma(i) \leq 3$ is a mapping to the number of hand-off/signal messages sent by n_i.

As all nodes send $O(1)$ messages during the solve, the aggregate message complexity for the distributed LU process is $O(m)$. Recalling that the number of Newton iterations will be $O(1)$ in practice, we expect that no more than $O(m)$ messages will be generated in the overlay network. Furthermore, since n_i manages some fixed-size K_c^i, b_i, p_i, and row data received by as many as 2ϕ peers, per-node storage is $O(1)$.

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. The objective was to minimize the total distance traveled by team members. Each Aibo was outfitted with a unique butterfly pattern [5] that was tracked via an overhead camera system serving as an indoor "GPS". Figure 4 (left) shows the initial configuration, along with lines mapping each to its computed optimal position, while Figure 4 (right) shows the Aibos after transitioning to the optimal shape configuration.
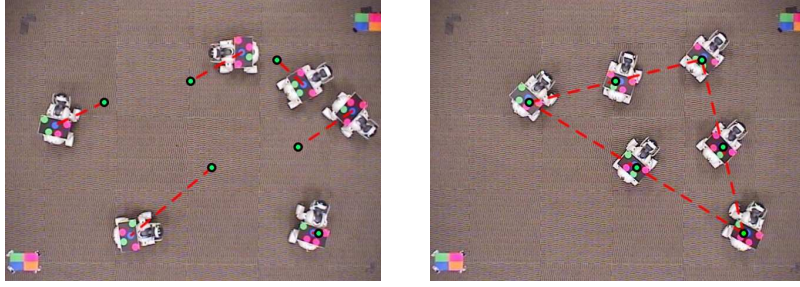
Fig. 4. (left) An initial dispersion of 6 Aibos, along with overlaid lines/points mapping each to its computed optimal position. (right) The Aibos after reconfiguring to the desired delta shape formation. All computations were done in a distributed fashion, with each dog being responsible for computing its optimal position and local control inputs.

6 A Decentralized Hierarchical Approach

For our decentralized approach, we assume a hierarchical model 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 [13], but also models where minimizing data communication is a primary objective [1]. 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 neighbors.

6.1 An $O(1)$ Decentralized Solution

Key to this approach is the realization that although the optimization problem in (5) 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 knowledge, 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 in

(4). Perhaps more significant is that the problem can now be solved by the follower nodes in a 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 (4). 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

$$
\begin{aligned}
(q_l - p_l)^T (\sin \omega_l, -\cos \omega_l) = 0, & \quad l = 2, \ldots, 5 \\
\| s_2 - s_1 \| (q_l^x - q_1^x) - (s_l^x, s_l^y)^T (q_2 - q_1) = 0, & \quad l = 3, \ldots, 5 \\
\| s_2 - s_1 \| (q_l^y - q_1^y) - (s_l^y, s_l^x)^T (q_2 - q_1) = 0, & \quad l = 3, \ldots, 5
\end{aligned}
\tag{19}
$$

where $l \in \{2 \ldots 5\}$ 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 in Figure 5.

6.2 Simulation Results

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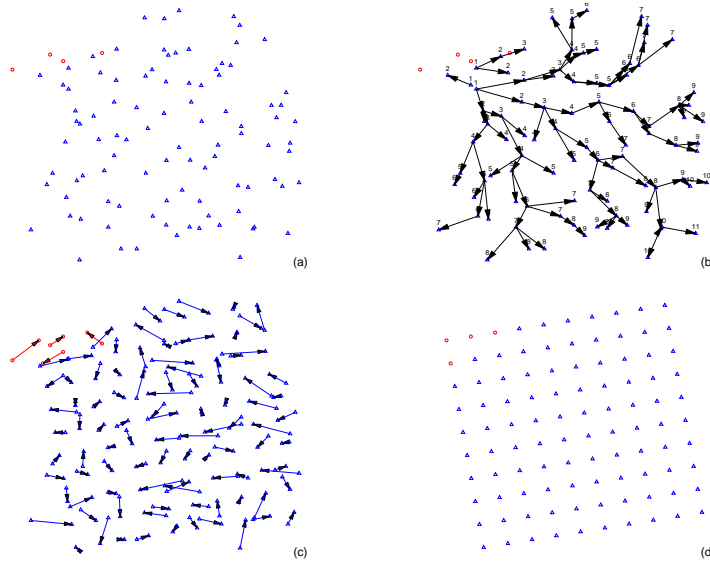
Fig. 5. Decentralized Motion Planning: (a) The initial network configuration with leader (red circle) and follower (blue triangle) nodes. (b) Evolution of the decentralized solution. (c) Node trajectories (d) Final network configuration achieving the desired {4,4} tessellation.

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 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 $O(m)$ time in practice through banding the Newton KKT system. The distributed approach reduces the expected per-node workload to $O(1)$ in exchange for $O(1)$ 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 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. Networking technology and systems (NeTS). NSF Solicitation 06-516, Dec 2005.
3. 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.
4. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Unviersity Press, 2004.
5. J. Bruce and M. Veloso. Fast and accurate vision-based pattern detection and identification. In *IEEE International Conference on Robotics and Automation*, May 2003.
6. Z. Butler and D. Rus. Event-based control for mobile sensor networks. *IEEE Pervasive Computing*, 2(4):10–18, 2003.
7. 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.
8. E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, New York, USA, 1969.
9. J. Derenick and J. Spletzer. TR LU-CSE-05-029: Optimal shape changes for robot teams. Technical report, Lehigh University, 2005.
10. I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.
11. 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.
12. G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
13. R. Govindan et al. Tenet: An architecture for tiered embedded networks. Technical report, Center for Embedded Networked Sensing (CENS), Nov 2005.
14. 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.
15. MOSEK ApS. *The MOSEK Optimization Tools Version 3.2 (Revision 8) User's Manual and Reference*. http://www.mosek.com.
16. A. Mourikis and S. Roumeliotis. Optimal sensing strategies for mobile robot formations: Resource constrained localization. In *Robotics: Science & Sys.*, pages 281–288, Jun 2005.
17. P. Ögren and N. Leonard. A tractable convergent dynamic window approach to obstacle avoidance. In *IEEE/RSJ IROS*, volume 1, Lausanne, Switzerland, October 2002.
18. W. Press et al. *Numerical Recipes in C*. Cambridge University Press, 1993.
19. Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.
20. 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.
21. 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.