European Journal of Control (2005)11:436–463

European Journal of Control

Sensor Networks and Cooperative Control

Christos G. Cassandras and Wei Li*

Department of Manufacturing Engineering, Center for Information and Systems Engineering, Boston University, Brookline, MA 02446, USA

This paper provides a tutorial-style overview of sensor networks from a systems and control theory perspective. We identify key sensor network design and operational control problems and present solution approaches that have been proposed to date. These include deployment, routing, scheduling and power control. In the case of mobile nodes, a sensor network is called upon to perform a "mission". We present solution approaches to two types of missions both involving stochastic mission spaces and cooperative control: reward maximization missions and coverage control missions. We conclude by outlining some fundamental research questions related to sensor networks and the convergence of communication, computing and control.

Keywords: Cooperative Control; Distributed Control; Optimization; Sensor Networks

1. Introduction

A sensor network consists of a collection of (possibly mobile) sensing devices that can coordinate their actions through wireless communication and aim at performing tasks, such as exploration, surveillance or monitoring and tracking "target points" over a specific region, often referred to as the "mission space". Collected data are then further processed and often support higher-level decision making processes. Nodes in such networks are generally inhomogeneous, they have limited on-board resources (e.g., power and computational capacity), and they may be subject to communication constraints. It should be pointed out that sensor networks differ from conventional communication networks in a number of critical ways. First, they allow us to interact with the *physical* world, not just computers, databases or human-generated data. By inserting decision making and control functionality into such networks one can envision closing the loop on remote processes that would otherwise be inaccessible. Thus, sensor networks are expected to realize a long-anticipated convergence of communication, computing and control [6]. Second, at least some nodes in such a network are "active", e.g., they execute sensing processes or they are are mobile; therefore, they are characterized by dynamics making a sensor network as a whole a challenging dynamic system. In addition, nodes are typically small and inexpensive, operating with limited resources, often in adverse stochastic environments. This implies that optimization in designing and operating sensor networks is a real need and not a mere luxury. Moreover, the limited computational capabilities of nodes often make *distributed* control or optimization methods indispensable. Finally, when it comes to measuring the performance of sensor networks, the metrics can be quite different from those used in standard communcation networks, giving rise to new types of problems. For example, because of limited energy, we recognize that nodes have finite lives and we often

*E-mail: wli@bu.edu
Correspondence to: C.G. Cassandras. E-mail: cgc@bu.edu

Received 1 May 2005; Accepted 15 June 2005.
Recommended by E.F. Camacho, R. Tempo, S. Yurkovich, P.J. Fleming.

seek control mechanisms that maximize an appropriately defined "network lifetime". Part of such mechanisms may involve switching nodes on and off so as to conserve their energy or finding means to periodically replenish their energy supply. When the nodes are mobile, mechanisms are also needed to determine desired trajectories for the nodes over the mission space and *cooperative control* comes into play so as to meet specific mission objectives.

The main goal of this paper is to provide a tutorial-style overview of sensor networks from a systems and control theory perspective. To do so, we identify key sensor network design and operational control problems and discuss some solution approaches that have been proposed to date. Emphasis is placed on rigorously formulating interesting optimization and cooperative control problems, describing the essential points of solution approaches without getting into technical details whenever these may be found in cited references. The last part of the paper addresses the "coverage control" problem, which we concentrate on because the formulation and solution presented constitute a new contribution and because this particular problem captures the main features and control challenges encountered in sensor networks: the need to define network performance in an unconventional manner, the involvement of cooperative control, the computational limitations of nodes that require a distributed control solution and the role of communication constraints. Simulation results are presented throughout the paper to illustrate various control schemes and algorithms and open research problem are identified.

The rest of the paper is organized as follows. Section 2 describes the basic structure of sensor networks and classifies them in a way that distinguishes between (1) those with fixed, known data sources and nodes that are not mobile, and (2) those where data sources may be unknown and nodes are mobile. In Section 3, we discuss the main problems related to the first network type, including deployment, power control, routing and scheduling. In Section 4, we consider networks with mobile nodes that are called upon to perform a "mission". Different types of missions lead to defining different types of problems and we present two such problems, both involving stochastic mission spaces and cooperative control: reward maximization missions and coverage control missions. In Section 5, we conclude by outlining some fundamental research questions related to sensor networks and the convergence of communication, computing and control, as well as some more specific issues that developments to date have brought forth.

2. Sensor Network Structure

In its most basic form, the main objective of a sensor network is to collect field data from an observation region (the "mission space"), denoted by $\mathcal{R}$, and route it to a *basestation*, denoted by B (also referred to as "data collection point" or "sink"). At any time instant, there may exist multiple data sources in $\mathcal{R}$ (also referred to as "target points" or simply "targets"). Nodes in a sensor network collaborate to ensure that every source is sensed and that the data gathered are successfully relayed to the basestation. During cooperation, a sensor node may fall into one of the following states: (1) *Sensing*: a sensing node monitors the source using an integrated sensor, digitizes the information, processes it and stores the data in its onboard buffer. These data will be eventually sent back to the basestation. (2) *Relaying*: a relaying node receives data from other nodes and forwards it towards their destination. (3) *Sleeping*: for a sleeping node, most device functions are either shut down or work in low-power mode. A sleeping node does not participate in either sensing or relaying. However, it "wakes up" from time to time and listens to the communication channel in order to answer requests from other nodes. Upon receiving a request, a state transition to "sensing" or "relaying" may occur. (4) *Dead*: a dead node is no longer available to the sensor network. It has either used up its energy or has suffered vital damage. Once a node is dead, it cannot re-enter any other state.

Instead of a *flat* structure, some sensor networks assume a more *hierarchical* one. In this case, besides sensors and a basestation, there are also nodes acting as *clusterheads*. These nodes generally have more powerful data processing and routing capabilites, at the expense of size and cost. Each clusterhead is in charge of a *cluster* of sensor nodes, which is obtained by making a spatial or logical division of the network. By aggregating the data sent from sensor nodes, a clusterhead refines the observation of the cluster's region. Then, it may produce some post-processed data and route them to the basestation. The links connecting clusterheads and the basestation may have larger data rate in order to support high-speed data transmission. Figure 1 illustrates this 3-layer structure with sensor nodes (S), routers (R) and a basestation (B).

In most current applications, sensor networks have low data rates. According to the IEEE 802.15.4 standard, the transmission capacity of each channel is 20 kb/s (868 MHz channel), 40 kb/s (915 MHz) or 250 kb/s (2.4 GHz). The data rate required for sensing may be much lower, which permits a sensor node to act as a data-source and a relay at the same time. Current

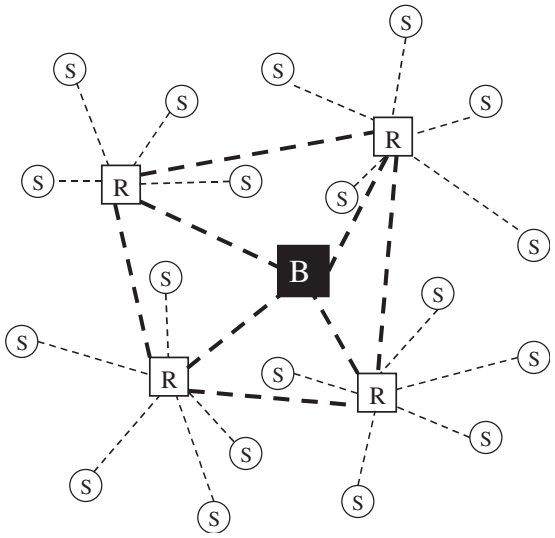

Fig. 1. A 3-layer sensor network structure.

sensor networks limit the transmission distance between nodes to <100 m (typically, 5–10 m depending on the environment) and the spatial density of nodes over a mission space varies between 0.1 and 20 nodes/m^2.

Let us start by considering sensor networks where all data sources are fixed and the nodes are not mobile. Then, the first and most basic problem we face is that of *deployment*, i.e., positioning the nodes so as to meet the goal of successfully transferring data from the sources to the basestation and, ultimately, to optimize some network performance metric. Once this is accomplished, there are numerous operational control issues at different layers (physical, data-link, network, transport, etc.); for a comprehensive overview, see Refs [3,10]. We shall limit ourselves here to some of the most important problems where dynamic control-oriented methodolgies may be used and precise mathematical formulations are possible. In particular, each node must, at a minimum, perform three tasks: (1) *Routing*, i.e., determining the destination node of data packets transmitted from some node i on their way to the basestation; (2) *Scheduling*, i.e., determining the precise timing mechanism for transmitting packets of possibly different types; and (3) *Power control*, i.e., making decisions aimed at conserving its energy in a way that benefits the network as a whole. In the next section, we will discuss these problems in some more detail.

The second class of sensor networks we will consider is one where nodes are mobile. This allows a sensor network to carry out missions such as visiting data source targets (so that the number of mobile nodes may be much smaller than that of the targets sought), searching for unknown data sources, and tracking targets that may be mobile as well. In this case, the role of cooperative control becomes essential, as nodes must collaborate to meet mission objectives with limited resources and under possibly critical time constraints. In Section 4.1, we will discuss a basic mission that requires cooperation among mobile nodes in visiting targets assigned different (possibly time-varying) rewards; the goal of the mission is to maximize the total reward that can be collected. In Section 4.2, we consider a deployment problem for mobile sensor networks, which, in this case, is commonly referred to as the *coverage control* problem. We will present in some detail a recently developed solution for this problem.

3. Networks with Fixed Data Sources and No Mobility

3.1. The Deployment Problem

The deployment of sensor nodes may be either deterministic or random. The latter situation arises in applications, such as reconnaissance and exploration, where sensors are randomly dropped into the mission space and their exact location cannot be precisely controlled. In this case, research focuses on the relationship between sensor density and network performance. In Ref. [25], an analysis is provided of how critical communication power scales with the size of the network under connectivity constraints, thus indicating the relationship between lifetime and density of the network. This work was extended in Ref. [63] by also considering the failure rate of a sensor node. More recent work [49] studies the deployment of heterogeneous sensors (i.e., normal nodes and clusterheads) and derives optimal node densities and energies that guarantee a desired lifetime, while ensuring connectivity and coverage.

Deterministic deployment takes place when the characteristics of the mission space are known in advance (e.g., in building monitoring). Fundamental questions in this case include (1) How many sensor nodes are needed to meet the overall system objectives? (2) For a given network with a certain number of sensor nodes, how do we precisely deploy these nodes in order to optimize network performance? (3) When data sources change or some part of the network malfucntions, how do we adjust the network topology and sensor deployment? These questions can be resolved by an off-line scheme which is akin to the widely studied facility location optimization problem. One of the commonly applied approaches is to dis-

cretize the mission space and place sensor nodes along grid points. The resulting optimal deployment problem can be formulated as a linear program. Since all grid points and interconnected links must be considered, this results in significant combinatorial complexity. Alternatively, one can formulate a nonlinear optimization problem and seek to exploit the structure of a sensor network in order to develop decomposition approaches to solve it. In what follows, we review such an approach introduced in Ref. [40].

Adopting the source/basestation structure of a sensor network discussed earlier, we consider M data sources residing at points $s_m \in \mathbb{R}^2$ $(m = 1, \ldots, M)$ and a single basestation B, with location $x_0 \in \mathbb{R}^2$. Each data source has a fixed position and a given data rate denoted by r_m $(m = 1, \ldots, M)$. To collect data at each data source, a sensor must be deployed at its location. In addition, since a data source may be far from the basestation and the distance may exceed the range of radio communication, we also need to deploy a certain number of sensor nodes that work as relays. Suppose there are N active sensor nodes and each has location $x_k \in \mathbb{R}^2$ $(k = 1, \ldots, N)$. Let $\mathcal{W} = (\mathcal{V}, \mathcal{E}, \mathbf{c}, \mathbf{e})$ be a flow network with underlying directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of links. A *capacity vector* $\mathbf{c} = [c_1, \ldots, c_{|\mathcal{E}|}]$ and a *cost vector* $\mathbf{e} = [e_1, \ldots, e_{|\mathcal{E}|}]$ are defined on every link $j \in \mathcal{E}$ with $c_j, p_j \in \mathbb{R}^+$. Each link j starts at node $s(j)$ and ends at node $t(j)$ and e_j denotes some cost metric per unit of data, which generally depends on the node locations. Over this flow network, we can formulate an optimization problem that minimizes the total cost by controlling on each link j the *locations of sensor nodes* $x_{s(j)}$ and $x_{t(j)}$ and the *data rate* f_j^m from each source $m = 1, \ldots, M$:

$$\min_{x_i, \mathbf{f}^m} \sum_{m=1}^{M} \sum_{j \in \mathcal{E}} e_j(x_{s(j)}, x_{t(j)}) f_j^m \tag{1}$$

$$\text{s.t.} \sum_{j \in \mathcal{E}} a_{ij} f_j^m = -r_m d_i^m \quad i \in \mathcal{V},\ m = 1, \ldots, M \tag{2}$$

$$\sum_{m=1}^{M} f_j^m \le c_j \quad j \in \mathcal{E} \tag{3}$$

$$f_j^m \ge 0 \quad j \in \mathcal{E},\ m = 1, \ldots, M \tag{4}$$

$$x_m = s_m \quad m = 1, \ldots, M \tag{5}$$

In (1), the decision variables are $\mathbf{f}^m$ and x_i, where a component f_j^m of the flow vector $\mathbf{f}^m$ denotes the data rate on link j $(j \in \mathcal{E})$ that originates from source m $(m = 1, \ldots, M)$. In the flow balance equations (2), $\mathbb{A} = \{a_{ij}\}$ is the node-link incidence matrix of graph $\mathcal{G}$ such that for all $i = 1, \ldots, N$ and $j = 1, \ldots, |\mathcal{E}|$ [57]:

$$a_{ij} = \begin{cases} +1 & \text{if arc } j \text{ leaves node } i \\ -1 & \text{if arc } j \text{ enters node } i \\ 0 & \text{otherwise} \end{cases}$$

and $\mathbf{d}^m = [d_i^m]$ is the flow balance vector for data source m such that

$$d_i^m = \begin{cases} -1 & i = 0 \\ +1 & i = m \\ 0 & \text{otherwise} \end{cases}$$

The remaining three equations represent the link capacity constraints, flow non-negativity and the fixed locations of the M sources.

Although this formulation is general, we shall consider our objective to be the determination of a *minimum power deployment*. In this case, the link cost $e_j(x_{s(j)}, x_{t(j)})$ denotes the energy consumed per unit of data. The function $e_j(x_{s(j)}, x_{t(j)})$ can be specified based on an energy model whose key parameters are the energy needed to sense a bit (E_{se}), receive a bit (E_{rx}) and transmit a bit over a distance d(E_{tx}). A $1/d^n$ $(n \ge 1)$ path loss is commonly assumed [4], in which case we can write

$$E_{tx} = \alpha_{11} + \alpha_2 d^n, \quad E_{rx} = \alpha_{12}, \quad E_{se} = \alpha_3 \tag{6}$$

where α_{11} is the energy/bit consumed by the transmitter electronics, α_2 accounts for energy dissipated in the transmit op-amp, α_{12} is the energy/bit consumed by the receiver electronics and α_3 is the energy cost of sensing a bit. Hence, the energy consumed by a node acting as a relay that receives a bit and then transmits it a distance d onward is

$$e(d) = \alpha_{11} + \alpha_2 d^n + \alpha_{12} \equiv \alpha_1 + \alpha_2 d^n \tag{7}$$

Typical numbers for current radios are $\alpha_1 = 180\,\text{nJ/bit}$ and $\alpha_2 = 10\,\text{pJ/bit/m}^2$ $(n = 2)$ or $0.001\,\text{pJ/bit/m}^4$ $(n = 4)$ [29]. Therefore, in (1), we have

$$\begin{aligned} e_j(x_{s(j)}, x_{t(j)}) &= e(||x_{s(j)} - x_{t(j)}||) \\ &= \alpha_1 + \alpha_2 ||x_{s(j)} - x_{t(j)}||^n \end{aligned} \tag{8}$$

for each link $j \in \mathcal{E}$ that starts at node $s(j)$ and ends at node $t(j)$. A property of $e_j(\cdot)$ as formulated in (8) is that it is a convex function of both $x_{s(j)}$ and $x_{t(j)}$. The solution of the deployment problem is in fact robust with respect to the specific form of $e_j(\cdot)$, as long as the convexity property is preserved.

In Ref. [4], a minimum-power topology was pro-

posed based on the assumption that there is no constraint on the number of intermediate sensor nodes. Under this assumption, the most energy-efficient path between a data source and the sink is a straight line with multiple hops, and the minimum-power topology is constructed by building such a path for each data source in the network. This topology consumes the least power since each data flow r_m takes the shortest path toward the sink and by optimizing the number of intermediate nodes, the power consumption on this shortest path is also minimized. The theoretical optimal number of hops, K_{opt}, including the node that acts as sensor at a data source s, is given by $K_{\text{opt}} = D/d_{\text{char}}$ where $D = \|s - b\|$ is the distance between s and the basestation b and d_{char} is the "characteristic distance" given by

$$d_{\text{char}} = \sqrt[n]{\frac{\alpha_1}{\alpha_2(n-1)}}$$

where α_i, n are defined by the node energy model (6). The corresponding lower bound for power consumption between some source s_m and the basestation is

$$P_m = \left(\alpha_1 \frac{n}{n-1}\frac{D_m}{d_{\text{char}}} - \alpha_{12}\right) r_m + \alpha_3 r_m \tag{9}$$

where $D_m = \|s_m - b\|$. However, in constructing this minimum-power topology, a large number of relays is needed, since each data flow is independent and shares no relays with the rest. When the number of nodes is limited, a natural idea is to minimize the power consumption by (1) making two or more data flows share some relays or (2) deploy fewer relays on some route. This brings us back to the minimum-power sensor deployment problem (1), which couples two traditional optimization problems: if flow vectors $\mathbf{f}^m$ are given and (1) is optimized only over the locations of sensors x_i, (1) can be viewed as a facility location problem; however, if sensor locations x_i are given and $\mathbf{f}^m$ are the only decision variables, it can be reduced to a minimum-cost flow problem. The nonlinearity of the cost function as well as the coupling of these two problems make (1) difficult to solve. For example, we have found that using standard Lagrangian relaxation methods does not substantially reduce complexity because this coupling is tight.

As an alternative, the solution approach proposed in Ref. [40] uses a decomposition method exploiting two facts: (1) the convexity of the link costs $e_j(x_{s(j)}, x_{t(j)})$ and (ii) the fact that in a sensor network data traffic always flows from the sources towards the basestation, which allows us to reduce the feasible space of $\mathbf{f}^m$ by only considering flow vectors that form a *tree structure* over the network. In addition, we also relax the capacity constraint (3); current sensor networks indeed operate with light traffic and the actual data flow over a link is unlikely to reach the link's capacity. When this capacity is not reached, it is also easy to see that no links other than those in a tree structure are ever used [57] (if any such link is used, the distance to the sink is increased; hence, the power consumption increases as well).

3.1.1. Problem Decomposition

The proposed decomposition method is motivated by the special structure of the problem. Since the cost $e_j(\cdot)$ of link j is a convex function of the location of its end points $x_{s(j)}$ and $x_{t(j)}$ and since the total cost in (1) is a weighted sum of all link costs, this implies that for a given set of flow vectors $\mathbf{f}^m$, the cost will also be a convex function of the locations of sensors x_i. This convexity permits the design of a fast algorithm to find the optimal sensor locations x_i^* and the corresponding minimal cost $g(\mathbf{f}^1, \ldots, \mathbf{f}^M)$ for a *given set of flow vectors*. More formally,

$$g(\mathbf{f}^1, \ldots, \mathbf{f}^M) = \min_{x_i} \sum_{m=1}^{M} \sum_{j \in \mathcal{E}} f_j^m e_j(x_{s(j)}, x_{t(j)}) \tag{10}$$

$$\text{s.t. } x_m = s_m, \quad m = 0, 1, \ldots, M$$

With $g(\mathbf{f}^1, \ldots, \mathbf{f}^M)$ as above, and keeping in mind the network tree structure and the elimination of (3), the main problem (1) becomes

$$\min_{\mathbf{f}^m} \; g(\mathbf{f}^1, \ldots, \mathbf{f}^M) \tag{11}$$

$$\text{s.t. } \sum_{j \in \mathcal{E}} a_{ij} f_j^m = -r_m d_i^m \quad i \in \mathcal{V},\ m = 1, \ldots, M \tag{12}$$

$$\sum_{j \in \mathcal{E}} b_{ij} f_j^m \le r_m \quad i \in \mathcal{V},\ m = 1, \ldots, M \tag{13}$$

$$f_j^m \in [0, r_m] \quad j \in \mathcal{E},\ m = 1, \ldots, M \tag{14}$$

where

$$b_{ij} = \begin{cases} 1 & \text{if arc } j \text{ leaves node } i \\ 0 & \text{otherwise} \end{cases}$$

In this formulation, (12) are still the flow balance equations, while constraints (13) and (14) build a unique path between each data source and the basestation; therefore guaranteeing the tree structure of the network.

Subproblems (10) and (11) suggest an iterative approach for solving the original problem. Starting with a feasible set of flow vectors $\mathbf{f}^1, \ldots, \mathbf{f}^M$, the first

step is to solve (10), which provides information used to update the flow vectors. An efficient gradient-based method for doing this (referred to as the "inner force method") is detailed in Ref. [40]. Briefly, it views the network as a dynamic system with "inner forces" applied to each node; in particular, a force applied to node i by link j is defined as

$$\mathbf{F}_{ij} = -\sum_{m=1}^{M} f_j^m \frac{\partial e_j(x_{s(j)}, x_{t(j)})}{\partial x_i} \tag{15}$$

Each such force causes node i to move towards the steepest descending direction that leads it to an equilibrium point (unique, owing to convexity) where all forces applied on i are balanced out.

The second step is to solve subproblem (11), i.e., find the optimal routing from all data sources to the sink in the tree structure resulting from the first step. Although this idea is straightforward, there is still a difficulty which prohibits its implementation. The difficulty is that $g(\mathbf{f}^1, \ldots, \mathbf{f}^M)$ is a non-convex and non-concave function of the flow vectors $\mathbf{f}^1, \ldots, \mathbf{f}^M$, which generally implies the existence of muliple local minima. Thus, we follow a different approach, based on the idea of (1) incrementing the number of nodes one at a time, (2) determining the optimal location of the new node and the corresponding flow vectors, and (3) repeating this process until the number of available nodes N is reached or the cost is sufficiently close to the known lower bound (9).

3.1.2. Incremental Iterative Approach

In an incremental deployment, the initial step is to begin with M nodes, each located at one of the M sources, and construct the corresponding tree structure with the basestation as its root. The associated flow vectors $\mathbf{f}^1, \ldots, \mathbf{f}^M$ are immediately given by (12) with a_{ij}, b_{ij} determined by this simple initial tree structure.

The next step is to add a node and determine its optimal location while preserving the network's tree structure. Unfortunately, as the number of nodes increases, the number of possible tree structures increases exponentially and constructing an efficient algorithm to find the optimal topology is a crucial issue. The approach proposed in Ref. [40] is based on a local topology adjustment, thus the size of the problem is limited; the price to pay is that global optimality can no longer be guaranteed. However, since, as discussed above, we know that the optimal deployment with an unlimited number of nodes consists of multi-hop straight line paths between every data source and the basestation, we have at our disposal the lower bound (9) that our solution can be compared to. As numerical results illustrate, this lower bound is rapidly approached by the proposed algorithm and with a number of nodes significantly smaller than the associated number K_{opt} given earlier.

The addition of a node and determination of its optimal location is a 3-step process. First of all, we determine which part of the network needs a new relay the most. Then, all possible topology adjustments around this area are obtained. Finally, the power improvement of each case will be checked and the one which provides the greatest improvement will become the new configuration. Figure 2 graphically summarizes the overall process.

Step 1. Determining the bottleneck node. A *bottleneck* node is defined as a node around which a new relay and corresponding new topology would bring the most improvement to the energy conservation of the whole network. The bottleneck node is determined by checking the inner forces applied to nodes: as mentioned earlier, the inner forces on a link contain the gradient information of the energy consumption on this link. The larger an inner force applied by a link on the node, the greater the energy savings by shortening this link. Before adding a new node, all nodes in the network have reached their equilibrium points. Thus,

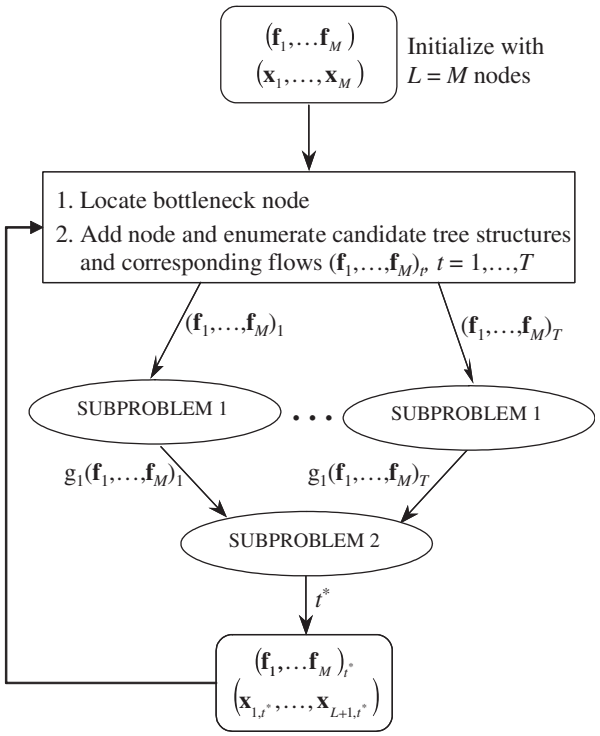

Fig. 2. Incremental iterative node deployment process.

if a node is balanced under several inner forces that have relatively larger magnitude, it follows that by shortening one of its links, the energy consumption on this link will improve greatly, but the cost involved on other links will overwhelm this improvement. Intuitively, we can visualize the area around this sensor node as being more sparse, and there is a higher need for a new relay in this region. With these observations in mind, we define the *sparseness* around node i as

$$SP_i = \sum_{j \in \mathcal{V}(i)} \|\mathbf{F}_{ij}\|$$

with $\mathbf{F}_{ij}$ given in (15) and the bottleneck node k is defined to be the node which has the greatest sparseness. That is,

$$k = \arg \max_{i=0,\ldots,N} SP_i$$

Obviously, there is no guarantee that the optimal location of the new node is indeed in the vicinity of the bottleneck node as defined above, so the solution implied by this approach is generally suboptimal.

Step 2. Enumerating topology adjustment options. The bottleneck node indicates the area which needs a new relay the most. Once it is determined, the precise placement of this new relay must be determined. Since we are working on a tree structure, the insertion of a new relay also means adding a new link. Thus, we need to consider topologies generated when an additional relay and link are present in the target area. An example will help to understanding the topology enumeration process. In Fig. 3, x_2 is the bottleneck node. As a node in the tree structure, it has three children nodes, x_3, x_4 and x_5, and its parent is x_1. The arrows in the figure indicate these relationships and x_6 is the new node we are adding. Several (but not all) possible topologies after inserting x_6 are shown in Fig. 3. As shown in Ref. [40], the number of all possible new topologies is $3 \cdot 2^{m-1} - 2$ where m is the number of children of the bottleneck node. For example, in Fig. 3 there are 10 possible new topologies. In case (a) of Fig. 3, we define x_3 to be the child of new node x_6 and x_2 to be its parent. As an alternative, if we define x_1 as x_6's parent, the parenthood relationship of x_2 and x_1 may or may not change: if it does not change, case (d) is obtained, and if it does, we get case (c).

Step 3. Obtaining a new deployment. The outcome of Step 2 when the current number of nodes is $L < N$ is a number of possible new network tree structures, say T_L, each with associated flow vectors $(\mathbf{f}^1, \ldots, \mathbf{f}^M)_t$, $t = 1, \ldots, T_L$. For each such structure t, subproblem (10) is solved (as described earlier), giving the corresponding optimal node locations $x_{i,t}$, $i = 1, \ldots, L+1$ and cost $g_t(\mathbf{f}^1, \ldots, \mathbf{f}^M)$. Next, the solution of problem (11) reduces to comparing all such costs and determining $t^* = \arg \min_{t=1,\ldots,T_L} g_t(\mathbf{f}^1, \ldots, \mathbf{f}^M)$, the corresponding node locations $x_{i,t^*}, i = 1, \ldots, L+1$, and the flows $(\mathbf{f}^1, \ldots, \mathbf{f}^M)_{t^*}$.

Example. As shown in Fig. 4, a deployment problem for a sensor network with $M = 9$ data sources is solved. In this network, $r_m = 1.0\,\text{kb/s}$ $(m = 1, \ldots, 9)$. Data sources are located on a 3×3 grid. The sink is located on the left of this grid. The distance between two neighboring data sources is 300 m. The distance between the sink and its nearest data source is also 300 m. In this example, we assume $n = 4$ and $\alpha_1 = 180\,\text{nJ/bit}$ and $\alpha_2 = 0.001\,\text{pJ/bit/m}^4$.

Initially, there are only nine sensors in the network (star nodes in Fig. 4), one for each data source $(N = M = 9)$. In order to send data back to the base-station, nine data links have been built as shown. Next, we incrementally add more sensors into the network which act as relays. Figure 4 demonstrates the min-power deployment configurations obtained when there are L relays present $(L = 5, 10, \ldots, 45)$. As more nodes are deployed, we expect to observe an improvement in transmission power consumption. Figure 5 demonstrates this improvement. In this figue, L is the total number of relays in the network $(L = N - M)$. P_{self} (solid line) is the total transmission power (mW) of the network given that relays are deployed using the proposed incremental self-deployment algorithm. P_{opt} (shown as a triangle) is the minimum power consumption of the network based on the assumption that there is no constraint on the number of relay nodes (in this case, $P_{\text{opt}} = 16.122\,\text{mW}$, which is obtained at $L = 59$). As shown in Fig. 5, for this nine data source setting, as L increases the transmission power monotonically decreases towards the lower bound P_{opt}. The tradeoff between transmission power consumption and the number of relays deployed in the network is also shown in this figure. When $L = 28$, the total transmission power consumption is 16.874 mW, which is $<5\%$ more than the minimum-power consumption P_{opt}. However, this deployment scheme uses less than half of the relays that are needed to build up the minimum-power network.

In closing, we should point out that the incremental deployment approach above is based on a centralized scheme. It assumes the existence of a controller with powerful computational capabilities, perfect information of the whole network and unlimited control over all sensor nodes. In the case of mobile nodes (to be discussed in Section 4) but still known data sources,

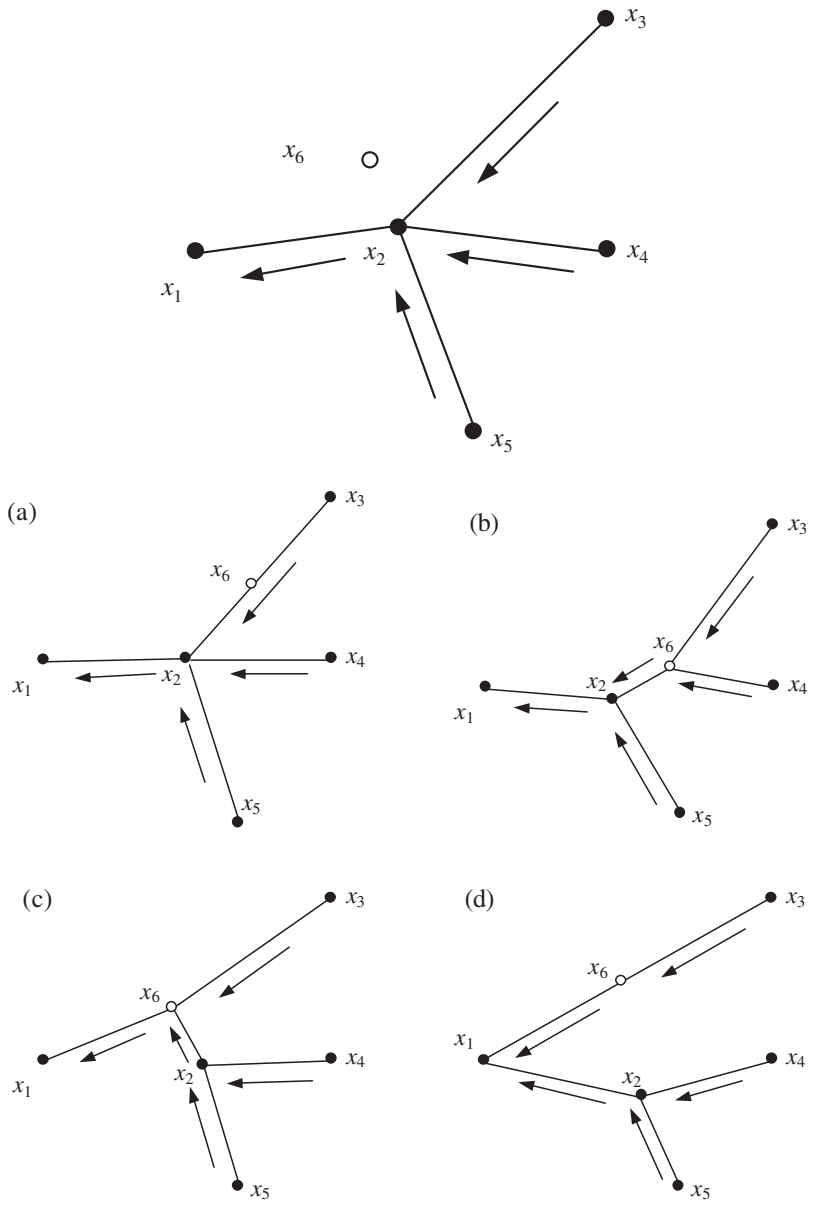

Fig. 3. Some possible topologies resulting from the addition of node $\mathbf{x}_6$.

an open problem is the development of distributed algorithms for sensor node deployment through which an individual sensor node can autonomously decide how to move based on its own knowledge of the overall system.

3.2. Routing and Scheduling

The wireless nature of sensor networks makes routing and scheduling issues particularly challenging, especially when one factor in the limited power and computational resources at sensor nodes, the fact that some nodes may occasionally be in a sleeping state or simply dead, as well as security concerns. A routing policy is responsible for forwarding packets to their ultimate destinations. Unlike traditional routing protocols in wired networks that can rely on global information, routing protocols in sensor networks typically adopt local cost information (e.g., distance between two nodes) in order to make routing decisions. In what follows, we limit ourselves to a brief overview of recent contributions to the routing problem so as to raise the reader's awareness of the issues involved as some of them play a crucial role in our discussion of networks with mobile nodes in Section 4.

A comprehensive survey of routing methods for

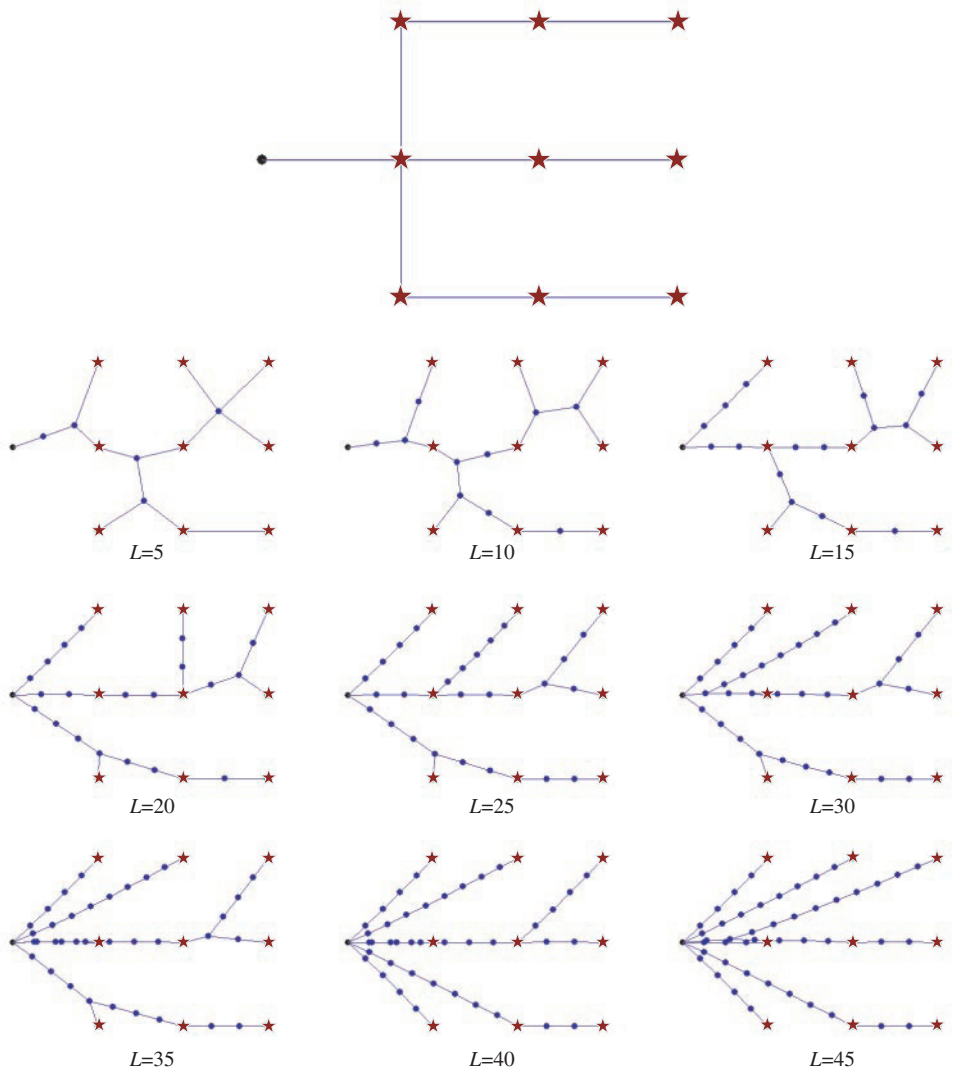

Fig. 4. A nine data source example and incremental node deployment.

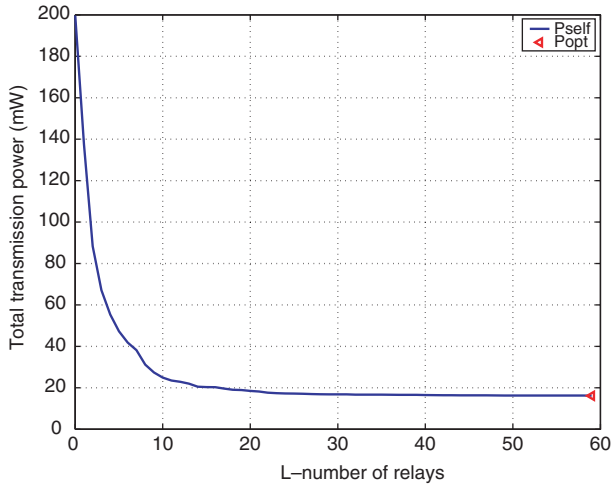

Fig. 5. Incremental self-deployment algorithm performance as a function of nodes deployed.

wireless networks in general can be found in Ref. [1]. For sensor networks in particular, routing algorithms that have been proposed may be divided into two categories, depending on the network structure adopted, as discussed in Section 2: flat and hierarchical routing algorithms. Examples of flat algorithms are the "directed diffusion" approach [32], a sink-initiated reactive routing technique in which routes are established when requested by the sink, and the minimum cost forwarding algorithm in Ref. [69], which exploits the fact that in sensor networks the data flow is always towards a fixed sink: similar to the natural gravity field that drives waterfalls from the top of a mountain to the ground, at each point, data traffic flows from a high post to a low post along the shortest path. For this algorithm to work, each node needs to have the least cost estimate from itself to the sink. Examples of

hierachical algorithms are the low energy adaptive clustering hierarchy (LEACH) scheme [28], a hierarchical routing algorithm in which elected clusterheads transmit aggregated data to the sink directly, and the power-efficient gathering in sensor information system (PEGASIS) algorithm [43], in which a node communicates only with its closest neighbor and takes turns being the leader for transmission to the sink.

Limiting ourselves to flat network structures, it is fair to say that the majority of the proposed routing methods is based on shortest path algorithms [34,51,53]. Such algorithms usually require each node to maintain a global cost (or state) information table, which is a significant burden for resource-constrained sensor networks. Karp and Kung [36] have proposed a greedy perimeter stateless routing (GPSR) protocol, which uses the real positions of nodes in the network and a packet's destination to make routing decisions. The advantage of this protocol is that each node only needs to keep track of local state information. In order to deal with the issue of node failures, Ganesan et al. [24] proposed a multipath routing algorithm, so that a failure on the main path can be recovered without initiating a network-wide flooding process for path rediscovery. Since flooding consumes considerable energy, this routing method can extend the network's lifetime when there are failures. However, finding multiple paths and sending packets through them also consumes energy, thus adversely impacting the lifetime of the network if there are no failures.

The routing policies mentioned above may indirectly reduce energy usage in sensor networks but they do not explicitly use energy consumption models to address optimality of a routing policy with respect to energy-aware metrics. In recent years, such "energy awareness" has motivated a number of minimum-energy routing algorithms that typically seek paths minimizing the energy per packet consumed (or maximizing the residual node energy) to reach a destination [65]. However, as also pointed out in Ref. [18], seeking a minimum energy (or maximum residual energy) path can rapidly deplete energy from some nodes and ultimately reduce the full network's lifetime by destroying its connectivity. Thus, an alternative perfomance metric is the *network lifetime*. Along these lines, Shah and Rabaey [62] proposed an energy aware routing (EAR) policy which does not attempt to use a single optimal path, but rather a number of sub-optimal paths that are probabilistically selected with the intent of extending the network lifetime by "spreading" the traffic and forcing nodes in the network to deplete their energies at the same time. In EAR, each node builds a cost information table and propagates local cost information to other nodes. Costs are determined by the residual energies of each node and by the distances between them. Each node also maintains a routing probability table determined by local cost information. In Ref. [18], routing with the goal of network lifetime maximization is formulated as a linear programming problem where the decision variables are source to destination path flows and a shortest cost path routing algorithm is proposed to efficiently approximate its solution; link costs are defined to combine energy consumption and residual energy at the end nodes of each link.

Let us take a closer look at the functionality of an energy-aware shortest path routing algorithm for sensor networks. Returning to the viewpoint of a sensor network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, as discussed in the previous section, let us now denote a node location by $s_i, i \in \mathcal{V}$ and a link by $(i, j) \in \mathcal{E}$ with a cost e_{ij} given by

$$e_{ij} = e(\|s_i - s_j\|), \quad (i, j) \in \mathcal{E} \tag{16}$$

where $e(\cdot)$ is the communication energy consumption on (i, j) as in (8). Over $\mathcal{G}$, a routing protocol generates a set of "shortest paths" $\mathcal{L} = \{l_1, \ldots, l_N\}$ between each sensor node and the basestation b. Here, a path $l_i = \{(I, j), \ldots, (k, 0)\}$ is said to be a "shortest path" between node i and 0 in the sense that $c_i = \Sigma_{(j,k)\in l_i} e_{jk}$ is minimized over all possible paths between i and 0. It is well known that the set of shortest paths $\mathcal{L}$ forms a *tree structure* [57], and it can be expressed by a *forward index vector* $H = (h_1, \ldots, h_N)$, where $h_i \in \{0, 1, \ldots, N\}$ denotes the index of the next-hop node when forwarding data from i. A particular routing protocol is responsible at each sensor node i for computing the forward index h_i and forward cost c_i. The routing protocol also provides node i an *upstream vector* $U_i = (u_1^i, \ldots, u_N^i)$ and a *cumulative flow factor* z_i defined as

$$u_j^i = \mathbf{1}[h_j = i] \quad \text{and} \quad z_i = r_i + \sum_{j=1}^{N} u_j^i z_j$$

where u_j^i indicates whether j is i's upstream node and z_i records the total data rate originated from i: r_i accounts for data collected at i and $\sum_{j=1}^{N} u_j^i z_j$ is the total traffic from upstream nodes.

An example of a routing protocol that dynamically updates the shortest-path set $\mathcal{L}$ and generates at each node i the forward index h_i, forward cost c_i, upstream vector U_i and cumulative flow z_i in a distributed manner is the geographic energy-aware routing (GEAR) protocol proposed in Ref. [70]. According to this protocol, nodes in a sensor network dynamically update next-hop destinations by repeatedly applying a

learning step. Suppose that after the k-th learning step, a node i stores a set of state variables h_i^k, c_i^k, U_i^k and z_i^k which define the current routing. Upon the $(k+1)$th learning step starting, node i broadcasts a control packet and requests both forwarding cost c_j^k and location s_j from all nearby nodes. After receiving replies, i updates routing by computing new h_i^{k+1} and c_i^{k+1} as follows:

$$h_i^{k+1} = \arg\min_{j\in\mathcal{U}_i}\left[c_j^k + e(\|s_i - s_j\|)\right]$$
$$c_i^{k+1} = \min_{j\in\mathcal{U}_i}\left[c_j^k + e(\|s_i - s_j\|)\right]$$

where $\mathcal{U}_i$ is the set of neighboring nodes that respond to node i's requests. In the case that $h_i^{k+1} \neq h_i^k$, node i sends control packets to node h_i^k and h_i^{k+1} in order to inform them of this routing change. Upon receiving the packet, node h_i^k removes i as an upstream node by updating its state variables $U_{h_i^k}^{k+1}$ and $z_{h_i^k}^{k+1}$ using

$$u_i^{h_i^k} = 0, \quad z_{h_i^k}^{k+1} = z_{h_i^k}^k - z_i^k$$

Node h_i^{k+1}, however, adds i as its upstream node, so that

$$u_i^{h_i^{k+1}} = 1, \quad z_{h_i^{k+1}}^{k+1} = z_{h_i^{k+1}}^k + z_i^k$$

In this way, after a learning step, node i updates its local routing and the corresponding state variables. As shown in Ref. [70], by repeatly performing learning steps at each sensor node, the resulting forward index vector H converges to the shortest paths in finite time.

Taking a network security viewpoint, deterministic routing policies (i.e., policies where source nodes send data through one or more fixed paths) are highly vulnerable to node compromising and falsifying cost information, leading to denial of service (DoS) attacks [68]. For example, a "sink-hole attack" compromises a node and broadcasts a fake low cost to neighboring nodes, thus enticing all such nodes to route packets to it. The neighboring nodes in turn broadcast the low cost of the compromised node to their neighbors and the end effect is that this node acts as a sink hole for all packets while also draining the energy of the network nodes. In order to reduce the effect of such attacks, probabilistic routing is an interesting alternative, since this makes it difficult for attackers to identify an "ideal" node to take over. In this sense, the EAR policy is attractive because of its probabilistic routing structure, even though it does not attempt to provide optimal routing probabilities for network lifetime maximization. Moreover, given the dynamic behavior of a sensor network in terms of changing topology, node failures and energy consumption, one can expect optimal routing probablities to be time-varying. Thus, an optimal control problem formulation is a natural setting. where the goal is to maximize the network's lifetime. In Ref. [67] it is shown that in a fixed topology case there exists an optimal policy consisting of fixed routing probabilities that may be obtained by solving a set of relatively simple nonlinear programming problems.

Turning to the issue of scheduling, the most challenging current issues are again related to energy awareness and an effort to control the transmission rate at a node based on the state of the system [66,71]. In an additive white Gaussian noise (AWGN) channel, channel capacity is determined by Shannon's theorem [64]: $C = B\log_2(1+\mathrm{SNR})$, where B is the channel bandwidth and SNR is the signal-to-noise ratio. Assuming the channel utilization ratio is a constant α, $0<\alpha<1$, the data rate can be represented by $r=\alpha C$. It can be seen that increasing the data rate linearly will increase the transmission power exponentially. The *dynamic transmission control* (DTC) problem involves controlling the transmission rates, which will in turn determine the transmission time of a set of tasks, with the objective of minimizing the total RF transmission energy consumption. This problem has many similarities to the *dynamic voltage scaling* (DVS) problem that will be discussed in the next section.

In some cases, the problems of routing and scheduling are jointly treated as a utility maximization problem [55] subject to explicit fairness constraints (e.g., imposing equality among all average transmission rates), modeled through linear inequalities. A transmission of a traffic class k packet by node i to node j is denoted by (i, j, k). Let r_{ijk} denote the net flow rate for an (i, j, k) or a (j, i, k) transmission and $\mathbf{r}$ the corresponding transmission rate vector. Then, the problem posed in Ref. [55] is

$$\max\ F(\mathbf{r})$$
$$\text{s.t. } \mathbf{r}\in \mathrm{Conv}(R)\cap S.$$

where R be the set of feasible transmission rate vectors (based on power constraints) and S is the set of rate vectors that satisfy a collection of fairness constraints and flow conservation. It is also assumed that $F(\mathbf{r})$ is continuous, concave and bounded in $\mathrm{Conv}(R)$. An obvious difficulty in solving this problem is the need to characterize $\mathrm{Conv}(R)$. An efficient decomposition algorithm is proposed in Ref. [55] that terminates in a finite number of iterations and produces a policy that is asymptotically optimal at low transmission power levels at which sensor networks typically operate.

The resulting policy selects a number of modes at which the network should operate (i.e., determining who transmits to whom and at what power) and an appropriate time-sharing schedule among these modes.

3.3. Power Control: The Dynamic Voltage Scaling Problem

The issue of controlling the power of a sensor node manifests itself in all problems we have discussed thus far, e.g., it appears in the objective function of the deployment problem and is a central part of all energy-aware routing algorithms that have been proposed for sensor networks. However, in addition to indirect ways of managing power by controling the placement of nodes, link data rates and transmission schedules, there are also direct ways of power control. One such way is by explicitly turning a node off and periodically entering a "sleeping" state, as discussed in Section 2. Another approach, which we will discuss in this section, is by dynamically controlling the voltage of a node's power source while it is not in a "sleeping" state.

As mentioned already, sensor nodes incorporate small, inexpensive devices with limited battery capabilities. Prolonging battery life is closely tied to the network's overall performance; in sensor networks, the failure of a few nodes can cause significant topological changes that require substantial additional power to reorganize the network. In low-power systems, the processor reportedly accounts for 18–30 of the overall power consumption and often exceeds 50% [54]. Controlling the voltage and clock frequency provides the means to regulate processor power consumption leading to DVS techniques [29,52,54,60]. At the heart of these techniques lies the basic fact that the energy consumption E of a device's processor is related to the operating voltage V through $E = C_1 V^2$ and its processing frequency (clock speed) is given by $f = (V - V_t)/C_2 V$ where C_1, C_2 are constants dependent on the physical characteristics of a device and V_t is the threshold voltage, so that $V \geq V_t$. These relationships may be approximate, but the functional interdependence of V, E and f clearly indicates that reducing the voltage provides an opportunity to reduce energy, which comes at at the expense of longer delays. This, in turn, adversely affects performance, with possibly catastrophic consequences in systems where tasks must satisfy hard real-time constraints [9]. Thus, managing this tradeoff to take advantage of a quadratic energy reduction through $E = C_1 V^2$ without substantially degrading latency performance becomes an essential design and dynamic control problem.

A number of DVS algorithms have been proposed over the last decade. Most of them are designed for preemptive scheduling of real-time systems, as in Refs [2,37]. Nonpreemptive scheduling is often a better choice in practice, especially for systems with very limited resources, because uncontrolled preemption can give rise to a large number of context switches requiring larger stack sizes and increased energy consumption [33,35]. DVS algorithms developed for the nonpreemptive case have been reviewed in Ref. [30]. Many of them were developed for systems with periodic tasks, as in Ref. [59]. Aperiodic tasks, however, are more likely to occur in a setting consisting of asynchronously operating components such as a sensor network where sensor nodes asynchronously supply data to a processing node such as the clusterheads discussed in Section 2.

The fundamental tradeoff between power and delay naturally gives rise to a variety of optimization problems that depend on (1) the importance of task timing, (2) the ability to control voltage continuously as opposed to in between tasks only and (3) the information based on which a DVS controller can operate.

Regarding (1), tasks are classified as having "soft" or "hard" timing requirements associated with them. In the case of soft requirements, the objective function of an optimization problem incorporates a cost that penalizes long task completion times (a "task" may involve data processing at a node or it may refer to transmitting a data packet). In the case of hard requirements, the problem is formulated with explicit constraints on the task completion times.

Regarding (2), the simplest form of DVS is to change the voltage with the start of any new task. Alternatively, however, control may also be applied while a task is processed when a significant new event takes place (typically, the arrival of a new task at the node) or even at arbitrary time instants. If a controller is constrained to remain invariant during the full processing of a task, we refer to it as *static*, otherwise it is called *dynamic*. Note, however, that a static controller can still assign a different voltage value to every new task.

Finally, (3) pertains to the knowledge the controller has at its disposal when called upon to act. Normally, we can assume that characteristics of a task such as its required number of operations (in the case of data processing) or number of bits in a packet (in the case of a transmission task) are known. However, the arrival times of tasks may not be known, unless a processor operates by scheduling in advance a given number of tasks. When the arrival times of tasks

are known, we may develop an *off-line* controller; otherwise, we refer to an *on-line* controller, which operates knowing only arrival times of tasks that are already in queue and, possibly, some limited future arrival times or their estimates.

The off-line static control problem with soft timing requirements may be solved using techniques developed for the optimal control of discrete event and hybrid systems. In particular, problems of these types were addressed in Ref. [11] where it was shown that an efficient forward decomposition algorithm (FA) may be used to avoid the combinatorial complexity that often comes with such problems. The FA still requires the solution of N convex optimization problems (where N is the number of tasks), which is generally demanding for on-line applications with limited on-board computational capacity. The off-line static control problem with hard timing requirements, however, possesses structural properties that allow us to further decompose the problem and solve it without having to rely on any optimization and independent of the physical characteristics of the devices involved [50]. An interesting recent result related to the static and dynamic versions of the off-line problem with hard timing requirements is that they both give the same solution [44]. The result is significant because it asserts the optimality of a simple controller that does not require any data collection or processing in environments where the cost of such actions is high. Moreover, a static controller requires no overhead that would otherwise be involved in making continuous control adjustments.

In what follows, we will limit our discussion to off-line static control problems. The on-line DVS control problem remains open when no knowledge of task arrival times is available. If these arrival times are probabilistically characterized, this gives rise to a stochasic optimal control problem. Alternatively, if some limited future information is possible, then a receding horizon control approach can be adopted, which is characterized by several attractive properties as recently shown in Ref. [45].

3.3.1. Off-Line Static Control Problem Formulation

The sensor network node we consider is modeled as a single-stage first-come-first-served queueing system. Let a_i denote the (known) arrival time of task i and x_i denote its departure upon completing processing. The dynamics of this system are given by $x_i = \max(x_{i-1}, a_i) + u_i,\ i = 1, \ldots, N$, where u_i is the processing time of task i. Let $\theta_i(u_i)$ be an energy consumption function for task i, which depends on u_i. Letting μ_i denote the number of operations needed for task i (which may depend on the specifics of this task) and using the relationships $E = C_1 V^2$ and $f = (V - V_t)/C_2 V$ we can write

$$\theta_i(u_i) = \mu_i E = \mu_i C_1 \left(\frac{V_t u_i}{u_i - \mu_i C_2} \right)^2 \tag{17}$$

We emphasize, however, that the precise form of $\theta_i(u_i)$ or the values of the constants are not essential; what matters is only that $\theta_i(u_i)$ is a continuously differentiable strictly convex and monotone decreasing function of u_i for $u_i > \mu_i C_2$. Note that an additional constraint on V is imposed by the requirement that $V \leq V_{\max}$, where $V_{\max}$ is the maximal operating voltage. This, in turn, leads to a constraint on the control variables:

$$u_i \geq u_{i\min} = \frac{\mu_i C_2 V_{\max}}{V_{\max} - V_t}$$

We can now formulate the following problem **P1** for the case of soft timing requirements:

$$\min_{u_1,\ldots,u_N} \left\{ J = \sum_{i=1}^{N} \theta_i(u_i) + \psi_i(x_i) \right\}$$
$$\text{s.t. } u_i \geq u_{i\min}, \quad i = 1, \ldots, N; \quad x_0 = 0;$$
$$x_i = \max(x_{i-1}, \bar{a}_i) + u_i, \quad i = 1, \ldots, N$$

where $\psi_i(x_i)$ is a continuously differentiable strictly convex function intended to penalize departure times. Thus, the cost function above captures the tradeoff between energy and task timing. In the case of hard timing constraints, each task is also assigned a deadline d_i and the problem, referred to as **P2**, becomes

$$\min_{u_1,\ldots,u_N} \left\{ J = \sum_{i=1}^{N} \theta_i(u_i) \right\}$$
$$\text{s.t. } u_i \geq u_{i\min}, \quad i = 1, \ldots, N; \quad x_0 = 0;$$
$$x_i = \max(x_{i-1}, \bar{a}_i) + u_i \leq d_i, \quad i = 1, \ldots, N.$$

3.3.2. Optimal Sample Path Decomposition

In both problems **P1** and **P2**, the key to obtaining efficient controllers without resorting to an explicit solution of these hard constrained nonlinear optimization problems is the fact that there exists a set of indices $\mathcal{M} = \{m_1, \ldots, m_M\}$ with $1 \leq m_1 < \cdots < m_K \leq N$ such that an optimal sample path $[a_1, x_N^*]$ can be decomposed into intervals

$$[a_1, x_{m_1}^*], [a_{m_1+1}, x_{m_2}^*], \ldots, [a_{m_k}, x_{m_k+1}^*], \ldots, [a_{m_M}, x_N^*]$$

with the following property: $x^*_{m_k} < a_{m_k+1}$ and the optimal cost J^* of **P1** can be written as

$$J^* = \sum_{k=1}^{M} J^*_k$$

where, for consistency, $x^*_{m_0} = a_1, x^*_{m_M+1} = x^*_N$, and J^*_k is the solution of

$$\min_{u_{m_k},\ldots,u_{m_k+1}} \sum_{i=m_k}^{m_k+1} [\theta_i(u_i) + \psi_i(x_i)]$$
$$\text{s.t. } u_i \geq u_{i\min}, \quad i = m_k, \ldots, m_k + 1;$$
$$x_i = x_{i-1} + u_i, \quad i = m_k, \ldots, m_k + 1.$$

and similarly for **P2**. It can be shown [11] that each subproblem above is a convex optimization problem with linear constraints. In other words, a complex nonlinear optimization problem with nonlinear, nondifferentiable constraints (because of the max operator involved) is replaced by a set of much simpler problems. Of course, for this decomposition to be practically useful we need an effective way for identifying the values of $m_1, \ldots, m_M$. It is shown in Ref. [11] that for **P1** this is possible through a simple efficient algorithm. The case of **P2** is particularly interesting because of the property that $d_i < a_{i+1}$ implies $i \in \mathcal{M}$ [50], so the set $\mathcal{M}$ can be *a priori* determined. Moreover, one can also show [50] that each $[a_{m_k}, x^*_{m_k+1}]$ can be further decomposed through a set of *critical tasks* $\{r_1(k), \ldots, r_R(k)\}$. Setting $\tau_i = u_i/\mu_i$, the property of this decomposition is that all optimal controls τ^*_i, $i = r_j(k) + 1, \ldots, r_{j+1}(k)$, in each $[x^*_{r_j(k)}, x^*_{r_{j+1}(k)}]$ are constant; in particular,

$$\tau^*_i = \frac{x^*_{r_{j+1}(k)} - x^*_{r_j(k)}}{\sum_{l=r_j(k)+1}^{r_{j+1}(k)} \mu_l}, \quad i = r_j(k) + 1, \ldots, r_{j+1}(k)$$

and the indices $\{r_1(k), \ldots, r_R(k)\}$ can also be determined through a low-complexity algorithm.

The resulting *critical task decomposition algorithm* (CTDA) derived in Ref. [50] is of limited computational complexity, requiring no optimization problem to be solved. It also has small space complexity, which makes it appealing for applications involving devices with limited memory, and it is independent of the details of the energy function, which implies that there is no need to measure parameters such as C_1 or C_2 in (17).

4. Networks with Cooperative Mobile Nodes

Endowing nodes in a sensor network with mobility drastically expands the spectrum of the network's capabilities. Moreover, assuming that each mobile node possesses a certain amount of decision making autonomy gives rise to a dynamic system with a considerable amount of flexibility, depending on the extent to which the nodes can cooperate in order to perform a "mission". This flexibility, for example, allows us to handle a large number of data source targets (which we shall henceforth refer to as simply "targets") with a much smaller number of nodes that can move and visit the targets over time to perform various tasks.

Naturally, mobility also implies an additional layer of complexity. For example, if communication connectivity is to be maintained, we must ensure that each node remains within the range of at least some other nodes. We must also take into account that mobility consumes a considerable amount of energy, which amplifies the need for various forms of power control. Another interesting aspect of mobility is that the exact location of nodes is not always available to other nodes or to a basestation. This is especially true in settings where GPS tracking is not applicable, such as locating people or important equipment in a building (this is referred to as "asset tracking"). The *location detection* problem is a particularly challenging one, although we do not discuss it in this paper.

Taking a system and control theory perspective, mobile networks provide the opportunity to exercise real-time cooperative control involving their nodes. The goal of cooperative control is to coordinate the actions of the nodes so as to achieve a common objective which we shall refer to as the *mission* of the network. Its most popular application to date has been in networks of uninhabited autonomous vehicles (UAVs) [13,15,56], in particular studying vehicle trajectory generation for the purpose of formation control, obstacle avoidance or stabilization [7,26,27,31,42].

Depending on the type of mission one wishes to define for a network, a variety of problems can be formulated in the context of cooperative control. In what follows, we will limit ourselves to two types of missions that are suited to sensor networks. First, we will discuss a *reward maximization mission*, where the mission space contains N *target points* indexed by $i = 1, \ldots, N$, with target i having an associated (possibly time varying) reward R_i. A mission here is the process of controlling the movement of the nodes and visiting various targets so as to maximize the total reward collected within a given mission time T. Second, we will consider a *coverage control mission*, where the main difficulty is that targets are unknown and the mission involves positioning the nodes so as to maximize the probability of detecting data originating at the targets.

4.1. Reward Maximization Missions

In this class of missions, we consider a set $\mathcal{A}$ of M mobile nodes indexed by $j=1,\ldots,M$ and a set $\mathcal{T}$ of N *targets* indexed by $i=1,\ldots,N$ in a 2D space. Associated with the i-th target is a reward R_i. The mission's objective is to maximize the total reward collected by visiting points in the set $\mathcal{T}$ within a given mission time T. The problem is complicated by several factors. (1) Target rewards may be time-dependent, typically decreasing in time; thus, the order in which targets are visited by nodes may be critical. (2) Different nodes have different capabilities so that assigning specific nodes to specific targets can also be critical. (3) The exact location of targets may not always be known in advance. (4) There may be obstacles in the mission space, which constrain the feasible trajectories of nodes.

This setting gives rise to a complex stochastic optimal control problem whose solution is computationally intractable even for relatively simple mission control settings. It is, therefore, natural to decompose it into various subproblems at different levels – from detailed motion control to higher-level path planning and assignment of nodes to targets. For example, Refs [19,23] address issues of dynamically allocating resources, while [8] formulates the problem of cooperative path planning as a mixed-integer linear program (MILP) that incorporates task timing constrains and the presence of obstacles. An alternative to this functional decomposition approach is one based on *time decomposition*. The main idea is to solve an optimization problem seeking to maximize the total *expected* reward accumulated by the network over a given time horizon, and then continuously extend this time horizon forward (either periodically or in purely event-driven fashion). This idea, introduced in Ref. [12], is in the spirit of receding horizon (RH) schemes, which are associated with model-predictive control and used to solve optimal control problems for which feedback solutions are extremely hard or impossible to obtain; see Refs [17,48,61] and more recently [20–22,58]. The resulting cooperative control scheme dynamically determines node trajectories by solving a sequence of optimization problems over a *planning* horizon and executing them over a shorter *action* horizon. We should emphasize that the optimization problem involved does not attempt to make any explicit node-to-target assignments, but only to determine headings that, at the end of the current planning horizon, would place nodes at positions such that a total expected reward is maximized. Thus, it is a relatively simple problem to solve. It turns out, however, that node trajectories actually *converge* to targets, despite the fact that this approach, by its nature, was never intended to perform any such discrete node-to-target assignment. In what follows we outline the approach which is described in detail elsewhere [38].

The location of the i-th target is denoted by $y_i \in \mathbb{R}^2$. Note, however, that some of the target locations may not be known to the cooperating nodes. Let $x_j(t) \in \mathbb{R}^2$ denote the position of the j-th node at time t, with intial positions given by x_{j0}, $j=1,\ldots,M$. For simplicity, we assume nodes travel at constant velocity throughout the mission, that is

$$\dot{x}_j(t) = V_j \begin{bmatrix} \cos u_j(t) \\ \sin u_j(t) \end{bmatrix}, \quad x_j(0) = x_{j0} \tag{18}$$

where $u_j(t) \in [0, 2\pi]$ is the controllable heading of node j and V_j is the corresponding velocity. We note that M, N and y_i may change in time.

To distinguish the relative importance of targets at time t, each target has an associated *reward function* denoted by $R_i\phi_i(t)$, where R_i is the maximal reward and $\phi_i(t) \in [0,1]$ is a discounting function which describes the reward change over time. Note that by appropriately selecting $\phi_i(t)$, it is possible to capture timing constraints imposed on target points, as well as precedence constraints. By allowing $R_i < 0$ for some i and properly selecting $\phi_i(t)$, we may also model an obstacle in the mission space (see also Ref. [39]). A simple discounting function we can use is the linear one:

$$\phi_i(t) = 1 - \frac{\alpha_i}{T}t, \quad \alpha_i \in (0,1] \tag{19}$$

When a deadline is associated with a paricular target point, we can use

$$\phi_i(t) = \begin{cases} 1 - \dfrac{\alpha_i}{D_i}t & \text{if } t \le D_i \\ (1-\alpha_i)\mathrm{e}^{-\beta_i(t-D_i)} & \text{if } t > D_i \end{cases} \tag{20}$$

where D_i is a deadline assigned to target point i, and $\alpha_i \in (0,1]$, $\beta_i > 0$ are parameters which may be target-specific and chosen to reflect different cases of interest.

In order to distinguish the effectiveness of nodes relative to a target i, we define a *node capability factor* $p_{ij}(t) \in [0,1]$, which reflects the probability that a node j visiting point i at time t will complete its task and collect the reward $R_i\phi_i(t)$. We say that vehicle j *visits* target point i at time t if $\|x_j(t) - y_i\| \le s_i$ and $p_{ij}(t) > 0$. Thus, $s_i > 0$ can be viewed as the *size* of a target. If during a visit the node successfully completes its task, it will collect the corresponding reward and, at the same time, target i is no longer of interest to the

mission and it is removed from the set $\mathcal{T}$. Since a visit at i is related to the consumption of node j's resources, the capability factor $p_{ij}(t)$ may decrease after the visit, i.e., $p_{ij}(t^+) \leq p_{ij}(t^-)$.

4.1.1. Cooperative Structure

Nodes cooperate by dynamically partitioning the mission space and implicitly allocating regions of it among themselves. Given an arbitrary point $y \in \mathbb{R}^2$ in the mission space (not necessarily a target), we would like to assign this point to vehicles at time t so that y is assigned to the closest vehicle with the highest probability. To formalize this idea, we first define a *neighbor set* $\mathcal{B}^b(y,t)$ to include the b closest vehicles to $y \in \mathbb{R}^2$ at time t, where $b \in \{1,\ldots,M\}$. Let $B^l(y,t)$ be the l-th closest vehicle to y, that is

$$B^l(y,t) = \arg\min\{\|x_k(t) - y\| : k \in \mathcal{A}, \; k \neq B^1(y,t),\ldots,B^{l-1}(y,t)\}, \quad l = 1,\ldots,M$$

so that $\mathcal{B}^b(y,t) = \{B^1(y,t),\ldots,B^b(y,t)\}$. We then define the *relative distance function*, $\delta_j(y,t)$, as follows:

$$\delta_j(y,t) = \begin{cases} \dfrac{\|x_j(t) - y\|}{\Sigma_{k\in\mathcal{B}^b(y,t)}\|x_k(t) - y\|} & \text{if } j \in \mathcal{B}^b(y,t) \\ 1 & \text{otherwise} \end{cases} \tag{21}$$

Of particular interest is the case $b = 2$, so we set $\mathcal{B}(y,t) \equiv \mathcal{B}^2(y,t)$. Moreover, when y coincides with a target point y_i, we write $\mathcal{B}_i(t) \equiv \mathcal{B}(y_i,t)$ and similarly $\delta_{ij}(t) \equiv \delta_j(y_i,t)$.

Next, we define a *relative proximity function* $q_j(y,\delta_j)$ to be any monotonically nonincreasing function of δ_j such that $q_j(y,0) = 1, q_j(y,1) = 0$. An example of such a function when $b = 2$ is

$$q_j(y,\delta_j) = \begin{cases} 1 & \text{if } \delta_j \leq \Delta \\ \frac{1}{1-2\Delta}[(1-\Delta) - \delta_j] & \text{if } \Delta < \delta_j \leq 1-\Delta \\ 0 & \text{if } \delta_j > 1-\Delta \end{cases} \tag{22}$$

where $\Delta \in [0, 1/2)$ is an adjustable parameter, which can be interpreted as a "capture radius": if a target i happens to be so close to node j as to satisfy $\delta_j(y_i,t) \leq \Delta$, then at time t node j is committed to visit target i. Again, when y coincides with a target y_i, we write $q_{ij}(\delta_{ij}) \equiv q_j(y_i,\delta_j)$. We can now view $q_{ij}(\delta_{ij})$ as the probability that target i is assigned to node j at time t, based on the value of $\delta_{ij}(t)$, and observe that $\Sigma_j q_{ij}(\delta_{ij}) = 1$.

4.1.2. Cooperative Receding Horizon (CRH) Trajectory Construction

The objective of the mission is to collect the maximal total reward by the end of some mission time T (or some specific event that defines the end of the mission). To meet this goal, we design a cooperative controller that generates a set of trajectories for each node in the team $\mathcal{A}$ during $[0,T]$. This on-line controller is applied at time points denoted by t_k, $k = 0,1,\ldots,$ during the mission time. At t_k, the controller operates by solving an optimization problem $\mathbf{P}_k$, whose solution is the control vector $\mathbf{u}_k = [u_1(t_k)\cdots u_M(t_k)]$. Next, we explain how $\mathbf{P}_k$ is formulated.

Suppose that nodes are assigned headings $u_1(t_k),\ldots,u_M(t_k)$ at time t_k, intended to be maintained for a *planning horizon* denoted by H_k. Then, at time $t_k + H_k$ the *planned* positions of the nodes are given by

$$x_j(t_k + H_k) = x_j(t_k) + \dot{x}_j(t_k)H_k \tag{23}$$

Define

$$\tau_{ij}(\mathbf{u}_k, t_k) = (t_k + H_k) + \|x_j(t_k + H_k) - y_i\|/V_j \tag{24}$$

and note that $\tau_{ij}(\mathbf{u}_k, t_k)$ is the earliest time that node j can reach target i under the condition that it starts at t_k with control dictated by $\mathbf{u}_k$ and then proceeds directly from $x_j(t_k + H_k)$ to the target at y_i. We are interested in the maximal reward that node j can extract from target i if it reaches the target at time $\tau_{ij}(\mathbf{u}_k, t_k)$. Clearly, this is given by $R_i\phi_i[\tau_{ij}(\mathbf{u}_k, t_k)]$. For convenience, define

$$\tilde{\phi}_{ij}(\mathbf{u}_k, t_k) = \phi_i[\tau_{ij}(\mathbf{u}_k, t_k)] \tag{25}$$

where it is worth pointing out that $\tilde{\phi}_{ij}(\cdot)$, unlike $\phi_i(\cdot)$, depends on both i and j. It is also clear that the probability of extracting this reward, evaluated at time t_k, is given by $p_{ij}[\tau_{ij}(\mathbf{u}_k, t_k)]$. For convenience, we set

$$\tilde{p}_{ij}(\mathbf{u}_k, t_k) = p_{ij}[\tau_{ij}(\mathbf{u}_k, t_k)] \tag{26}$$

Returning to the function $q_{ij}(\delta_{ij}) \equiv q_j(y_i,\delta_j)$ defined earlier, we are interested in its value at $t = t_k + H_k$ and define

$$\tilde{q}_{ij}(\mathbf{u}_k, t_k) = q_{ij}[\delta_{ij}(t_k + H_k)] \tag{27}$$

Using the notation $\mathcal{A}_k$ and $\mathcal{T}_k$ to denote dependence of these sets on t_k, we can now present the optimization

problem $\mathbf{P}_k$, formulated at time t_k, as follows:

$$\max_{\mathbf{u}_k} \sum_{i\in\mathcal{T}_k}^{N} \sum_{j\in\mathcal{A}_k}^{M} R_i \tilde{\phi}_{ij}(\mathbf{u}_k, t_k) \cdot \tilde{p}_{ij}(\mathbf{u}_k, t_k) \cdot \tilde{q}_{ij}(\mathbf{u}_k, t_k) \tag{28}$$

with $\tilde{\phi}_{ij}(\mathbf{u}_k, t_k)$, $\tilde{p}_{ij}(\mathbf{u}_k, t_k)$ and $\tilde{q}_{ij}(\mathbf{u}_k, t_k)$ as defined in (25), (26) and (27), respectively. The expression $R_i\tilde{\phi}_{ij}(\mathbf{u}_k, t_k) \cdot \tilde{p}_{ij}(\mathbf{u}_k, t_k) \cdot \tilde{q}_{ij}(\mathbf{u}_k, t_k)$ in (28) can be seen as the *expected reward* that node j collects from target i, evaluated at time t_k using a planning horizon H_k.

Problem $\mathbf{P}_k$ is parameterized by the planning horizon H_k, which is critical in obtaining desirable properties for this CRH controller. In particular we will set

$$H_k = \min_{j\in\mathcal{A}_k, i\in\mathcal{T}_k, p_{ij}(t_k)>0} \{\|y_i - x_j(t_k)\|/V_j\} \tag{29}$$

that is, the smallest "distance" (in time units) between any target and any capable node at time t_k. It is shown in Ref. [38] that this choice is crucial to ensure that CRH trajectories always converge to targets. Moreover, from our definition of a node "visiting" a target, it follows that $H_k > \min_{j\in\mathcal{A}_k, i\in\mathcal{T}_k, p_{ij}(t_k)>0}\{s_i/V_j\}$ where $s_i > 0$ is the size of i. At the time of a visit, either the task at the target is completed and the target is removed from the set $\mathcal{T}_k$ or the node depletes its resources (i.e., $p_{ij}(t_k) = 0$); the planning horizon is also re-evaluated at this point.

Upon getting the optimal $\mathbf{u}_k$ for (28) based on H_k and all state information available at t_k, all nodes follow this control for an *action horizon* $h_k \leq H_k$. The process is then repeated at time $t_{k+1} = t_k + h_k$, $k = 0, 1, \ldots$. The value of h_k is determined by two factors. First, if an unexpected event takes place at some $t_e \in (t_k, t_k + h_k)$, then we set $h_k = t_e - t_k$. Otherwise, we simply update the control after a prespecified amount of time. Thus, in general, $\{t_k\}$ is a random sequence. The CRH controller terminates when (1) all the target rewards are collected, (2) all nodes are eliminated, (3) nodes deplete all their resources or (4) the mission time expires.

As far as computation is concerned, the complexity of this approach is independent of N and several orders of magnitude more efficient than a discrete assignment algorithm where each node i is assigned to a sequence of targets, as discussed in Ref. [38]. The main question, however, is whether CRH trajectories are guaranteed to visit targets, since nodes are never explicitly assigned to them. This property is referred to as trajectory *stationarity*. Specifically, for a trajectory $X(t) = \{\mathbf{x}(t) : \mathbf{x}(t) = [x_1(t) \cdots x_M(t)]\}$, if there exists some $t_v < \infty$, such that $\|x_j(t_v) - y_i\| \leq s_i$, $i \in \mathcal{T}$, $j \in \mathcal{A}$, then $\mathbf{x}(t)$ is a *stationary trajectory*, and we say that the trajectory $x_j(t)$ *converges* to target i. A key contribution of Ref. [38] is to show that the CRH controller indeed generates stationary trajectories under a certain condition which characterizes a potential function derived from the cost function in (28). It is, however, difficult to show that this condition is satisfied for all N, M. For the 1-node N-target case, it can be shown that this condition reduces to a simple test. For 2-node problems, it is still possible to show analytically that this condition is satisfied for one or two targets. Beyond these cases, the analysis becomes highly complicated, but simulation results illustrate the effectiveness of this approach for very large values of N, M. More recently, a distributed version of the CRH controller developed in Ref. [39] has also been successfully implemented in a laboratory setting with small robots playing the role of mobile nodes. Figure 6 shows a picture of this setting with three wireless robots performing a reward maximization mission involving eight targets (the color coding denotes different rewards and deadlines).

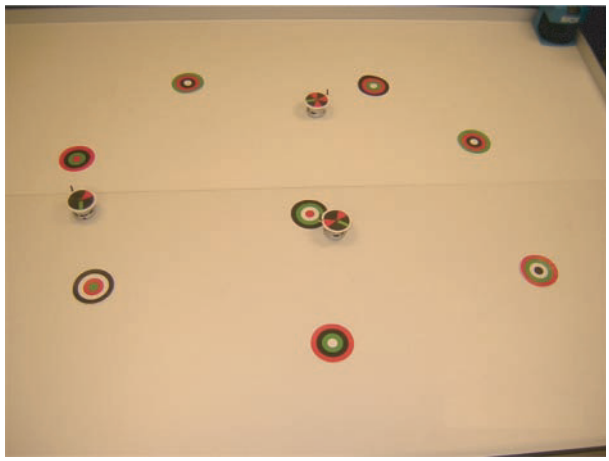

Fig. 6. Laboratory setting for various reward maximization missions.

Example. Figure 7 shows a reward maximization mission example with four nodes (P, G, BK, BL, all initially located at the same point), where only a single target is initially known (labeled 1). The remaining points indicated are unknown to the network, but can be detected if a node happens to be within a "sensor detection area" (in this case, a circle of radius 10/3 units). Thus, an essential aspect of the mission is the search for targets. The implementation of the controller requires selecting the planning and action horizon H_k and h_k, respectively. In what follows, H_k is given by (29) and h_k is selected so that

$$h_k = \begin{cases} \frac{1}{2}H_k & \text{if } H_k > r \\ H_k & \text{if } H_k \leq r \end{cases} \tag{30}$$

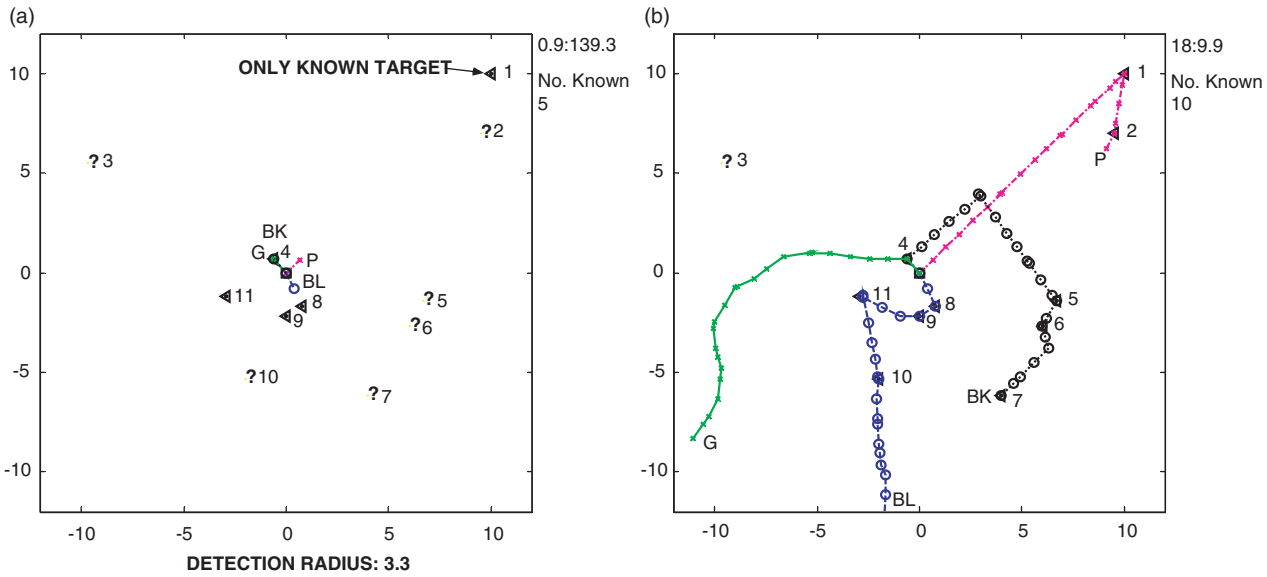

Fig. 7. An example of a mission with unknown targets.

where, for any target point, r is a small distance from it (in the sequel, $r = 2$). The reward discounting function $\phi_i(t)$ is given by (20), $q_{ij}(\delta_{ij})$ is given by (22) and $\delta_{ij} = \delta_j(y_i, t)$ is defined by (21) with $b = 2$.

In this case, the CRH controller serves to disperse the four nodes so that they are headed in different directions. In Fig. 7(a), because targets 4, 8, 9, 11 are located within the detection radius defined by the initial location, they become known to the network (it is assumed that nodes can fully communicate among them). Vehicles BK, G and BL are headed for these targets, while vehicle P is heading for target 1. In Fig. 7(b) all, but one, target points have been detected. An example of the controller's cooperative nature is seen in the behavior of BK: it was originally accompanying P towards 1, up to a point where the effect of $q_{ij}(\delta_{ij})$ in (28) is such that the expected reward is maximized by having node BK be "repelled", which is tantamount to this node searching for unknown targets and in fact detecting some as shown in Fig. 7(b) relative to Fig. 7(a). Along the same lines, node G, after visiting target 4, is directed away from the areas that the other three nodes are covering. Although this node fails to detect any of the remaining unknown targets, it will continue in "search mode" during the mission. However, node BL visits a number of targets that it happens to detect along its trajectory.

4.2. Coverage Control Missions

As we have already seen in the deployment problem considered in Section 3.1, the performance of a sensor network is sensitive to the location of its nodes in the mission space. When nodes are mobile and data source targets are either unknown or are mobile as well, the problem of deploying sensors in order to meet the overall system objectives is referred to as the *coverage control* or *active sensing* problem [14,46,47]. In particular, sensors must be deployed so as to maximize the information extracted from the mission space while maintaining acceptable levels of communication and energy consumption. The *static* version of this problem involves positioning sensors without any further mobility; optimal locations can be determined by an off-line scheme which is akin to the widely studied facility location optimization problem. The *dynamic* version allows the coordinated movement of sensors, which may adapt to changing conditions in the mission space, typically deploying them into geographical areas with the highest information density.

Owing to the similarity of coverage control with facility location optimization, the problem is often viewed in that framework. In Ref. [14], the authors develop a decentralized coverage control algorithm which is based on Voronoi partitions and the Lloyd algorithm. In Ref. [46] a coverage control scheme is proposed which aims at the maximization of target exposure in some surveillance applications, while in Ref. [72] a heuristic algorithm based on "virtual forces" is applied to enhance the coverage of a sensor network. Much of the active sensing literature Ref. [47] also concentrates on the problem of tracking specific targets using mobile sensors and the Kalman filter is extensively used to process observations and generate estimates.

Some of the methods that have been proposed for coverage control assume uniform sensing quality and an unlimited sensing range. Partition-based deployment methods, however, tend to overlook the fact that the overall sensing performance may be improved by sharing the observations made by multiple sensors. There are also efforts which rely on a centralized controller to solve the coverage control problem. As we have already discussed, a centralized approach does not suit the distributed communication and computation structure of sensor networks. In addition, the combinatorial complexity of the problem constrains the application of such schemes to limited-size sensor networks. Finally, another issue that appears to be neglected is the cost of relocating sensors. The movement of sensors not only impacts sensing performance but also it influences other quality-of-service aspects in a sensor network, especially those related to wireless communication: because of the limited on-board power and computational capacity, a sensor network is not only required to sense but also to collect and transmit data as well. For this reason, both sensing quality and communication performance need to be jointly considered when controlling the deployment of sensors.

This motivates a distributed coverage control approach for cooperative sensing [41], which we will present in this section. The mission space will now be modeled using a density function representing the frequency that specific events take place (e.g., data are generated at a certain point). At the two extremes, this allows us to model a mission space with no information on target locations (using a uniform density function) or one with known locations (using a probability mass function). We assume that a mobile sensor node has a limited range which is defined by a probabilistic model. A deployment algorithm is applied at each mobile node such that it maximizes the joint detection probabilities of random events. We assume that the event density function is fixed and given; however, in the case that the mission space (or our perception of the mission space) changes over time, the adaptive relocation behavior naturally follows from the optimal coverage formulation.

4.2.1. Mission Space and Sensor Model

We model the mission space as a polyhedron $\Omega \subset \mathbb{R}^2$, over which there exists an event density function $R(x), x \in \Omega$, that captures the frequency or density of a specific random event taking place (in $\mathrm{Hz/m^2}$). $R(x)$ satisfies $R(x) \geq 0$ for all $x \in \Omega$ and $\int_\Omega R(x) < \infty$. Depending on the application, $R(x)$ may be the frenquency that a certain type of data source appears at x, or it could be the pobability that a variable sensed (e.g., temperature) at x exceeds a specific threshold. In the mission space Ω, there are N mobile nodes located at $\mathbf{s} = (s_1, \ldots, s_N)$, $s_i \in \mathbb{R}^2$, $i = 1, \ldots, N$. When an event occurs at point x, it emits a signal and this signal is observed by a sensor node at location s_i. The received signal strength generally decays with $\|x - s_i\|$, the distance between the source and the sensor. Similar to the model in Ref. [16], we represent this degradation by a monotonically decreasing differentiable function $p_i(x)$, which expresses the probability that sensor i detects the event occurring at x.

As an example, if we assume signal strength declines polynomially with distance and taking into consideration environmental noise, the signal strength received at s_i is expressed by $E_i(x) = E/(\|x - s_i\|^n) + \eta_i$, where E is the total energy emitted when an event takes place, η_i is the noise and n is a decay coefficient (typically selected so that $2 \leq n \leq 5$). If a sensor detects an event when E_i is beyond some threshold, then $p_i(x)$ can be expressed as

$$p_i(x) = \mathrm{Prob}\left[\frac{E}{\|x - s_i\|^n} + \eta_i > c\right] = \mathrm{Prob}\left[\eta_i > c - \frac{E}{\|x - s_i\|^n}\right]$$

With a given probability distribution of noise (e.g., additive white Gaussian noise), this may be used as the sensor model. Alternatively, a sensor model with a simpler form may be

$$p_i(x) = p_{0i} \mathrm{e}^{-\lambda_i \|x - s_i\|} \tag{31}$$

where the detection probability declines exponentially with distance, and p_{0i}, λ_i are determined by physical characteristics of the sensor.

4.2.2. Optimal Coverage Problem Formulation and Distributed Control

When deploying mobile sensor nodes into the mission space, we want to maximize the probability that events are detected. This motivates the formulation of an optimal coverage problem. Assuming that sensors make observations independently, when an event takes place at x and it is observed by sensor nodes, the joint probability that this event is detected can be expressed by

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^{N} [1 - p_i(x)] \tag{32}$$

The optimal coverage problem can be formulated as a maximization of the expected event detection frequency by the sensor nodes over the mission space Ω:

$$\max_{\mathbf{s}} \int_{\Omega} R(x)P(x,\mathbf{s})\,\mathrm{d}x \tag{33}$$

In this optimization problem, the controllable variables are the locations of mobile sensors in the vector $\mathbf{s}$. The problem may be solved by applying a nonlinear optimizer with an algorithm that can evaluate integrals numerically. In this case, a centralized controller with substantial computational capacity is required. In a mobile sensor network, the basestation is a likely candidate for such a controller. However, this solution is only suitable for networks of limited size. Otherwise, both the complexity of the optimization problem and the communication overhead will make this centralized scheme infeasible.

Thus, instead of using a centralized scheme, we will develop a distributed control method to solve the optimal coverage problem. We denote the objective function in (33) by

$$F(\mathbf{s}) = \int_{\Omega} R(x)P(x,\mathbf{s})\,\mathrm{d}x \tag{34}$$

When taking partial derivatives with respect to s_i, $i = 1,\ldots,N$, we have

$$\frac{\partial F}{\partial s_i} = \int_{\Omega} R(x)\frac{\partial P(x,\mathbf{s})}{\partial s_i}\,\mathrm{d}x \tag{35}$$

If this partial derivative can be evaluated locally by each mobile node i, then a gradient method can be applied which directs nodes towards locations that maximize $F(\mathbf{s})$. In view of (32), the partial derivative (35) can be rewritten as

$$\frac{\partial F}{\partial s_i} = \int_{\Omega} R(x)\prod_{k=1,k\neq i}^{N}[1-p_k(x)]\frac{dp_i(x)}{dd_i(x)}\frac{s_i-x}{d_i(x)}\mathrm{d}x \tag{36}$$

where $d_i(x) \equiv \|x - s_i\|$. It is hard for a mobile sensor node to directly compute (36), since it requires global information such as the value of $R(x)$ over the whole mission space and the exact locations of all other nodes. In addition, the evaluation of integrals remains a significant task for a sensor node to carry out. To address these difficulties, we first truncate the sensor model and constrain its sensing capability by applying a *sensing radius*. This approximation is based on the physical observation that when $d_i(x)$ is large, $p_i(x) = 0$ for most sensing devices. Let

$$p_i(x) = 0, \frac{dp_i(x)}{dd_i(x)} = 0 \quad \text{for all } x \text{ s.t. } d_i(x) \geq D \tag{37}$$

where D denotes the sensing radius. Thus, (37) defines node i's region of coverage, which is represented by $\Omega_i = \{x : d_i(x) \leq D\}$. Since $p_i(x) = 0$, $dp_i(x)/dd_i(x) = 0$ for all $x \notin \Omega_i$, we can use Ω_i to replace Ω in (36). Another byproduct of using (37) is the emergence of the concept of *neighbors*. In (36), for a point $x \in \Omega_i$ and a node $k \neq i$, a necessary condition for the detection probability $p_k(x)$ to be greater than 0 is $d_k(x) \leq D$. As shown in Fig. 8, when the distance between nodes i and k is greater than $2D$, every point x in Ω_i satisfies $d_k(x) > D$; thus, $p_k(x) = 0$ and $[1 - p_k(x)] = 1$ for all $x \in \Omega_i$. If we define a set $\mathcal{B}_i = \{k : \|s_i - s_k\| < 2D,\ k = 1,\ldots,N,\ k \neq i\}$, then, any sensor node $k \notin \mathcal{B}_i\ (k \neq i)$ will not contribute to the integral in (36).

After applying (37) and using $\mathcal{B}_i$, (36) reduces to

$$\frac{\partial F}{\partial s_i} = \int_{\Omega_i} R(x)\prod_{k\in\mathcal{B}_i}[1-p_k(x)]\frac{dp_i(x)}{dd_i(x)}\frac{s_i-x}{d_i(x)}\,\mathrm{d}x \tag{38}$$

The final step in making (38) computable is to discretize the integral evaluation. As shown in Fig. 9, a grid is applied over the coverage region Ω_i. Thus, Ω_i is represented by a $(2V+1)\times(2V+1)$ grid with $V = \lfloor D/\Delta \rfloor$, where $\Delta \ll D$ is the resolution of the grid. On the grid of each node i, a Cartesian coordinate system is defined, with its origin located at s_i, its axes parallel to the grid's setting, and the unit length being Δ. In this local coordinate system, let (u, v) denote the location of a point x. Then, the transformation that maps (u, v) onto the global coordinate

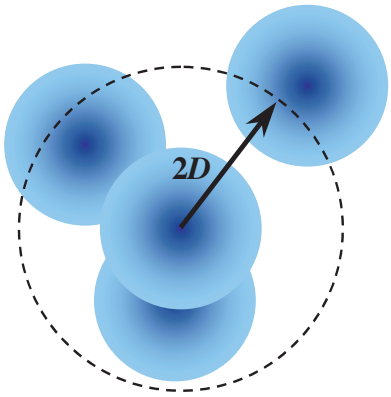

Fig. 8. Defining neighbor sets.

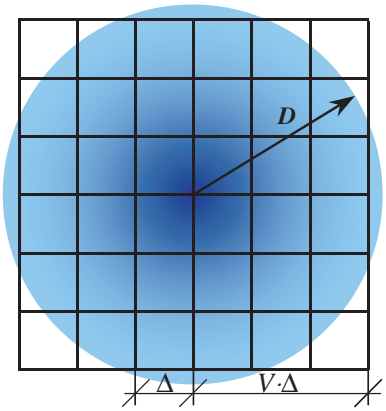

Fig. 9. Discretization using a grid.

system is $x = s_i + [\,u\Delta \quad v\Delta\,]^{\mathrm{T}}$. Upon switching to this local coordinate system, the terms in (38) become

$$R(x) = \tilde{R}_i(u,v), \;\; p_i(x) = \tilde{p}_i(u,v), \;\; \frac{dp_i(x)}{dd_i(x)} = \tilde{p}_i'(u,v)$$

where $\tilde{R}_i(u,v)$ indicates node i's local perception (map) on the event density of the mission space. In a typical dynamic deployment application, all sensor nodes start with the same copy of an estimated event density function at the beginning of the deployment. As nodes are deployed and data are collected, an individual node may update its local map through merging new observations into its perception, and by exchanging information with nearby neighbors.

We also rewrite the product term in (38) as

$$\prod_{k\in\mathcal{B}_i}[1-p_k(x)] = \prod_{k\in\mathcal{B}_i}\left[1-\tilde{p}_k\left(u-\frac{s_{k1}-s_{i1}}{\Delta}, v-\frac{s_{k2}-s_{i2}}{\Delta}\right)\right] \equiv \tilde{B}_i(u,v)$$

where $(u-((s_{k1}-s_{i1})/\Delta), v-((s_{k2}-s_{i2})/\Delta))$ are the coordinates of x in the k-th node's local coordinate system. By applying the grid and the coordinate tranformation, (38) can be rewritten as

$$\frac{\partial F}{\partial s_{i1}} \approx \Delta^2 \sum_{u=-V}^{V}\sum_{v=-V}^{V} \frac{\tilde{R}_i(u,v)\tilde{B}_i(u,v)\tilde{p}_i'(u,v)u}{\sqrt{u^2+v^2}}$$
$$\frac{\partial F}{\partial s_{i2}} \approx \Delta^2 \sum_{u=-V}^{V}\sum_{v=-V}^{V} \frac{\tilde{R}_i(u,v)\tilde{B}_i(u,v)\tilde{p}_i'(u,v)v}{\sqrt{u^2+v^2}} \tag{39}$$

These derivatives can be easily computed by mobile sensor nodes using only the local information available. An advantage of switching to the local coordinates in (39) is that the values of $\tilde{p}_i(u,v)$ and $\tilde{p}_i'(u,v)$ are uniquely determined by (u, v) and the sensor model. This motivates the storage of $\tilde{p}_i(u,v)$ and $\tilde{p}_i'(u,v)$ as two matrices in the on-board memory of a sensor node. Through acquiring key sensor model parameters [e.g., p_{0k} and λ_k in (31)] from neighbors and properly rescaling $\tilde{p}_i(u,v)$ and $\tilde{p}_i'(u,v)$, node i can also easily evaluate $\tilde{B}_i(u,v)$ using stored matrices. By doing so, the computation effort in repeatedly evaluating (39) is drastically reduced.

The gradient information above provides a direction for a mobile node's movement. The precise way in which this information is used depends on the choice of motion scheme. The most common approach in applying a gradient method is to determine the next waypoint on the i-th mobile node's motion trajectory through

$$s_i^{k+1} = s_i^k + \alpha_k \frac{\partial F}{\partial s_i^k} \tag{40}$$

where k is an iteration index, and the step size α_k is selected according to standard rules [5] in order to guarantee the convergence of motion trajectories.

The computational complexity in evaluating the gradient shown in (39) depends on the scale of the grid and the size of neighbor set $\mathcal{B}_i$. In the worst case, node i has $N-1$ neighbors and the number of operations needed to compute $\partial F/\partial s_i$ is $O(NV^2)$. The best case occurs when there is no neighbor for node i, and the corresponding complexity is $O(V^2)$. In both cases, the complexity is quadratic in V.

4.2.3. Optimal Coverage Problem with Communication Costs

Besides sensing and collecting data, the coverage control mission includes the task of forwarding data to the basestation. Assuming a flat network structure (i.e., no clusterheads as discussed in Section 2), the cost of communication comes from the power consumption for wireless transmissions. We shall use once again the link energy models (6) and (7) of Section 3.1. The basestation location is represented as $s_0 \in \mathbb{R}^2$ and the data rate originating from the i-th sensor node is denoted by $r_i(s_i)$, $i = 1, \ldots, N$. Note that r_i is defined as a function of s_i because the amount of data forwarded at i is determined by the number of events detected, and the latter depends on the node's location. Here we assume that $r_i(s_i)$ is proportional to the frequency that events are detected, that is

$$r_i(s_i) = \alpha_3 \int_{\Omega} R(x)p_i(x)\,dx \tag{41}$$

where α_3 (bits/detection) is the amount of data forwarded when the sensor node detects an event. Let $c_i(\mathbf{s})$ be the *total* energy consumed by the network in order to deliver a bit of data from node i to the basestation. Then, the optimal coverage problem can be revised by combining sensing coverage and communication cost as follows:

$$\max_{\mathbf{s}} \left\{ w_1 \int_{\Omega} R(x) P(x, \mathbf{s}) \, \mathrm{d}x - w_2 \sum_{i=1}^{N} r_i(s_i) c_i(\mathbf{s}) \right\} \tag{42}$$

where w_1, w_2 are weighting factors. One can think of w_1 as the reward for detecting an event and w_2 as the price of consuming a unit of energy.

Let us denote the communication cost by

$$G(\mathbf{s}) = \sum_{i=1}^{N} r_i(s_i) c_i(\mathbf{s}) \tag{43}$$

so that, recalling (34), the overall objective function is written as

$$J(\mathbf{s}) = w_1 F(\mathbf{s}) - w_2 G(\mathbf{s})$$

In order to derive partial derivatives $\partial J / \partial s_i$ as performed earlier, we shall focus on the evaluation of $\partial G / \partial s_i$, which can be expressed as

$$\frac{\partial G}{\partial s_i} = c_i(\mathbf{s}) \frac{dr_i(s_i)}{ds_i} + \sum_{k=1}^{N} r_k(s_k) \frac{\partial c_k(\mathbf{s})}{\partial s_i} \tag{44}$$

In this expression, both r_i and $\partial r_i / \partial s_i$ can be obtained by applying the same method as before. That is, recalling that $x = s_i + [\, u\Delta \quad v\Delta \,]^{\mathrm{T}}$,

$$\begin{aligned} r_i &\approx \alpha_3 \Delta^2 \sum_{u=-V}^{V} \sum_{v=-V}^{V} \tilde{\boldsymbol{R}}(u, v) \, \tilde{\boldsymbol{p}}_i(u, v) \\ \frac{dr_i}{ds_{i1}} &\approx \alpha_3 \Delta^2 \sum_{u=-V}^{V} \sum_{v=-V}^{V} \frac{\tilde{\boldsymbol{R}}(u, v) \, \tilde{\boldsymbol{p}}_i'(u, v) u}{\sqrt{u^2 + v^2}} \\ \frac{dr_i}{ds_{i2}} &\approx \alpha_3 \Delta^2 \sum_{u=-V}^{V} \sum_{v=-V}^{V} \frac{\tilde{\boldsymbol{R}}(u, v) \, \tilde{\boldsymbol{p}}_i'(u, v) v}{\sqrt{u^2 + v^2}} \end{aligned} \tag{45}$$

The only term remaining to derive in (44) is $c_i(\mathbf{s})$ and its gradient. The cost of delivering a bit of data from i to the basestation, $c_i(\mathbf{s})$, is determined by the way in which data forwarding paths are constructed, i.e., the precise routing mechanism used. The issue of routing was discussed in Section 3.2, where we saw that a typical shortest-path based routing scheme must specify at each node i a forward index h_i, a forward cost c_i, an upstream vector $U_i = (u_1^i, \ldots, u_N^i)$ and a cumulative flow factor z_i (see Section 3.2). Given h_i, c_i, U_i and z_i, a node i can evaluate $\partial G / \partial s_i$ locally. To accomplish this, let us rewrite $G(\mathbf{s})$ in (43) as

$$\begin{aligned} G(\mathbf{s}) &= \sum_{i=1}^{N} r_i(s_i) c_i(\mathbf{s}) = \sum_{i=1}^{N} r_i \sum_{(j,k) \in l_i} e_{jk} \\ &= \sum_{i=1}^{N} \sum_{(j,k) \in \mathcal{E}} \mathbf{1}[(j,k) \in l_i] r_i e_{jk} \\ &= \sum_{(j,k) \in \mathcal{E}} e_{jk} \left\{ \sum_{i=1}^{N} \mathbf{1}[(j,k) \in l_i] r_i \right\} \end{aligned} \tag{46}$$

In the last expression above, $\sum_{i=1}^{N} \mathbf{1}[(j,k) \in l_i] r_i$ is actually equivalent to the total flow on link (j,k). Because of the tree structure of the network, we have

$$\sum_{i=1}^{N} \mathbf{1}[(j,k) \in l_i] r_i = \begin{cases} z_j & \text{if } k = h_j \\ 0 & \text{otherwise} \end{cases}$$

By removing all the terms with value zero in (42), we get

$$G(\mathbf{s}) = \sum_{i=1}^{N} e_{ih_i} z_i \tag{47}$$

Using (47), the term $\sum_{k=1}^{N} r_k \frac{\partial c_k}{\partial s_i}$ in (44) is equivalent to

$$\sum_{k=1}^{N} r_k \frac{\partial c_k}{\partial s_i} = \sum_{k=1}^{N} z_k \frac{\partial e_{kh_k}}{\partial s_i}$$

where we assume that network routing remains fixed when sensor locations (i.e., $\mathbf{s}$) change slightly. Using (7) and (8),

$$\frac{\partial e_{kh_k}}{\partial s_i} = \begin{cases} n\alpha_2 \left\| s_i - s_j \right\|^{(n-2)} (s_i - s_j) & \text{if } k = i \text{ or } h_k = i \\ 0 & \text{otherwise} \end{cases}$$

Thus,

$$\sum_{k=1}^{N} r_k \frac{\partial c_k}{\partial s_i} = \left[z_i \frac{\partial e_{ih_i}}{\partial s_i} + \sum_{\{j | u_j^i = 1\}} z_j \frac{\partial e_{jh_j}}{\partial s_i} \right] \tag{48}$$

By combining (39), (44), (45) and (48), sensor node i can derive $\partial J / \partial s_i$ locally. Then, each sensor uses gradient information to direct motion control as in (40) with $\partial J / \partial s_i^k$ replacing $\partial F / \partial s_i^k$. With properly selected step sizes, mobile sensors will finally converge to a maxmium point of $J(s)$.

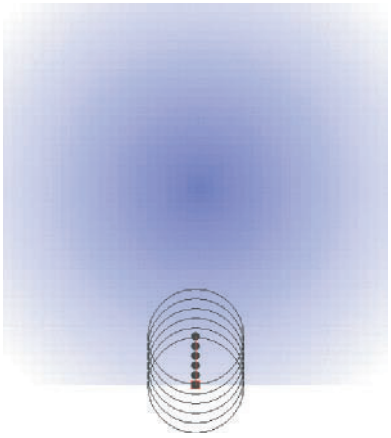

Fig. 10. Cooperative coverage control problem with six mobile sensor nodes.

4.2.4. Simulation Examples

The distributed deployment algorithm above has been implemented in a Java-based simulation environment. In the example shown in Fig. 10, a team of six mobile sensors is waiting to be deployed into a 40×40 (meter) mission space. The event density function $R(x)$ is given by,

$$R(x) = R_0 - \beta \|x - x_0\| \tag{49}$$

where $R_0 = 3.0,\ \beta = 0.1,\ x_0 = [0, 20]$. According to (49), the event density of a point $x \in \Omega$ declines linearly with the distance between x and the center point x_0 of the mission space.

At time $t = 0$, mobile sensor nodes reside near the origin of the mission space. Each mobile node is equipped with a sensor whose detection probability is modeled as in (31) by $p_i(x) = p_{0i} e^{-\lambda_i \|x - s_i\|}$ where $p_{0i} = 1.0,\ \lambda_i = 1.0$ for all $i = 1, \ldots, N$. The sensing radius is $D = 5.0$, as illustrated by black cirlces in Fig. 10. A mobile node also has a wireless transeiver whose energy consumption is modeled by (7) with $\alpha_1 = 0.01$ nJ/bit, $\alpha_2 = 0.001$ nJ/bit/m^4 and $n = 4$. In the mission space, there is a radio basestation residing at $s_0 = [0, 0]$ (marked by a red square in Fig. 10). Upon a sensor detecting an event, it collects 32 bits of data and forwards them back to the basestation [so that $\alpha_3 = 32$ in (41)].

We will present simulation results for this coverage control problem by looking at two distinct cases. In the first case, no communication cost is considered, which corresponds to $w_1 > 0,\ w_2 = 0$ in the optimal coverage formulation (42). In the second case, both sensing coverage and communication cost are included ($w_1,\ w_2 > 0$).

Figure 11 presents several snapshots taken during the deployment process of the first case. Starting with Fig. 11(a), six nodes establish a formation and move towards the center of the mission space. During its movement, the formation keeps evolving, so that nodes expand the overall area of sensing and at the same time jointly cover the points with high event density. In addition, nodes also maintain wireless communication with the basestation. This is shown in Fig. 11 as links between sensor nodes and the basestation. The team of nodes finally converges to a stationary formation as shown in Fig. 11(d). It can be seen in this symmetric formation that all six nodes are jointly sensing the area with the highest event density.

We incoporate communication cost into the optimal coverage formulation by setting $w_2 = 0.0008$ and $w_1 = 1 - w_2$ in (42). The corresponding deployment simulation results are shown in Fig. 12. Comparing with the first case, a critical difference can be observed in the network formation: nodes not only move towards the area with high event density but also they maintain an economical multi-hop path to the basestation. The network reaches a stationary deployment as illustrated in Fig. 12(d). In contrast to the final formation of the first case [Fig. 11(d)], only four nodes gather around the center of the mission space. The other two nodes are aligned as relays to support the communication with the basestation.

Figures 13 and 14 demonstrate the sensing coverage and communication cost associated with the previously shown two cases. Figure 13 depicts the change in sensing coverage (measured by the expected frequency of event detection) when nodes move towards the optimal deployment. A direct observation is that in both cases, sensing coverage increases monotonically with the evolution of formations. If no communication cost is considered during deployment, sensing coverage reaches a maximum at 91.47 Hz. However, in the case that communication cost is considered, when sensors reach optimal deployment, only 84.74 events can be detected per second, which corresponds to a 7.36% coverage loss. This coverage loss is natural, since the optimal coverage formulation (42) actually trades off sensing coverage for a lower communication cost. This tradeoff can be further examined by looking at Fig. 14. If communication cost is considered, the final power consumption is 8.01×10^3 nW. Compared with the communication cost of the first case (1.877×10^5 nW), there is a 95.73% power saving. One issue that we have not explicitly addressed in the development of this distributed cooperative coverage control approach is that of the global optimality of the gradient-based algorithm involved. This remains a topic of ongoing research. In particular, one expects

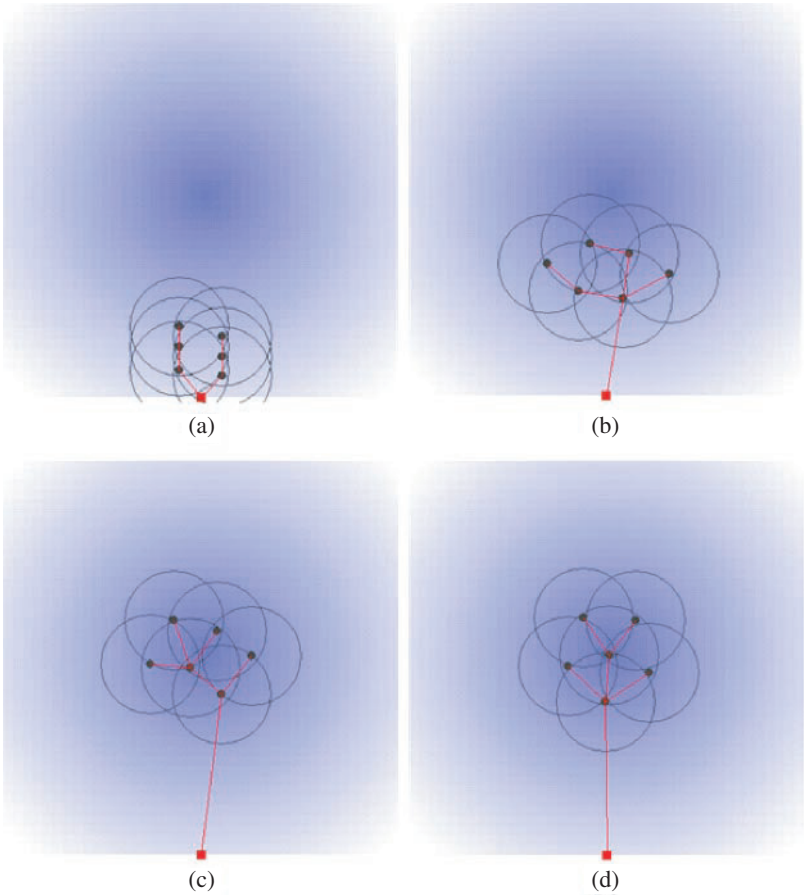

Fig. 11. Sensor node deployment without communication cost consideration.

that global optimality is intrically connected to properties of the event density function and the sensor model adopted.

5. Future Research Directions

We have attempted in this paper to categorize and describe major design and control problems related to sensor networks, as well as to cooperative control as it pertains to sensor networks with moblie nodes. For some of these problems, we have also presented selected solution approaches, some already published in the literature and some novel. Open research issues were mentioned or briefly hinted at throughout the previous sections and we summarize some of them here. First, we mentioned in Section 2 that a potentially better structure for sensor networks is a hierarchical one, making use of clusterheads acting as intermediate processing nodes between data sources and a basestation. The presense of clusterheads implies different approaches for some of the problems we have discussed; for example, deployment may be quite different if a clusterhead can "aggregate" data from neighboring nodes and avoid the need for these nodes to use up energy for direct communication with the basestation. Second, we also briefly mentioned that one form of power control is to switch the state of sensor nodes between "sleeping" and "sensing" or "relaying" (see Section 2). Formulating such a switching control problem and devising solution methods dependent on the information available to each node is an interesting direction for research. Third, the problem of location detection when nodes are mobile is also one that was only briefly mentioned (see Section 4) and it clearly deserves in-depth study. In the context of cooperative control, we saw that one can define different types of sensor network missions, two classes of which were discussed in Section 4. Some of the most pressing technical concerns here are related to the properties of the associated optimization problems involved; particularly, the questions of local versus global optimality and the need for mechanisms consistent with the distributed nature of sensor networks.

Although many of the open questions above are technically challenging in their own right, we would

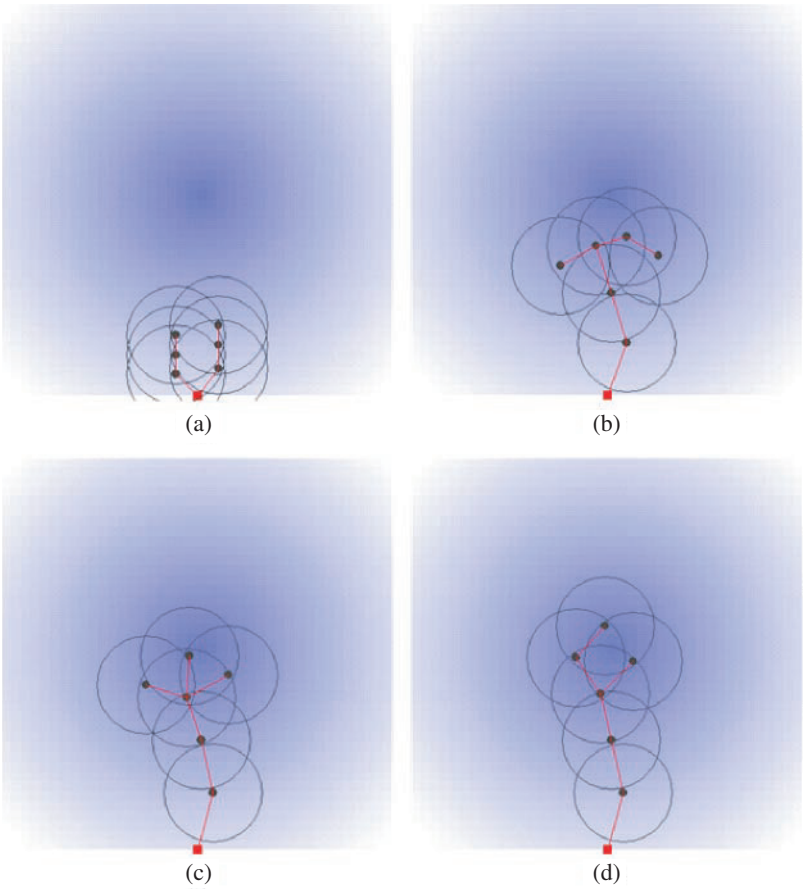

Fig. 12. Sensor node deployment with communication cost consideration.

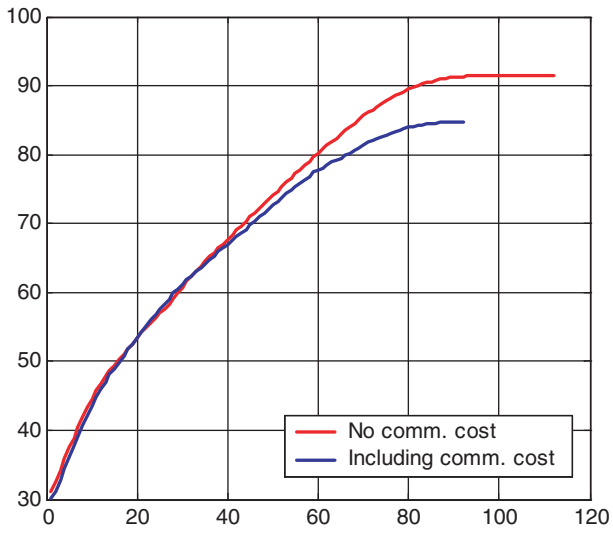

Fig. 13. Comparison of sensing coverage.

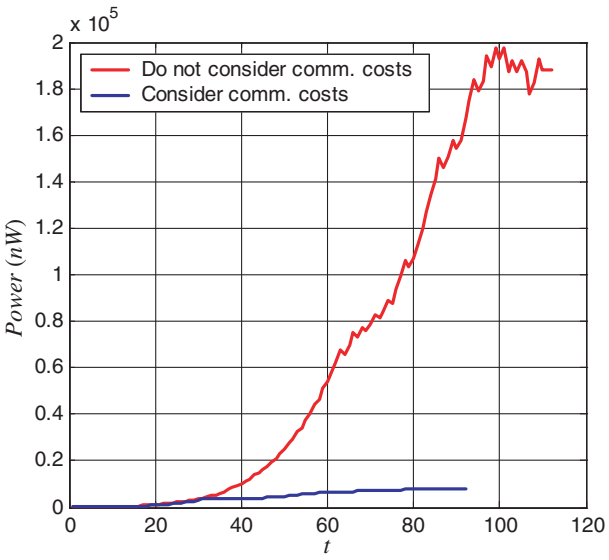

Fig. 14. Comparison of communication costs.

like to bring this section to a close by turning our attention to some more fundamental issues of much broader long-term impact where progress has been minimal. These issues are closely related to the convergence of communication, computing and control, which brings together three disciplines that often use different modeling paradigms and different ways of thinking. Naturally, bridging the gaps between them is a real challenge. One of these issues concerns the

combination of *asynchronous* and *synchronous* modes of operation in a common system setting. Although the gathering of data is inherently asynchronous (owing to multiple sensor nodes operating in different temporal and spatial scales), the processes of data fusion and control are traditionally based on a synchronized time structure. This is one manifestation of the difference between *time-driven* and *event-driven* behavior; designing a system environment where both can coexist remains an open problem. The traditional setting of differential equation models and time-driven digital sampling provides a comfortable infrastructure for communication and control methodologies, but that is being challenged by computational models which rely on event-driven processes and by the simple intuitive observation that time-driven sampling is inherently wasteful. The limited resources of sensor network nodes emphasize the need to switch to a more efficient event-driven sampling approach, where data are collected only when "something interesting" happens. To do so, however, requires new sampling mechanisms and possibly new data collection hardware as well.

A second research issue of particular importance to control theory is the obvious shift from *sensor-poor* to *data-rich* control systems. Traditional feedback control systems have been designed under the premise that sensors are few and expensive and much of the "intelligence" in such systems is concentrated on compensating for limited state information. The sudden wealth of sensor data (subject, of course, to bandwidth and delay limitations) shifts the need for "intelligence" towards processing potentially huge amounts of data and combining *model-based* methodologies with increasingly *data-driven* ones. To date, there appears to be a significant gap between schools of thought advocating one versus the other approach. One would expect that a combination can enable us to exploit advantages of both.

Acknowledgement

Supported in part by the National Science Foundation under Grant DMI-0330171, by AFOSR under gants F49620-01-0056 and F49620-01-1-0348, by ARO under grant DAAD19-01-0610, and by a grant from Honeywell Laboratories.

References

1. Al-Karaki JN, Kamal AE. Routing techniques in wireless sensor networks: a survey. IEEE Trans Wireless Commun 2004; 11(6): 6–28
2. Aydin H, Melhem R, Mossé D, Mejia-Alvarez P. Power-aware scheduling for periodic real-time tasks. IEEE Trans Comput 2003; (to appear)
3. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. IEEE Commun Mag 2002; 40(8): 102–114
4. Bhardwaj M, Chandrakasan A, Garnett T. Upper bounds on the lifetime of sensor networks. In: Proceedings of the IEEE international conference on communications. 2001, pp 785–790
5. Bertsekas DP. Nonlinear programming. Athena Scientific, 1995
6. Baliga G, Graham S, Kumar PR. Middleware and abstractions in the convergence of control with communication and computation. In: Proceedings of the 44th IEEE conference on decision and control, 2005, submitted for publication
7. Bachmayer R, Leonard NE. Vehicle networks for gradient descent in a sampled environment. In: Proceedings of the 41st IEEE conference on decision and control. 2002, pp 112–117
8. Bellingham JS, Tillerson M, Alighanbari M, How JP. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In: Proceedings of the 41st IEEE conference on decision and control. 2002, pp 2816–2822
9. Buttazzo GC. Hard real-time computing systems: predictable scheduling algorithms and applications. Kluwer Academic Publishers, Norwell, MA, 1997
10. Chong C. Sensor networks: evolution, opportunities, and challenges. Proc IEEE 2003; 91: 1247–1256
11. Cho YC, Cassandras CG, Pepyne DL. Forward decomposition algorithms for optimal control of a class of hybrid systems. Int J Robust Nonlinear Control 2001; 11(5): 497–513
12. Cassandras CG, Li W. A receding horizon approach for solving some cooperative control problems. In: Proceedings of the 41st IEEE conference on decision and control. 2002, pp 3760–3765
13. Clough BT. Unmanned aerial vehicles: autonomous control challenges, a researcher's perspective. In: Murphey R, Pardalos PM (eds). Cooperative control and optimization. Kluwer Academic Publishers, 2000, pp 35–53
14. Cortes J, Martinez S, Karatas T, Bullo F. Coverage control for mobile sensing networks. IEEE Trans Robotics Autom 2004; 20(2): to appear
15. Chandler PR, Pachter M, Rasmussen S. UAV cooperative control. In: Proceedings of the 2001 American control conference. 2001, pp 50–55
16. Clouqueur T, Phipatanasuphorn V, Ramanathan P, Saluja K. Sensor deployment strategy for target detection. In: Proceedings of 1st ACM international workshop on wireless sensor networks and applications. Atlanta, GA, 2002, pp 42–48
17. Cruz JB, Simaan MA, Gacic A, Liu Y. Moving horizon game theoretic approaches for control strategies in a military operation. In: Proceedings of the 40th IEEE conference on decision and control. 2001, pp 628–633
18. Chang JH, Tassiulas L. Maximum lifetime routing in wireless sensor networks. IEEE/ACM Trans Networking 2004; 12(4): 609–619
19. Castañón DA, Wohletz JM. Model predictive control for dynamic unreliable resource allocation. In: Proceedings

of the 41st IEEE conference on decision and control. 2002, pp 3574–3579
20. Dunbar WB, Murray RM. Receding horizon control of multi-vehicle formations: a distributed implementation. In: Proceedings of the 43rd IEEE conference on decision and control. 2004, pp 1995–2002
21. Frazzoli E, Bullo F. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In: Proceedings of the 43rd IEEE conference on decision and control. 2004, pp 3357–3363
22. Franco E, Parisini T, Polycarpou MM. Cooperative control of discrete-time agents with delayed information exchange: a receding-horizon approach. In: Proceedings of the 43rd IEEE conference on decision and control. 2004, pp 4274–4279
23. Finke J, Passino KM, Sparks AG. Cooperative control via task load balancing for networked uninhabited autonomous vehicles. In: Proceedings of the 42nd IEEE conference on decision and control. 2003, pp 31–36
24. Ganesan D, Govindan R, Shenker S, Estrin D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. Mobile Comput Commun Rev 2001; 4(5): 11–25
25. Gupta P, Kumar P. Critical power for asymptotic connectivity. In: Proceedings of the 37th IEEE conference on decision and control. 1998, pp 1106–1110
26. Gazi V, Passino KM. Stability analysis of social foraging swarms: combined effects of attractant/repellent profiles. In: Proceedings of the 41st IEEE conference on decision and control. 2002, pp 2848–2853
27. Ganguli A, Susca S, Martinez S, Bullo F, Cortes J. On collective motion in sensor networks: sample problems and distributed algorithms. In: Proceedings of the 44th IEEE conference on decision and control 2005. submitted for publication
28. Heinzelman W, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocols for wireless microsensor networks. In: Proceedings of the Hawaaian international conference on systems science, 2000
29. Heinzelman W. Application-specific protocol architectures for wireless networks. PhD thesis, Massachusetts Institute of Technology, 2000
30. Hong I, Kirovski D, Qu G, Potkonjak M, Srivastava MB. Power optimization of variable-voltage core-based systems. IEEE Trans Comput-Aided Design Integrated Circuits Syst 1999; 18(12): 1702–1714
31. Hu J, Sastry S. Optimal collision avoidance and formation switching on Riemannian manifolds. In: Proceedings of the 40th IEEE conference on decision and control. 2001, pp 1071–1076
32. Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Proceedings of the IEEE/ACM MobiCom, 2000
33. Jonsson J, Lonn H, Shin KG. Non-preemptive scheduling of real-time threads on multi-level-context architectures. In: Proceedings of the IEEE workshop on parallel and distributed real-time systems, vol 1586. Springer Verlag, 1999, pp 363–374
34. Johnson DB, Maltz DA. Dynamic source routing in ad hoc wireless networks. In Imielinski, Korth (eds). Mobile computing, vol 353. Kluwer Academic Publishers, 1996
35. Jeffay K, Stanat DF, Martel CU. On non-preemptive scheduling of periodic and sporadic tasks. In: Proceedings of the IEEE real-time systems symposium. 1991, pp 129–139
36. Karp B, Kung HT. GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th annual international conference on mobile computing and networking 2000, pp 243–254
37. Kim W, Shin D, Yun HS, Kim J, Min SL. Performance comparison of dynamic voltage scaling algorithms for hard real-time systems. In: Proceedings of the RTAS'02. 2002, pp 219–228
38. Li W, Cassandras CG. A Cooperative Receding Horizon Controller for Multi-Vehicle Uncertain Environments. IEEE Trans Autom Control 2004; submitted for publication
39. Li W, Cassandras CG. Centralized and distributed cooperative receding horizon control of autonomous vehicle missions. J Math Comput Model (2005), to appear
40. Li W, Cassandras CG. A minimum-power wireless sensor network self-deployment scheme. In: Proceedings of the IEEE wireless communications and networking conference, 2005
41. Li W, Cassandras CG. Distributed cooperative coverage control of sensor networks. In: Proceedings of the 44th IEEE conference on decision and control. 2005, submitted for publication
42. Lian F, Murray R. Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In: Proceedings of the 41st IEEE conference on decision and control. 2002, pp 3766–3769
43. Lindsey S, Raghavendra CS. Pegasis: power-efficient gathering in sensor information systems. In: Aerospace conference proceedings. IEEE, vol 3, 2002, pp 1125–1130
44. Miao L, Cassandras CG. Optimality of static control policies in some discrete event systems. IEEE Trans Autom Control, to appear.
45. Miao L, Cassandras CG. Receding horizon control for a class of discrete event systems with real-time constraints. In: Proceedings of the 44th IEEE conference on decision and control and European Control Conference, Seville, Spain, 2005, submitted
46. Meguerdichian S, Koushanfar F, Potkonjak M, Srivastava MB. Coverage problems in wireless ad-hoc sensor networks. In: Proceedings of the IEEE INFOCOM. 2001, pp 1380–1387
47. Mihaylova L, Lefebvre T, Bruyninckx H, Gadeyne K. Active sensing for robotics-a survey. In: Proceedings of the 5th international conference on numerical methods and applications Borovets, Bulgaria, 2002, pp 316–324
48. Mayne DQ, Michalska L. Receding horizon control of nonlinear systems. IEEE Trans Autom Control 1990; AC-35(7): 814–824
49. Mhatre VP, Rosenberg C, Kofman D, Mazumdar R, Shroff N. A minimum cost heterogeneous sensor network with a lifetime constraint. IEEE Trans Mobile Comput 2005; 4(1): 4–15
50. Mao JF, Zhao QC, Cassandras CG. Optimal dynamic voltage scaling in power-limited systems with real-time constraints. In: Proceedings of the 43rd IEEE conference on decision and control, December 2004, pp 1472–1477

51. Perkins CE, Bhagwat P. Highly dynamic destination-sequenced distance-vector (DSDV) routing for mobile computers. In: Proceedings of the ACM SIGCOMM. 1994, pp 234–244
52. Pering T, Burd T, Brodersen R. Dynamic voltage scaling and the design of a low-power microprocessor system. In: Proceedings of the power driven microarchitecture workshop (ISCA98), 1998
53. Park VD, Corson MS. A highly adaptive distributed routing algorithm for mobile wireless networks. In: Proceedings of the IEEE INFOCOM. 1997, pp 1405–1413
54. Pouwelse J, Langendoen K, Sips H. Dynamic voltage scaling on a low-power microprocessor. In: Proceedings of the 7th annual international conference on mobile computing and networking. 2001, pp 251–259
55. Paschalidis I Ch, Lai W, Starobinski D. On maximizing the utility of uplink transmissions in sensor networks under explicit fairness constraints. In: Proceedings of the 43rd IEEE conference on decision and control. 2004, to appear
56. Passino K, Polycarpou M, Jacques D, Pachter M, Liu Y, Yang Y, Flint M, Baum M. Cooperative control for autonomous air vehicles. In Murphey R, Pardalos PM (eds). Cooperative control and optimization. Kluwer Academic Publishers. 2000, pp 233–269
57. Papadimitriou CH, Steiglitz K. Combinatorial optimization: algorithms and complexity. Dover Publications, 1998
58. Richards A, How J. Decentralized model predictive control of cooperating UAVs. In: Proceedings of the 43rd IEEE conference on decision and control. 2004, pp 4286–4291
59. Swaminathan V, Chakrabarty K. Real-time task scheduling for energy-aware embedded system. In: Proceedings of the IEEE real-time systems symposium, work-in-progress session, 2000
60. Shih E, Cho S, Ickes N, Min R, Sinha A, Wang A, Chandrakasan A. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In: Proceedings of the ACM MobiCom 2001. 2001, pp 272–287
61. Singh L, Fuller J. Trajectory generation for a UAV in urban terrain using nonlinear MPC. In: Proceedings of the 2001 American control conference. 2001, pp 2301–2308
62. Shah R, Rabaey J. Energy aware routing for low energy ad hoc sensor networks. In: Proceedings of the IEEE wireless communications and networking conference. 2002
63. Shakkottai S, Srikant R, Shroff N. Unreliable sensor grids: coverage, connectivity and diameter. In: Proceedings of the IEEE INFOCOM. 2003
64. Shannon CE, Weaver W. The mathematical theory of communication
65. Singh S, Woo M, Raghavendra CS. Power-aware routing in mobile ad hoc networks. In: Proceedings of the IEEE/ACM MobiCom. 1998, pp 181–190
66. Uysal-Biyikoglu E, Prabhakar B, Gamal AE. Energy-efficient packet transmission over a wireless link. IEEE/ACM Trans Networking 2002; 10: 487–499
67. Wu X, Cassandras CG. A maximum time optimal control approach to routing in sensor networks. In: Proceedings of the 44th IEEE conference on decision and control. 2005, submitted
68. Wood AD, Stankovic JA. Denial of service in sensor networks. Computer 2002; 35(10)
69. Ye F, Chen A, Liu S, Zhang L. A scalable solution to minimum cost forwarding in large sensor networks. In: Proceedings of the tenth international conference on computer communications and networks. 2001, pp 304–309
70. Yu Y, Govindan R, Estrin D. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, 2001. Available at: http://citeseer.nj.nec.com/yu01geographical.html
71. Yu Y, Krishnamachari B, Prasanna VK. Energy-latency tradeoffs for data gathering in wireless sensor networks. In: Proceedings of the IEEE infocom, Hong Kong, China, 2004
72. Zou Y, Chakrabarty K. Sensor deployment and target localization based on virtual forces. In: Proceedings of the IEEE INFOCOM. 2003, pp 1293–1303