Increasingly Correct Message Passing Algorithms for Heat Source Detection in Sensor Networks[1]

K. Plarre and P. R. Kumar
Electrical and Computer Engineering Department
and Coordinated Science Laboratory
1308 W. Main St., Urbana, IL, 61801
Email: {plarre, prkumar}control.csl.uiuc.edu

T. Seidman
Department of Mathematics and Statistics
University of Maryland, Baltimore County
Baltimore, MD, 21250
Email: seidman@math.umbc.edu

***Abstract*— Solving complex source inference and estimation problems in the distributed environment of sensor networks is a difficult task. Data is distributed, as is computational power, and energy is limited.**

We consider the problem of detecting and locating a heat source appearing in a region covered by a sensor network. This task requires complex computations that must be shared by all sensors.

One significant difficulty we overcome is the problem of local minima. By using a two step procedure, where in the first step nodes estimate their distances to the source, and in the second step localize it, we avoid the problem of having very erroneous estimates of location that local minima can produce. Further, to organize the computations involved, we draw on ideas from graphical models. We develop an algorithm that has the a property which we call "increasing correctness," that at any time the algorithm can be stopped and it nevertheless provides the correct answer for the problem defined by the information that has been fed into the algorithm up to that time.

I. Introduction

There are a number of issues that make the design of applications for sensors networks a difficult task.

Power limitation is a central issue in sensor networks with nodes powered by batteries. To reduce power consumption, nodes are provided with a "sleep" mode in which they consume only a fraction of the power when awake. Management of sleeping and awake modes in sensor networks is an important problem.

Power limitations also make communications in a sensor network an expensive asset. Any application for sensor networks has to be designed to operate with minimal communications. The joint design of computation and communications constitutes another challenging problem in sensor networks; see [1].

Computational power in sensor networks is also limited, making complex computations prohibitively slow. Moreover information in the network is distributed, and so computations cannot be centralized on one sensor, even if its computational power would allow it. Even algorithms that could be trivially programmed on a PC, prove to be difficult to implement in sensor networks. The organization of such distributed algorithms is a challenging task, e.g., [2]–[4].

In this paper we explore these issues through a very specific problem, namely the detection and location of a heat source. A number of motes with omnidirectional temperature sensors are located in a certain region. At some unknown time a heat source appears in the region. The tasks of the network are to detect the onset of the heat source, determine its time of appearance, its amplitude and location, and then issue an alarm. We will see that this problem involves a large number of the issues mentioned above.

Our distributed algorithm draws on ideas from graphical models and message passing [5]–[8]. We provide an algorithm that has a property which we call "increasing correctness." This property allows us to stop the algorithm at any given time and still obtain a result that is correct for the problem defined by all the information included in the computation up to that time.

Our algorithm also avoids the problem of getting stuck in local minima since they can give totally erroneous estimates of the source ocation. This is done by using a two step procedure where in the first step each node estimates its distance to the source, and in the second step the source is localized by message passing.

Our algorithm also features distributed detection to

[1]Please address all correspondence to the first author.

determine the onset of a source at a random time.

Graphical models and message passing for computation in sensor and ad-hoc networks have been considered before, [3], [4], although in different contexts and with different approaches than the one we take here.

II. Contributions

We consider the problem of detecting a heat source appearing at some random time and position. For this we devise a multistep algorithm that aims to reduce the probability of false alarm, and to estimate the location of the source.

The organization of such an algorithm in a distributed environment is a difficult task. To aid us in the design of the algorithm we use ideas from message passing in graphical models. These ideas are not limited to the application we consider, but can be used in the design of other algorithms.

We introduce the notion of an “increasingly correct” algorithm. We show that message passing for marginalization on trees satisfies this property.

We then discuss an algorithm for the detection and location of a heat source in a sensor network, and construct an increasingly correct algorithm that can be implemented in a sensor network, and one which avoids local minima that can lead to very wrong estimates of source location.

Graphical models and message passing have been considered before as candidates for the implementation of algorithms in sensor and ad-hoc networks, for example, in [3], [4]. The approach we take here is different from those.

III. Problem formulation

We consider a collection of sensors $S = \{s_1, s_2, \ldots, \}$ distributed on the plane. Let $x_i \in I\!R^2$ denote the location of sensor s_i. We assume that each sensor knows its own location.

We assume that sensors can communicate via a multi-hop protocol, i.e., the communication graph is connected. See Figure 1. The goal of the network is to detect and locate a heat source appearing at some location on the plane. Before the source appears, the temperature at any point of the plane is some u_0. At some random time $\bar{\tau}$, a heat source appears at some position $\bar{x}$. The presence of the source should be detected as quickly as possible, with minimal false alarms.

There are high costs associated with a delay or failure in detecting the source, as well as with false alarms.

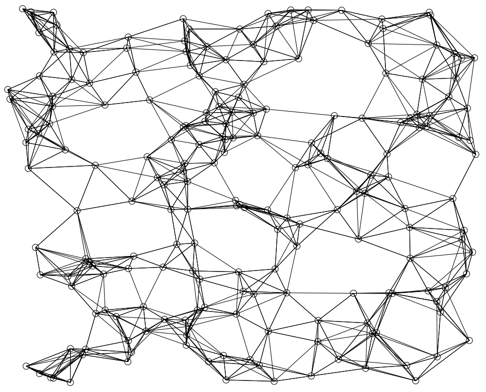

Fig. 1. Connectivity graph of a network of sensors.

In addition to this, sensors run on batteries, so that computation and communication are costly in terms of network lifetime.

To deal with these issues we propose a three phase algorithm:

1) The first phase corresponds to a setup phase in which sensors are deployed, set up the communication network, and perform some initial computations.
2) During the second phase most sensors are in a “sleep” state, a low power, inactive state. The rest of the sensors, which we call “sentinels” are awake and periodically sense the temperature. When a sentinel sensor detects the presence of a heat source, it issues an alert signal, waking up other sensors in its vicinity.
3) Third phase: After a source has been detected, the sensors that have been woken up cooperate to minimize the possibility of false alarm. If the presence of the source is affirmed, an alarm is issued. Otherwise, non-sentinel sensors go back to sleep, and the operation returns to the first phase.

The set of sentinel sensors is periodically changed, so that no sensor is continually awake.

We assume the following approximate heat equation model for the behavior of the temperature:

$$u(x,t) = \begin{cases} u_0 & \text{if } t \le \bar{\tau}\,, \\ \bar{a}\mu(||x-\bar{x}||, t-\bar{\tau}) + u_0 & \text{if } t > \bar{\tau}\,, \end{cases}$$

where

$$\mu(\Delta x, \Delta t) := \int_0^{\Delta t} \frac{1}{4\kappa\pi\theta} \mathrm{e}^{-\frac{\Delta x^2}{4\kappa\theta}}\, d\theta\,.$$

Here κ is the diffusion coefficient, which we assume known.

If node s_i senses the temperature at time t_j, its measurement is given by

$$u_i(t_j) = u(x_i, t_j) + w_i(t_j)\,,$$

where $w_i(t)$, $t \geq 0$ is a white Gaussian noise process, independent of the noise processes at other nodes.

IV. Graphical models and message passing

Graphical models, e.g., [5], [6], [8], represent the structure of marginalization problems through a graph, and allow us to construct efficient algorithms for computing marginals. A marginalization problem [5] refers to a problem where $\{\alpha_j(x_{S_j}) | 1 \leq j \leq m\}$ is input data and the goal is to compute the "marginals,"

$$\sigma(x_S) = \sum_{x_{S^c}} \prod_{j=1}^{m} \alpha_j(x_{S_j})\,, \tag{1}$$

where S, and $\{S_j\}_{j=1}^m$ denote index sets, and we have used the usual notational convention that variables indexed by a set denote a set of variables, for example, $x_{\{1,2,3\}} = \{x_1, x_2, x_3\}$. Note that S need not be one of the sets S_j, but usually one assumes that it is. In our example, $x_1, x_2, \ldots x_n$ are discrete variables, and the sum is taken over all combinations of values of all variables in the complement of S.

The graphical structure of a marginalization problem like (1) can be represented by a graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$, and $(i, j) \in E$, if and only if $S_i \cap S_j \neq \phi$. For example, if we let $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, $S_3 = \{3, 4\}$, and $S_4 = \{3, 5\}$, we obtain the graphical representation shown in Figure 2(a). Message passing algorithms [5] solve marginalization problems efficiently by constructing algorithms that work on the corresponding graph. These algorithms solve a marginalization problem by letting nodes in the graph send "messages" to their neighbors, and computing "beliefs," which are estimates of the marginals.

Let N_i denote the set of neighbors of i in G, and $N_{i,j} := N_i \setminus \{j\}$. Denote $S_{i,j} := S_i \cap S_j$. The message that node i sends to node j is a function of $x_{S_{i,j}}$,

$$m_{i,j}(x_{S_{i,j}}) = \sum_{x_{S_i \setminus S_j}} \alpha_i(x_{S_i}) \prod_{k \in N_{i,j}} m_{k,i}(x_{S_{k,i}})\,. \tag{2}$$

the product over the empty set being unity.

The belief at node i, i.e., the estimate of the marginal $\sigma(x_{S_i})$ is

$$b(x_{S_i}) = \alpha(x_{S_i}) \prod_{k \in N_i} m_{k,i}(x_{S_{k,i}})\,. \tag{3}$$

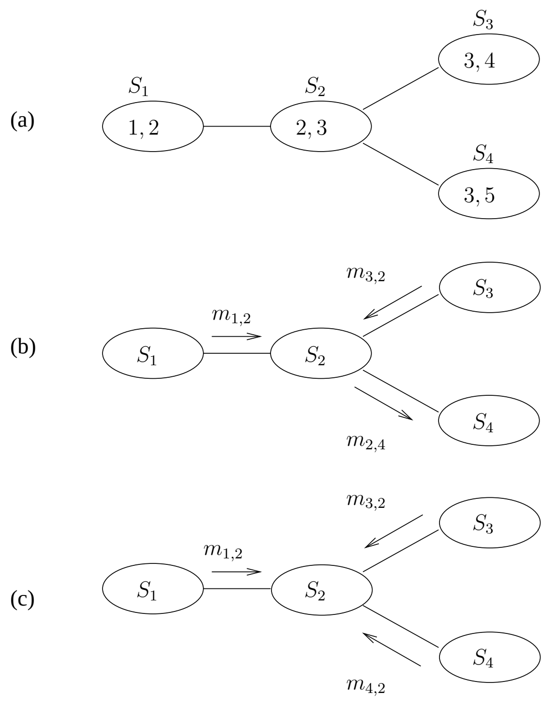

Fig. 2. Graphical representation of marginalization problem.

This message passing process is shown in Figure 2(b), and the computation of the belief is shown in Figure 2(c). Note that message passing allows us to construct estimates $b_{x_{S_i}}$, for the marginal at each node $\sigma_{x_{S_i}}$ at the same time.

It is well known that when G is a tree, message passing will converge after a finite number of steps, and deliver the correct marginals; see for example [5], [7]. When G has cycles (i.e., "is a loopy graph"), message passing might not converge, and, even if it converges, produces only approximations to the marginals; see [5], [7]).

We will construct a message passing algorithm to estimate the location, amplitude, and onset time of the source. We will not use message passing for marginalization directly, but construct an algorithm that follows similar ideas. We will design our algorithm to operate on a spanning subtree of the communication graph; hence we do not need to worry about cycles.

One of the properties of message passing in trees, which we will attempt to replicate, is that, at each time, the belief at node i is the correct solution of a subproblem of (1), which includes all the nodes that i "has heard of" up to that time. This set can be formally defined in the following way: Let $H_i \subset V$ be the set of nodes i has heard of at time t. Then, if i sends a message to j, H_j is updated according to the rule

$$H_j \leftarrow H_j \cup H_i \cup \{i\}\,.$$

Initially $H_i = \{i\}$, for $i = 1, 2, \ldots, n$. We call this property "increasing correctness," and an algorithm with this property "increasingly correct." Note also that this algorithm is asynchronous. Hence little or no synchronization is necessary.

Increasingly correct algorithms are of interest to us because they allow us to control the time of operation of the algorithm in a large network. An increasingly correct algorithm can be stopped at any given time, and node i would have the correct answer to a problem that includes all the nodes in H_i.

V. Increasingly Correct Algorithms

In this section we generalize the notion of increasing correctness of increasing correctness to trees. We begin with some definitions.

Definition 1: Let $G = (V, E)$ be a tree, with $V = \{1, 2, \ldots, n\}$. To each i we associate a vector of parameters a_i. Let $a := \{a_1, a_2, \ldots, a_n\}$. For each i, let $\sigma_i(G, a)$ be a function of G and a, and let $\sigma := \{\sigma_1, \sigma_2, ..., \sigma_n\}$. We let $\mathcal{P}\left[G, a, \sigma\right]$ denote the problem of computing $\sigma_i(G, a)$, for each $i = 1, 2, \ldots, n$.

Definition 2: Given G, a, and σ, a message passing algorithm for $\mathcal{P}\left[G, a, \sigma\right]$ is an algorithm of the form:

$$\begin{aligned}
m_{i,j} &\leftarrow f_{i,j}\left(\{m_{k,i}\}_{k\in N_{i,j}}, a_i\right)\,, \\
b_i &\leftarrow f_i\left(\{m_{k,i}\}_{k\in N_i}, a_i\right)\,, \\
H_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} H_{k,i} \cup \{i\}\,, \\
H_i &\leftarrow \bigcup_{k\in N_i} H_{k,i} \cup \{i\}\,,
\end{aligned}$$

such that $b_i = \sigma_i(G, a)$, when $H_i = \{1, 2, \ldots, n\}$. As above N_i denotes the set of neighbors of i in G, $N_{i,j} := N_i \setminus \{j\}$, while $f_{i,j}$ and f_i are given functionals.

Note that it is not necessary to include $H_{i,j}$ and H_i in the definition of a message passing algorithm, since many computations can be performed without them. We include them here because these sets carry information about the identity of the nodes that have already participated in the computations, which is useful information, especially in sensor networks, as we will see.

Let $G = (V, E)$ be a tree, with $V = \{1, 2, \ldots, n\}$. Let a and σ be given. For any $S \subset V$, let G_S denote the subgraph of G induced by S. Assume that for every $S \subset V$, we can define a restriction of σ_i to (G_S, a_S), which we denote by $\sigma_i(G_S, a_S)$. We can then define a family of problems $\mathcal{P}_S\left[G_S, a_S, \sigma_S\right]$.

Definition 3: A message passing algorithm for a problem $\mathcal{P}\left[G, a, \sigma\right]$ is called increasingly correct if its restriction to G_S is a message passing algorithm for $\mathcal{P}_S\left[G_S, a_S, \sigma_S\right]$ for each $S \subset G$.

We state the following lemma here and give the proof in Appendix A.

Lemma 1: Message passing for marginalization on trees is increasingly correct.

A. Example: Computation of the maximum

As a small example, consider the computation of the maximum of a set of numbers. Let G be as before, and $a = \{a_1, a_2, \ldots, a_n\} \in I\!R$. We wish to compute the maximum of the a_i. We then have $\sigma_i(G, a) = \max\{a_i | i = 1, 2, \ldots, n\}$. For $S \subset V$, the restriction of σ_i to (G_S, a_S) is $\sigma_i(G_S, a_S) = \max\{a_i | i \in S\}$ We can accomplish this with the following message passing algorithm

$$\begin{aligned}
m_{i,j} &\leftarrow \max\{\max\{m_{k,i} | k \in N_{i,j}\}, a_i\} \\
b_i &\leftarrow \max\{\max\{m_{k,i} | k \in N_i\}, a_i\} \\
H_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} H_{k,i} \cup \{i\}\,, \\
H_i &\leftarrow \bigcup_{k\in N_i} H_{k,i} \cup \{i\}\,.
\end{aligned}$$

It is easy to see that this algorithm will indeed compute the maximum of the a_i, and also that it is increasingly correct.

VI. Algorithm

We present here the algorithm to estimate the location and amplitude of the source, and the time it appears. It is a simple algorithm, but implementing it in a sensor network presents a large number of difficulties.

As was already mentioned, the algorithm is divided into three phases. We now describe in detail the tasks that the sensor network has to perform in each phase, and their implementation.

A. First phase: Setup

During the setup phase, all sensors are awake, and cooperate to estimate the background temperature and a model for the noise. Sensors also precompute and store some of the computations they will require later.

Let us first consider the estimation of u_0 and the measurement noise variance. For this the sensors first find a spanning subtree of the communication graph, and collect temperature measurements. Let $U_i := \{u_{i,j}, u_{i,2}, \ldots, u_{i,m_i}\}$ be the set of temperature measurements taken by sensor s_i.

To compute the average temperature and noise variance we use a message passing algorithm:

$$\begin{aligned}
m^{\mu}_{k,i} &\leftarrow \sum_{k\in N_{i,j}} m^{\mu}_{k,i} + \sum_{j=1}^{m_i} u_{i,j} \\
b^{\mu}_{i} &\leftarrow \sum_{k\in N_{i}} m^{\mu}_{k,i} + \sum_{j=1}^{m_i} u_{i,j} \\
m^{\sigma}_{k,i} &\leftarrow \sum_{k\in N_{i,j}} m^{\sigma}_{k,i} + \sum_{j=1}^{m_i} u^2_{i,j}\,, \\
b^{\sigma}_{i} &\leftarrow \sum_{k\in N_{i}} m^{\sigma}_{k,i} + \sum_{j=1}^{m_i} u^2_{i,j}\,, \\
H_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} H_{k,i} \cup \{i\}\,, \\
H_{i} &\leftarrow \bigcup_{k\in N_{i}} H_{k,i} \cup \{i\}\,.
\end{aligned}$$

Then, s_i can estimate the background temperature and noise variance as

$$\begin{aligned}
u_0 &= \tfrac{b^{\mu}_i}{|H_i|} \\
\sigma^2 &= \tfrac{b^{\sigma}_i}{|H_i|} - u_0^2\,.
\end{aligned}$$

During this phase, sensors also precompute some of the computations they will require during the third phase, and store them. In particular, the values of $\mu(\Delta x, \Delta t)$ are needed. These values are computationally intensive to compute. They are thus especially beneficial to precompute initially, when no other computational load is present, rather than in real-time when the computational resources are needed for other purposes.

In order to reduce the number of values to be stored, we exploit the structure of the function $\mu(\Delta x, \Delta t)$. It can be rewritten as

$$\begin{aligned}
\mu(\Delta x, \Delta t) &= \textstyle\int_0^{\Delta t} \frac{1}{4\kappa\pi\theta} \mathrm{e}^{-\frac{\Delta x^2}{4\kappa\theta}}\, d\theta \\
&= \tfrac{1}{4k\pi}\eta(\Delta t/\Delta x^2)\,,
\end{aligned}$$

where

$$\eta(z) := \int_0^z \frac{1}{\theta}\mathrm{e}^{-\frac{1}{4\kappa\theta}}\, d\theta\,.$$

This is a one dimensional function. Each sensor computes a table of values of $\eta(z_l)$, for $z_l = (l-1)h$, where h is a fixed small number. Let

$$f(\theta) := \frac{1}{\theta}\mathrm{e}^{-\frac{1}{4\kappa\theta}}\,.$$

To compute the integrals we can use any numerical integration algorithm. When a value of $\mu(\Delta x, \Delta t)$ is needed, the values in the table can be interpolated, for example by cubic splines, and the needed value computed.

B. Second phase: Sentinels

Each sentinel sensor s_i periodically senses the temperature, and uses these data to perform a statistical test to either declare the presence of a source or continue sampling.

Let H_0 be the hypothesis "no source present," and H_1 be "source present." Then under each hypothesis we have:

$$\begin{aligned}
H_0:&\quad u_i(t_{i,j}) = u_0 + w_i(t_{i,j})\,, \\
H_1:&\quad u_i(t_{i,j}) > u_0 + w_i(t_{i,j})\,.
\end{aligned}$$

where $u_i(t_{i,j})$ is the temperature measured by sensor s_i at time $t_{i,j}$.

To decide between H_0, and H_1, s_i keeps a record of its $M+1$ last measurements. Let $\{u(t_k), u_(t_{k-1}), \ldots, u_(tk-M)\}$ be these measurements. A Student t-test is used to make the decision. For this, s_i also keeps record of the following running sums

$$\begin{aligned}
A_k &= \sum_{j=0}^{M} u(t_{k-j})\,, \\
S_k &= \sum_{j=0}^{M} u(t_{k-j})^2\,.
\end{aligned}$$

These values are updated at each time according to the rules:

$$\begin{aligned}
A_{k+1} &= A_k - u(t_{k-M}) + u(t_{k+1})\,, \\
S_{k+1} &= S_k - u(t_{k-M})^2 + u(t_{k+1})^2\,.
\end{aligned}$$

With these values we can estimate the noise variance as

$$\hat{\sigma}^2 = \frac{1}{M+1}S_k - \frac{1}{M^2}A_k^2\,.$$

The statistics for the Student t-test is given by

$$t_{\text{test}} = \frac{A_k - M\mu_0}{\sqrt{S_k}} \,.$$

which has the t-distribution wirth M degrees of freedom.

Since the cost of failing to detect the source is higher than the cost of a false alarm, we reject H_0 at a low significance level. For example, for $M = 15$, with a significance level of 0.05, we reject H_0, and hence declare an alert, if $t_{\text{test}} \geq 1.735$.

C. Third phase: Location detection

The sentinel sensor that issues the warning wakes up other sensors in its vicinity. After waking up, these sensors quickly take their own temperature samples. Let $\{u_i(t_{i,1}), u_i(t_{i,2}), ..., u_i(t_{i,m_i})\}$ denote the set of measurements taken at sensor S_i.

We define the following local cost functions:

$$\begin{aligned} J_i(a,\tau,x) &:= \sum_{j=1}^{m_i} \left[a\mu(||x-x_i||, t_{i,j}-\tau) - u_i(t_{i,j})\right]^2 \\ \bar{J}_i(a,\tau,d_i) &:= \sum_{j=1}^{m_1} \left[a\mu(d_i, t_{i,j}-\tau) - u_i(t_{i,j})\right]^2 \,. \end{aligned}$$

Given any set of sensors $S' \subset S$, we define the following cost functions:

$$\begin{aligned} J_{S'}(a,\tau,x) &:= \sum_{s_i \in S'} J_i(a,\tau,x)\,, \\ \bar{J}_{S'}(a,\tau,d_{S'}) &:= \sum_{s_i \in S'} \bar{J}_i(a,\tau,d_i)\,. \end{aligned}$$

The estimates of the amplitude, time, and location of the source, will be obtained from:

$$(\hat{a},\hat{\tau},\hat{x}) = \underset{a,\tau,x}{\text{argmin}} J_{S'}(a,\tau,x)\,, \tag{4}$$

where S' is the set of sensors that have time to participate in above the computations.

Before discussing the implementation of this phase of the algorithm, we briefly mention the difficulties we expect to encounter.

Besides the already mentioned difficulties, such as power, time constraints, and false alarms, we see that the algorithm presents the following challenges:

1) Computational complexity: The algorithm we propose requires computationally intensive operations. These operations are performed in-network, on the relatively slow processors of the sensors.

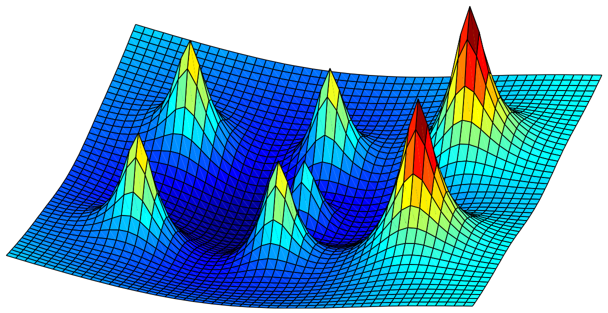

Fig. 3. Typical landscape of cost function to be minimized to find location of the source. Note the presence of local minima, and that the locations of the local minima are scattered all over the domain.

2) Organization: It is a challenging task to organize the computations in the sensor network. It is here that the message passing approach is helpful.
3) Nonconvex optimization: We see that (4) is a nonconvex optimization problem. We are forced to find a global minimum of this problem because local minima will indicate false positions of the source. Even if $\bar{a}$ and $\bar{\tau}$ are known exactly, the optimization problem (4) is nonconvex in x. A typical landscape for this cost function is shown in Figure 3. Note that the locations of the local minima are scattered all over the domain and so a localization procedure which finds only a local minimum can be completely useless.

The general approaches we will follow to solve these problems are:

1) Power constraints: Besides the use of sentinel sensors, we will reduce the power consumption by adequate coding of data transmitted in the network. This will reduce the lengths of the messages to be transmitted.
2) Time constraints: By reducing the sizes of the messages transmitted, we also reduce the time required for transmission and the probability of collisions. The fact that a large part of the necessary computations have been precomputed and stored during the setup phase, and can be retrieved instead of recomputed, largely reduces the reaction time of the network to the presence of a source.
3) Complexity: Computational complexity is greatly reduced by the use of precomputation. We complement this with the use of parallel implementations

of the most computationally demanding portions of the algorithm, to reduce the workload of individual sensors. Note that the use of parallelism also helps in reducing the reaction time of the network.

4) Organization: We make use of the message passing approach to facilitate the organization of the algorithm. We will design the algorithm in such a way that it is increasingly correct. This will also help us in managing the time constraint, because the algorithm can be stopped at any time, and yet it is guaranteed to produce the correct answer given all the information that has been fed into the computation up to that point in time.
5) Local minima: To avoid local minima in the optimization problems we will adopt a two step procedure. First, each sensor estimates its distance to the source and then, in the second step, the nodes collaborate to find the location of the source.

Now we address the above issues (2-5) in detail.

To define a message passing algorithm to solve (4), we need to define the messages and beliefs. Assume first that, at the beginning of the third phase, all the sensors posses a "good enough" estimate of $\bar{a}$, $\bar{\tau}$, and $\bar{x}$. We discuss how to obtain this initial estimate in Section 6.

For each i, we let

$$\begin{aligned} U_i &:= \{u_i(t_{i,1}), u_i(t_{i,2}), \ldots, u_i(t_{i,m_i})\}\,, \\ T_i &:= \{t_{i,1}, t_{i,2}, \ldots, t_{i,m_i}\}\,. \end{aligned}$$

The message that s_i sends to s_j has the form:

$$m_{i,j} = (H_{i,j}, U_{i,j}, T_{i,j}, X_{i,j}, P_{i,j})\ ,$$

where $P_{i,j} = (a_{i,j}, \tau_{i,j}, x_{i,j})$. The update rules for these messages are:

$$\begin{aligned} H_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} H_{k,i} \cup \{i\}\,, \\ U_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} U_{k,i} \cup U_i\,, \\ T_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} T_{k,i} \cup T_i\,, \\ X_{i,j} &\leftarrow \bigcup_{k\in N_{i,j}} X_{k,i} \cup \{x_i\}\,, \end{aligned}$$

The computation of $P_{i,j}$ is more involved. First note that $H_{k,i}$ contains the indices of all sensors s_k has heard from in the subtree $G_{k,i}$; see Figure 4. From the way we will

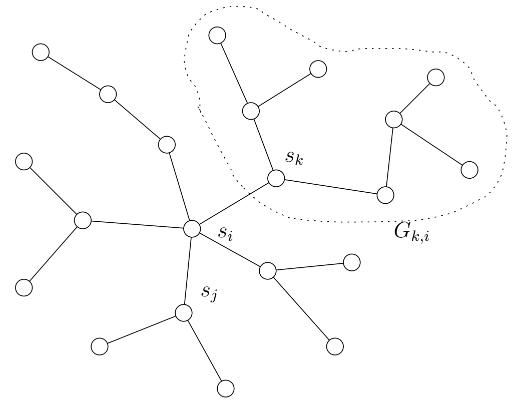

Fig. 4. The message that s_k sends to s_i contains information coming only from the subtree $G_{k,i}$.

construct the algorithm, we will also have

$$(a_{k,i}, \tau_{k,i}, x_{k,i}) = \underset{a,\tau,x}{\mathrm{argmin}} J_{H_{k,i}}(a,\tau,x)\,.$$

To compute $P_{i,j}$, sensor i then solves

$$(a_{i,j}, \tau_{i,j}, x_{i,j}) = \underset{a,\tau,x}{\mathrm{argmin}} J_{H_{i,j}}(a,\tau,x)\,. \qquad (5)$$

Note that the information that s_i receives from its neighbors allows it to do these computations.

To reduce the time to solve (5), sensor s_i uses the received estimated parameters $P_{k,i}$, $k \in N_{i,j}$ as initial conditions for a descent algorithm, for example Newton's method. Note that s_i starts Newton iterations at $|N_{i,j}|$ different points. In the case that these iterations do not converge to the same point, s_i chooses the minimum of the resulting costs.

The belief at node i is given by

$$b_i = (H_i, U_i, T_i, X_i, P_i)\ ,$$

where $P_i = (a_i, \tau_i, x_i)$. The update of these beliefs is done according to the following rules:

$$\begin{aligned} H_i &\leftarrow \bigcup_{k\in N_i} H_{k,i} \cup \{i\}\,, \\ U_i &\leftarrow \bigcup_{k\in N_i} U_{k,i} \cup U_i\,, \\ T_i &\leftarrow \bigcup_{k\in N_i} T_{k,i} \cup T_i\,, \\ X_i &\leftarrow \bigcup_{k\in N_i} X_{k,i} \cup \{x_i\}\,, \end{aligned}$$

P_i is obtained from

$$(a_i, \tau_i, x_i) = \underset{a,\tau,x}{\mathrm{argmin}} J_{H_i}(a,\tau,x)\,.$$

Again, s_i uses the received $P_{k,i}$ as initial conditions for a descent method, and chooses the minimum of the obtained results.

Figure 5 shows how the information from sensors in a vicinity of s_i is incorporated progressively into the belief at s_i. It illustrates the property of increasing correctness.

We have not yet specified how to obtain the initial estimate of $(\bar{a}, \bar{\tau}, \bar{x})$. We do so now.

1) Obtaining the initial estimate: Let s_0 be the first sensor that detects the source. Let N_0 be the set of neighbors of s_0. After detecting the source, s_0 wakes up its neighbors. Together they form a crude, initial estimate of the source location.

Let $\bar{N}_0 := N_0 \cup \{0\}$. The sensors $s_{\bar{N}_0}$ obtain the initial estimate by solving

$$(a^0, \tau^0, x^0) = \underset{a,\tau,x}{\operatorname{argmin}} J_{\bar{N}_0}(a, \tau, x)\,. \tag{6}$$

This is however a nonconvex problem in four variables, to overcome it the sensors first solve the related problem

$$(a^0, \tau^0, d^0_{\bar{N}_0}) = \underset{a,\tau,d_{\bar{N}_0}}{\operatorname{argmin}} \bar{J}_{\bar{N}_0}(a, \tau, d_{\bar{N}_0})\,. \tag{7}$$

Note that (7) involves more variables than (6). It might seem that this is then a harder problem. But notice that we can rewrite (7) as

$$(a^0, \tau^0, d^0_{\bar{N}_0}) = \underset{a,\tau}{\operatorname{argmin}} \sum_{i\in\bar{N}_0} \underset{d_i}{\operatorname{argmin}} \bar{J}_i(a, \tau, d_i)\,.$$

We can thus divide the optimization problem into an "outer loop" in a and τ, and an "inner loop" in each d_i, $i \in \bar{N}_0$. These last series of optimizations can be performed in parallel by each s_i in $\bar{N}_0$. Specifically, each sensor s_i, $i \in \bar{N}_0$, computes a table

$$d_{i,l,m} = \underset{d_i}{\operatorname{argmin}} \bar{J}_i(a_l, \tau_m, d_i)\,,$$

for every (a_l, τ_m) on a grid:

$$\begin{array}{ccccccccc} 0 & = & a_0 & < & a_1 & < & \ldots & < & a_{n_a} & = & a_{\max}\,, \\ 0 & = & \tau_0 & < & \tau_1 & < & \ldots & < & \tau_{n_\tau} & = & \tau_{\max}\,, \end{array}$$

where $a_{\max}$, and $\tau_{\max}$ are the maximum possible values of $\bar{a}$, and $\bar{\tau}$, respectively.

Each sensor sends its corresponding table to s_0. For each (l, m), the following triangulation problem is solved:

$$x_{l,m} = \underset{x}{\operatorname{argmin}} \sum_{i\in\bar{N}_0} \left[||x - x_i||^2 - d_{i,l,m}\right]^2\,.$$

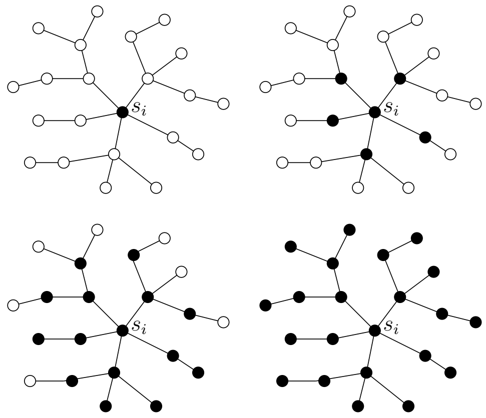

Fig. 5. Illustration of how the information from the vicinity of s_i is progressively included into the belief at s_i. Black disks represent sensors in H_i as time passes.

To reduce the workload of each single sensor, s_0 reassigns these problems to its neighbors, which solve them in parallel.

In the last step of this process, each $(a_l, \tau_m, x_{l,m})$ is used as an initial condition for a descent algorithm for solving (6). Again, this step can be performed in parallel. The minimizing value found in this process is then (a^0, τ^0, x^0).

2) Coding: We notice that the computation of the initial parameter estimate involves the communication of a large amount of data. Although this communication is local, i.e., it does not involve multi-hop communication, we find it convenient to reduce the length of the messages sent, by coding them in an appropriate way.

For doing so, we view $d_{i,l,m}$ as a two dimensional function of l and m. We can thus construct a linear regression

$$d_{i,l,m} = \alpha l + \beta m + \gamma + \epsilon_{l,m}\,.$$

We can therefore encode the values of α, β, γ, and $\epsilon_{l,m}$. Since, in general $|\epsilon(l, m)| \ll |d_{i,l,m}|$, this encoding reduces the number of required bits.

VII. Simulations

In this section, we present some simulation results. where, given the space constraints we present simulations of the triangulation algorithm only.

We let 10 sensors located in the unit square, determine the location of a heat source. The source appears at time $\bar{\tau} = 0.6$. Sensors have available temperature measurements at times $\{1, 1.2, 1.4, 1.6, 1.8, 2\}$. The amplitude of the source is $\bar{a} = 1$. The noise variance is $\sigma^2 = 0.001$. We for these simulations we use a value $\kappa = 1$. The

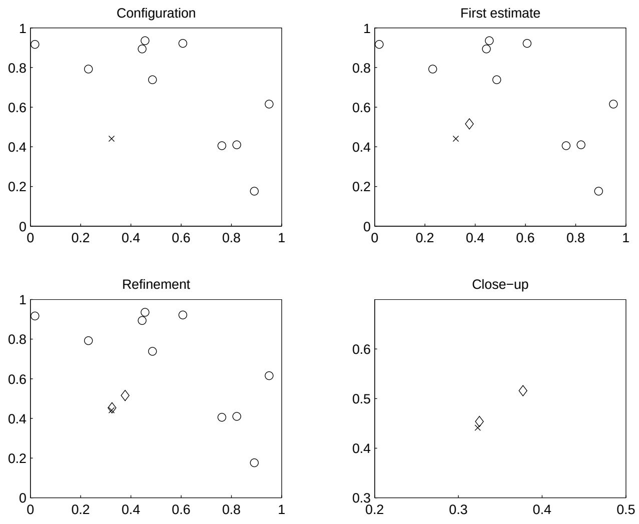

Fig. 6. Two step location process: First a rough estimate of the source location is found, then this estimate is refined.

value of u_0 is irrelevant to the results, therefore we set $u_0 = 0$.

In the upper left plot in Figure 6, we show the configuration of sensors. Sensors are represented by circles, while the true heat source location is shown as a cross.

The first approximation of the source location, given by the triangulation step is shown in the top right plot in Figure 6. The diamond shows the estimated location.

The previous location estimate is then used as initial condition to minimize the global cost function (4). The refined estimated location is shown in Figure 6 (bottom left).

Figure 6 (bottom right) shows a close-up of the region surrounding the source. It can be seen that the our two step procedure converges to the source location.

VIII. Conclusions

The notion of incresing correctness we introduced proves to be useful in the design of algorithm for computations in distributed environments, especially in large networks and time constrained situations, and with unreliable communication. In fact, increasingly correct algorithms always give the correct answer to the problem defined on the data they have been fed with. This allows each node to gracefully incorporate new arriving information into the computations, in a graceful way.

We have used these ideas to construct a multistage algorithm for the detection and localization of a heat source in a sensor network.

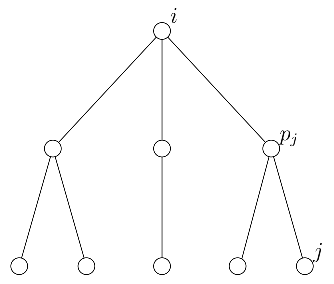

Fig. 7. Subtree of G, induced by H_i, and rooted at i.

Appendix

A. Proof of lemmma 1

Consider the marginalization problem defined by (1), and the message passing iteration defined by (2) and (3). Suppose that at some given time the algorithm is stopped. Call this time t. Let b_i^t be the last belief at node i computed before the algorithm stopped. Likewise, let H_i^t be the set of nodes i “heard from,” before the algorithm was stopped.

We can show that b_i^t is the correct marginal for the marginalization problem:

$$\sigma_i^t(x_{S_i}) = \sum_{x_{H_i \setminus S_i}} \prod_{j \in H_i} \alpha_j(x_{S_j}) \, .$$

Let G_{H_i} be the subgraph of G induced by the nodes in H_i. It is easy to see that G_{H_i} is connected: If $j \in H_i$, then there exists a path from j to i in G, otherwise i would “never have heard from j.” Then G_{H_i} is a subtree of G containing i. Let T_i be the corresponding tree, rooted at i, and for each $j \in H_i$, $j \neq i$, let p_j denote the parent if j in T_i. For each $j \in H_i$, let C_j be the set of children of j in T_i. See Figure 7.

Let M^T be the set of all messages sent during message passing up to time t. We note that not all messages in M^t contributed to the computed value of b_i^t. We now construct the minimal set of messages that determined the value of b_i^t. We call this set M_i^t

First note that no message from a node $j \in H_i$ to a node in C_j can be in M_i^t. This is so because a message from j to k never uses the information sent to j by k, and T_i is cycle free. So, only messages from a node j to p_j can be in M_i^t.

We also note that, for each $j \in H_i$, only one message from j to p_j can be in M_i^t. We call this message m_j^t. We now construct M_i^t recursively:

1) If $j \in C_i$, then m_j^t is the last message that j sent to i before i computed b_i^t.

2) For every $j \in H_i$, $j \notin C_i$, m_j^t is the last message that j sent to p_j before p_j sent $m_{p_j}^t$.

Given M^t, M_i^t is unique by construction.

Let j be a leaf node of T_i. Then m_j^t contains only information that is local to j. This is so because:

- m_j^t cannot contain information from any of its ancestors, because T_i is cycle free.
- m_j^t cannot carry information from any $k \in N_{j,p_j}$. Otherwise $k \in H_i$, which is a contradiction.

In fact,

$$m_j^t(x_{S_{j,p_j}}) = \sum_{x_{S_j \setminus S_{p_j}}} \alpha_j(x_{S_j}) \, .$$

For any $j \in H_i$, j not a leaf of T_i, we then have

$$m_{j,p_j}^t(x_{S_{j,p_j}}) = \sum_{x_{S_j \setminus S_{p_j}}} \alpha_j(x_{S_j}) \prod_{k \in C_j} m_{k,j}(x_{S_{k,j}}) \, .$$

We also have:

$$b_i^t(x_{S_i}) = \alpha_i(x_{S_i}) \prod_{k \in C_i} m_{k,i}(x_{S_{k,i}}) \, .$$

These are exactly the messages and belief that a synchronous message passing (started at the leaves, then the parents of the leaves, etc.) produced. We know that, since T_i is a tree, this set of messages produces the correct marginal b_i^t. So, message passing for marginalization in trees is increasingly correct.

References

[1] A. Giridhar and P. R. Kumar, “Data fusion over sensor networks: Computing and communicating functions of measurements,” *Submitted to Journal on Selected Areas in Communications*, December 2003.

[2] D. Scherber and H. Papadopoulos, “Locally constructed algorithms for distributed computations in ad-hoc networks,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, 2004, pp. 11–19.

[3] A. Ihler, J. Fisher, R. Moses, and A. Willsky, “Nonparametric belief propagation for self-calibration in sensor networks,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, 2004, pp. 225–233.

[4] R. Biswas, L. Guibas, and S. Thrun, “A probabilistic approach to inference with limited information in sensor networks,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, 2004, pp. 269–276.

[5] M. Jordan and C. Bishop, *An introduction to graphical models*. Preprint, October 2001.

[6] B. Frey, *Graphical models for machine learning and digital communication*. Cambridge, MA: MIT Press, 1998.

[7] S. M. Aji and J. McEliece, “The generalized distributive law,” *IEEE Transactions on Information Theory*, vol. 46, pp. 325–343, 1999.

[8] F. R. Kschiang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.