Networking Sensors Using Belief Propagation

Sujay Sanghavi Dmitry Malioutov Alan Willsky

***Abstract*—This paper investigates the performance of belief propagation (BP) as a distributed solution to two combinatorial resource allocation problems arising in sensor networks: network formation and fusion center location. We model these problems by max-weight b-matching and uncapacitated facility location, respectively. Each of these is a classical optimization problem. For both problems, we *(a)* show how BP can be simplified for implementation in distributed environments where transmissions are broadcast and can interfere, *(b)* derive a principled interpretation of estimates before convergence, and *(c)* compare the performance of BP to that of linear programming.**

I. Introduction

There has been much recent interest in the use of (the max-product form of) belief propagation (BP) as a distributed solver for a variety of optimization problems defined on graphs [5], [6]. While much of the interest in BP stems from its distributed and lightweight nature, in general, BP remains a heuristic: without additional assumptions on the problem it is difficult to have a-priori guarantees on the quality of the output. However, for some combinatorial optimization problems on graphs, including the ones considered in this paper, recent results [14], [3], [5] point to a connection between the output of BP and the LP relaxation of the original problem.

Applying BP in sensor network settings poses some unique challenges. Typically, BP updates are executed until convergence, i.e. until the numerical values of the messages do not change much from iteration to iteration. Most of the analytical results on BP performance pertain to the final, converged output. In sensor networks, it may not be possible to detect convergence globally. It may also not be possible to wait until convergence; recovering good (but not necessarily optimal) solutions from non-converged estimates is also important. Finally, most sensor network communications occur over a wireless medium, with every transmission broadcast to all neighbors of a node. Broadcasting causes interference between the messages that BP needs to pass between nodes.

In this paper we investigate the effectiveness of BP in solving two problems that arise in sensor network operation:

(a) Network formation,
(b) Locating fusion centers.

Each problem is well-modeled by a classical combinatorial optimization problem. We provide simplified, asynchronous implementations of BP, that allow distributed implementation in broadcast environments. We also show how to interpret intermediate messages, which have not converged, in a consistent way. Finally, we also compare BP to smoothed dual coordinate descent (since both share the same *per-update* complexity). Empirically, BP is seen to perform much better, in the sense that it converges to the correct answer in much *fewer updates*.

Email: sanghavi@purdue.edu, dmm@microsoft.com, willsky@mit.edu.
This paper appears in the Allerton Conference, 2008

II. Distributed network formation

In many distributed sensor network applications, in particular if the sensors are randomly deployed in bulk over an area, as a first step they may be required to self-organize into a network [1]. Such a network should be connected, be robust to occasional failing links, but at the same time it should be sparse (i.e. have nodes with small degrees) due to severe limitations on power available for communication. Furthermore, due to the nature of the application, the network formation itself should occur in a distributed fashion. These considerations are not fully met by any available approaches, such as spanning trees, selecting b nearest neighbors[1], or simply connecting each node to its neighbors within some distance R. An interesting set of sparse planar subgraph constructions is proposed in [10] that requires the use of directional information.

Instead, we formulate sensor network formation as a weighted b-matching problem, a well-studied problem in combinatorial optimization. Our formulation only requires the knowledge of distances between neighboring nodes, does not require planarity, and admits an efficient distributed solution via the max-product form of belief propagation. We describe the b-matching problem next, its solution via max-product in Section II-B, and our experiments in Section II-E.

A. Weighted b-matching problem

Given a graph $G = (V, E)$ with non-negative weights w_e on its edges $e \in E$, the *weighted b-matching problem* (MWbM) is to find the heaviest subset of edges such that not more than b edges share the same node. Such a subset of edges is called a b-matching – each node is 'matched' to at most b other nodes. The special case with $b = 1$ reduces to the maximum weight matching (MWM) problem, one of the most widely studied problems in combinatorial optimization.

Weighted b-matching can be naturally formulated as an integer program (IP). Let binary variable $x_e \in \{0, 1\}$ represent whether the edge $e \in E$ is part of the b-matching. Then MWbM is the solution of the following IP:

$$\begin{aligned} \mathsf{IP}: \quad & \max \sum_{e \in E} w_e x_e, \\ \mathsf{s.t.} \quad & \sum_{e \in E_i} x_e \le b \quad \text{for all } i \in V, \\ & x_e \in \{0, 1\} \quad \text{for all } e \in E. \end{aligned}$$

[1] A b-nearest neighbors solution may lead to high degrees (more than b) due to asymmetry – if i belongs to the set of b nearest neighbors of j, then the converse does not need to hold. Note that a b-matching solution disposes of this anomaly.

Here E_i is the set of edges incident to node i, and the first constraint enforces that the solution is a valid b-matching.

The b-matching problem provides a very appealing view of the sensor network formation problem. The original graph G represents the nodes that can communicate with each other (e.g. nodes within some radius R). In a random deployment scenario such a graph may have a very wide degree distribution with the undesirable presence of large hubs. We assign edge weights to be proportional to the throughput of the link, which can be argued to decay as the received power, typically as d^{-p} with distance, where $p \in [2,4]$ depending on the communications channel. We set $p = 2$ for concreteness, and let the edge weights be $w_e = d_e^{-p}$. The b-matching objective is now to maximize the total throughput (received power) among sparse subgraphs with degree at most b.

The final requirement that the solution be distributed requires some innovation. The b-matching problem originally comes from the combinatorial optimization literature, and while in general, the IP problem can be solved in polynomial time e.g. by Pulleyblank's or Edmond's algorithms, [4], these algorithms are not easily distributed.

Instead, we will view MWbM IP as a MAP estimation problem in a graphical model, and apply the max-product algorithm. The special structure of the problem allows us to make strong statements about the convergence and correctness of the MP solution, and give a connection to a linear programming relaxation of the IP, which we discuss in Section II-D.

B. Max-product solution of MWbM

We now formulate weighted b-matching on G as a MAP estimation problem by constructing a suitable probability distribution in a Markov Random Field (MRF) form over the valid matchings. This construction is naturally suggested by the form of the integer program IP. Associate a binary random variable $x_e \in \{0,1\}$ with each edge $e \in E$, and consider the following probability distribution:

$$p(x) \quad \propto \quad \prod_{i \in V} \psi_i(x_{E_i}) \prod_{e \in E} \exp(w_e x_e), \tag{1}$$

where the factors $\psi_i(x_{E_i})$ for each node $i \in V$ represent the b-matching constraints – that at most b edges incident to node i can be assigned the value "1": i.e. $\psi_i(x_{E_i}) = 1$ if $\sum_{e \in E_i} x_e \leq b$, and 0 otherwise[2]. Note that we use i to refer *both* to the nodes of G and factors of p, and e to refer both to the edges of G and variables of p. It is easy to see that, for any x, $p(x) \propto \exp(\sum_e w_e x_e)$ if the set of edges $\{e | x_e = 1\}$ constitute a b-matching in G, and $p(x) = 0$ otherwise. Thus the max-weight b-matching of G corresponds to the MAP estimate of p.

Max-product for the MRF (1) above can be simplified to the following iterative update algorithm. We provide the derivation of this scheme starting from the standard max-product message updates in the appendix. For two sets A and B the set difference is denoted by the notation $A \backslash B$. Also, $y_+ = \max(0, y)$, $\text{rank}_b(A)$ denotes the b-th rank element in the set A, and $\mathcal{N}(i)$ is the neighborhood of i. An edge estimate $\hat{x}_{ij} = 1$ corresponds to that edge being in the MWbM, "0" means it is not in the MWbM, and "?" means undecided.

[2]We will see in the next section that storing this (potentially very large) factor as a table is never necessary, and message updates involving this factor reduce to simple maximization operations.

Simplified Max-Product for Weighted b-matching

(o) Set $t = 0$ and initialize each $a^0_{i \to j} = 0$

(i) Update as follows

$$a^{t+1}_{i \to j} \quad = \quad \text{rank}_b \left\{ (w_{ik} - a^t_{k \to i})_+ \mid k \in \mathcal{N}(i) \backslash j \right\} \tag{2}$$

(ii) For each edge set $\hat{x}^t_{(i,j)} = 0, 1$ or ? if $(a^t_{i \to j} + a^t_{j \to i})$ is respectively $>, <$ or $= w_{ij}$.

We remark that for the matching problem $b = 1$, and rank-1 element is simply the maximum. The corresponding max-product updates for MWM are:

$$a^{t+1}_{i \to j} = \max_{k \in \mathcal{N}(i) \backslash j} \left(w_{ik} - a^t_{k \to i} \right)_+ . \tag{3}$$

C. Broadcast implementation

To further simplify the protocol for networks where the number of neighbors may be large, we describe an equivalent broadcast scheme, where a node broadcasts a summary message, and all the recipients can calculate their intended personal message based on the summary message and their local information.

For clarity we start with the broadcast scheme for simple matching, and the one for b-matching immediately follows. Suppose we are at node i, and we are interested in message $a^{t+1}_{i \to j}$. Instead of calculating the maximum over the remaining neighbors (i.e. $k \in \mathcal{N}(i) \backslash j$) as in (3), we calculate the maximum over *all* the neighbors:

$$\overline{a}_i{}^{t+1} = \max_{k \in \mathcal{N}(i)} \left(w_{ik} - a^t_{k \to i} \right)_+ \tag{4}$$

and let k^* denote the corresponding $\arg\max$. If $j \neq k^*$ then the message is correct, $a^{t+1}_{i \to j} = \overline{a}_i{}^{t+1}$. While the recipient node j does not know the identity of k^*, it can find out whether $j = k^*$ by comparing $w_{ij} - a^t_{j \to i}$ to $\overline{a}^{t+1}_i$. Otherwise, if $j = k^*$, we need to find the second-in-order value, i.e.

$$\underline{a}^{t+1}_i = \max_{k \in \mathcal{N}(i) \backslash k^*} \left(w_{ik} - a^t_{k \to i} \right)_+ \tag{5}$$

and then $a^{t+1}_{i \to j} = \underline{a}^{t+1}_i$. This way, instead of sending a real-valued message to *each* of its neighbors, the broadcast version only requires broadcasting a single message consisting of two real values, $\overline{a}^{t+1}_i$ and $\underline{a}^{t+1}_i$. This results in a substantial reduction in communications without sacrificing accuracy.

The extension for b-matching is immediate: instead of sending the maximum and the second-to-maximum values from $(w_{ik} - a^t_{k \to i})_+$ over $k \in \mathcal{N}(i)$, we send the b-th and $(b+1)$-th ranked values. If $w_{ij} - a^t_{j \to i}$ exceeds the rank-b value, then it must have been included in the top b elements, and to calculate the correct message we must remove it and use the rank-$(b+1)$ element. Otherwise, we use the rank-b element. We summarize this scheme for b-matching next.

Broadcast Max-Product for Weighted b-matching

(o) Set $t = 0$ and initialize each $a^0_{i\to j} = 0$

(i) Pick a node i at random, and compute the summary message consisting of two real values:

$$\begin{aligned} \overline{a}_i^{\,t+1} &= \text{rank}_b \left\{ (w_{ik} - a^t_{k\to i})_+ \mid k \in \mathcal{N}(i) \right\} \quad (6) \\ \underline{a}_i^{\,t+1} &= \text{rank}_{b+1} \left\{ (w_{ik} - a^t_{k\to i})_+ \mid k \in \mathcal{N}(i) \right\} \end{aligned}$$

For each node recipient node, $j \in \mathcal{N}(i)$:
Set $a^{t+1}_{i\to j} = \overline{a}_i^{\,t+1}$ if $w_{ij} - a^t_{j\to i} < \overline{a}_i^{\,t+1}$, and $a^{t+1}_{i\to j} = \overline{a}_i^{\,t+1}$ otherwise.

(ii) Upon convergence, output estimate $\hat{x}$: for each edge set $\hat{x}_{(i,j)} = 0, 1$ or ? if $(a_{i\to j} + a_{j\to i})$ is respectively $>, <$ or $= w_{ij}$.

D. Theoretical guarantees for MWbM Max-product via LP

The special structure of the MWbM problem has strong implications for the performance of max-product. In a previous publication [13] we studied the convergence and accuracy properties of max-product for weighted matching and b-matching, and have found striking connections with the *linear programming (LP) relaxation* of the corresponding IP. The LP relaxation of MWbM replaces the constraint $x_e \in \{0, 1\}$ with the constraint $0 \le x_e \le 1$ for each $e \in E$. For bipartite graphs it gives correct MWbM solutions, but in general non-bipartite graphs, it can have fractional optima – i.e. those that assign fractional mass to edges. In [13], [14] we have established the following connection between convergence of max-product and integrality and uniqueness of the MWbM LP relaxation:

Theorem 1: (a) If the LP has a unique optimum that is integral, then max-product will converge and the resulting solution will be exactly the max-weight b-matching.

(b) Suppose the LP is loose, or has multiple optima. Then for any edge, if there exists any optimum of LP that assigns fractional mass to that edge, then the max-product estimate for that edge will either oscillate or be ambiguous.

This theorem implies that max-product is equally powerful to the LP relaxation. This gives a strong justification for using max-product in distributed applications, where there is no straightforward way to apply linear programming[3], and it also gives insight into situations in which max-product may or may not be appropriate.

In our simulations in Section II-E we observe an even stronger connection between LP relaxation and the damped version of max-product, which also conveniently eliminates the non-convergence issues for max-product. For ordinary max-product we observe that it converges for the majority of the edges and oscillates on a small subset of edges. While one can run max-product for a fixed number of iterations, and then terminate it, a better solution is to use a 'damped' version of max-product with the message update:

$$a^{t+1}_{i\to j} = \lambda a^t_{i\to j} + (1-\lambda) \max_{k \in \mathcal{N}(i)\setminus j} \left(w_{ik} - a^t_{k\to i} \right)_+$$

and $\lambda \in [0, 1)$ is the damping factor, with $\lambda = 0$ corresponding to no damping. We empirically observe that the damped version of max-product with $\lambda > 0$ always converges and gives the same solution as the linear programming relaxation (where we map the non-informative max-product estimates $x_e =$ '?' to value 0.5).

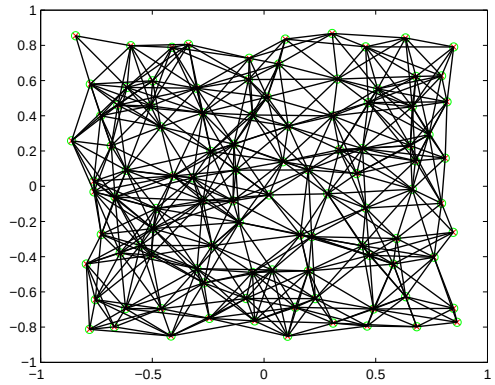

Fig. 1. Original dense graph: pairs of sensors within distance R are linked by an edge.

E. Numerical experiments

We now study the behavior of max-product for distributed sensor network formation on numerical simulations. For our first experiment we randomly disperse $N = 100$ nodes in a square region $[-1, 1] \times [-1, 1]$, and create an initial adjacency graph for nodes that are close enough to communicate, we set the threshold to be $R = 0.4$. The original dense graph is displayed in Figure 1.

Next we set edge weights to be proportional to throughput, $w_e = d_e^{-2}$, and apply max-product to find the maximum weight b-matching, with $b = 5$. We use an asynchronous version where a node is chosen at random and it broadcasts a message to all of its neighbors. We use moderate damping $\alpha = 0.25$. In Figure 2 we plot the weight of the max-weight b-matching[4] found after n node updates of max-product. We note that each update originates from a single node, so 100 updates are required just to visit the whole graph once. Clearly, max-product rapidly, within several passes through the graph, finds a matching with close-to-optimal weight.

In Figure 3 we show the corresponding max-product matching solution upon convergence. For almost all the edges max-product converges to an informative solution, i.e. either $\hat{x}_{(i,j)} = 1$ or 0, with the edge displayed or removed respectively. However, for a small cycle in the center the edges are non-informative: $\hat{x}_{(i,j)} =$?, as shown in red. Plain max-product

[3]In our experiments we study distributed implementation of the coordinate descent for the dual of the LP relaxation, and show that max-product is preferable.

[4] If in the intermediate steps the selected edges do not form a b-matching, we throw away enough violated edges until the configuration is a valid b-matching.

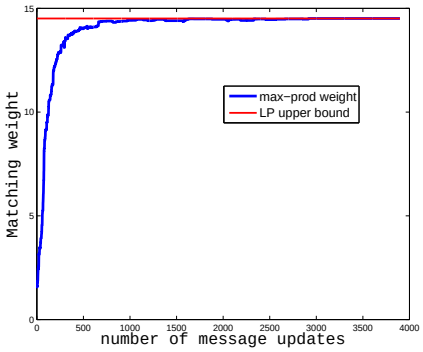

Fig. 2. Weight of the b-matching selected after n node updates of max-product.

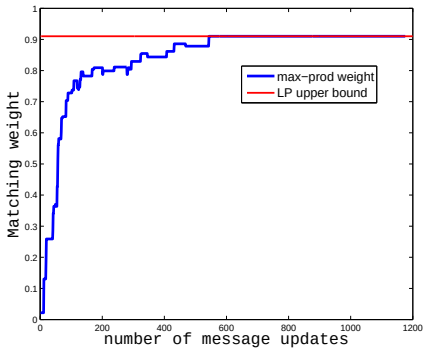

Fig. 4. Max-product matching weight vs. node updates.

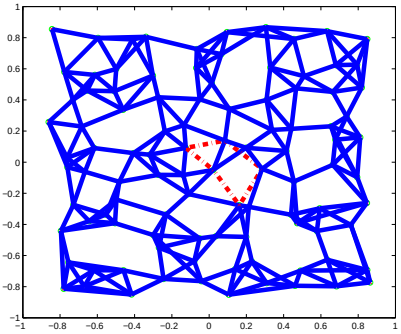

Fig. 3. Max-product b-matching solution. The original dense graph appears in Figure 1. Most of the edges converge to the correct values (edges with $x_e = 0$ are removed, and ones with $x_e = 1$ are shown in bold), while a small cycle in the center converges to an uninformative solution (shown in red).

would result in oscillations for these edges, but damped max-product converges to uninformative values for these edges, equivalent to the fractional solution of the LP relaxation.

a) Comparison to dual coordinate descent: Our next experiment investigates the viability of another distributed solution of the MWM problem, via coordinate descent (CD) on the dual of the LP relaxation, and compares the performance of dual coordinate descent to that of max-product. For clarity we consider the simple MWM problem (b-matching with $b = 1$), which has the following dual of its LP relaxation (e and (i, j) denote the same edge):

$$\begin{aligned} \text{DUAL}: \quad & \min \quad \sum_{i \in V} m_i, \\ \text{s.t.} \quad & m_i + m_j \geq w_{ij} \quad \text{for all } e \in E, \\ & m_i \geq 0 \quad \text{for all } i \in V. \end{aligned}$$

It has an intuitive interpretation of assigning mass m_i to node i, and minimizing the total mass while satisfying that for each edge the mass on its vertices exceeds w_e.

To solve the MWM LP relaxation, we first solve the dual, and then use complementary slackness conditions to recover the optimal primal solution:

$$\begin{aligned} m_i \ (\sum_{e \in E_i} x_e - 1) &= 0 \\ x_e \ (m_i + m_j - w_e) &= 0. \end{aligned}$$

We consider only those MWM problems which have a unique integral solution for their LP primal. Then the primal optimum can be recovered by finding those edges that satisfy $m_i + m_j = w_{ij}$ for the dual optimum. Furthermore, if we start with a sub-optimal dual solution, then we also use this as a heuristic to get sub-optimal primal solutions.

Coordinate descent on the dual starts from a feasible solution, e.g. $m_i = \max_e w_e$ and at each step selects a node at random and makes an update $m_i^{t+1} = \min_{j \in \mathcal{N}(i)} (w_{ij} - m_j^t)_+$, where $(x)_+ = \max(x, 0)$. While the iterations are extremely simple, we shall see that plain coordinate descent can converge without reaching the global minimum. To alleviate this issue, we also consider a smooth version of the dual problem:

$$\begin{aligned} \text{SM.DUAL} \quad & \min \left(\sum_i m_i - \epsilon \sum_{e \in E} \log(m_i + m_j - w_{ij}) \right) \\ \text{s.t.} \quad & m_i \geq 0 \quad \text{for all } i \in V. \end{aligned}$$

For problems where the LP primal has a unique integral optimum, the primal solution can be recovered from the dual solution, as $\epsilon \to 0$. We use coordinate descent on SM.DUAL, which selects a node and updates it as follows:

$$m_i^{t+1} = \min_{m_i \geq 0} \left(m_i - \epsilon^t \sum_{j \in \mathcal{N}(i)} \log(m_i + m_j^t - w_{ij}) \right).$$

This is a simple one-dimensional optimization problem. We gradually decrease ϵ as we run coordinate descent, as $\epsilon^{t+1} = 0.99\epsilon^t$, and keep it fixed after it reaches some fixed small threshold e.g. 10^{-6}. We produce the matchings at intermediate steps by relaxing the requirement that $m_i + m_j = w_{ij}$ to $m_i + m_j - w_{ij} \leq 2\alpha$, where $\alpha = \min_e (m_i + m_j - w_{ij})$.

We apply max-product, plain dual CD, and smooth dual CD to a MWM problem on a graph of $N = 40$ nodes, where the LP relaxation solution is integral. We plot the weight of the matching selected after n updates of max-product in Figure 4, the same graph for plain dual CD in Figure 5, and one

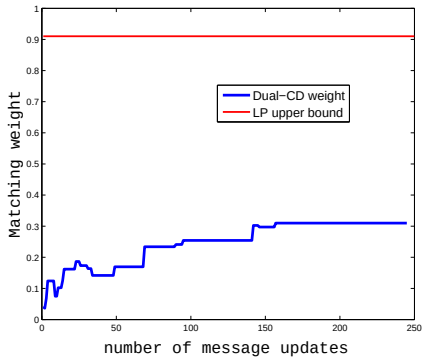

Fig. 5. Plain dual CD matching weight vs. node updates.

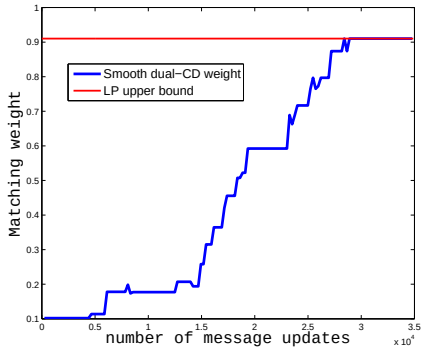

Fig. 6. Smooth dual CD matching weight vs. node updates.

for smooth dual CD in Figure 6. We fix intermediate invalid matchings as described in footnote 4. The simple version of dual CD converges extremely rapidly (in a few passes through the graph), but very often gets stuck at very bad solutions: in Figure 5 the resulting matching has about one third of the weight of the optimal matching. Max-product converges in about a dozen passes through the graph to the optimal solution. The smooth dual CD solution takes much longer to converge to the optimal solution, and in addition it has the undesirable property of yielding very poor matchings upon early termination. Thus, while smooth dual CD can also find the optimal matchings for simple MWM problems, max-product is considerably more efficient.

III. Locating Fusion Centers

A common model for sensor network aggregation is to have the sensors forward their data to one or more fusion centers, which then either process the data, store it for future reference or transmit it to locations further away. The sensor network operator/designer that decides to use fusion centers needs to tradeoff between two requirements:

(a) having each sensor close to its assigned fusion center, so that sensors can reliably forward their data with low latency and power consumption.

(b) limit the total number, and cost, of opening fusion centers. For example, it would be infeasible to open a fusion center for every sensor.

A classical optimization problem that exactly captures this tradeoff is the *facility location problem*. There are many variants of this problem, each with differing requirements on where the facilities can be placed, and what/how many customers(in our case sensors) they can serve. Nice surveys on this classical NP-hard problem can be found in [19], [20], for example. We will concern ourselves with the uncapacitated facility location problem, which we describe below.

This section has been inspired by the empirical success of Affinity Propagation [18], for exemplar-based clustering. The underlying motivation there is the same as ours, and they too apply BP for their problem. However, they use a different MRF, one that is *not* equivalent to the uncapacitated facility location problem described here. That said, many of the simplifications we derive below are similar to the ones derived in [18]. For our problem, we empirically observe several interesting properties connecting the performance of BP with the LP relaxation of uncapacitated facility location.

A. Uncapacitated Facility Location

We now paraphrase the standard uncapacitated facility location problem to fit the setting of this paper. We are given a set of sensors $V = \{1, \ldots, n\}$, and a set of edges E connecting them. For example, edges might represent the physical range of communication. Any sensor can be possibly upgraded, at a cost, to a fusion center. Any sensor that is *not* a fusion center has to be assigned to a neighbor that is a fusion center (for the moment, we consider only this kind of "one hop" service). We need to determine where to have the fusion centers, and how to make the subsequent assignments.

The uncapacitated facility location problem puts this in a cost minimization framework. Upgrading sensor $i \in V$ to a fusion center incurs a cost c_{ii}. Assigning sensor i to a fusion center at $j \in \mathcal{N}(i)$ incurs a cost c_{ij}, ($\mathcal{N}(i)$ denotes the neighborhood of i). Our objective is to decide where to locate the fusion centers, and then assign sensors to them, so as to minimize the total cost. This problem can be stated as the following integer program. There is a variable x_{ij} for each *ordered* pair of $(i, j) \in E$ (the variables x_{ij} and x_{ji} are distinct). A value $x_{ij} = 1$ means that i is assigned to a center at j, and $x_{ij} = 0$ means that it is not. Also, for each node j there is a variable y_j: $y_j = 1$ means that j has been upgraded to a fusion center, and $y_j = 0$ means it has not.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_j c_{jj}y_j, \\ \text{s.t.} \quad & y_i + \sum_{j\in\mathcal{N}(i)} x_{ij} = 1 \quad \text{for all } i \in V, \\ & y_j \geq x_{ij} \quad \text{for all } j \in V \text{ and } i \in \mathcal{N}(j) \\ & x_{ij}, y_j \in \{0, 1\}. \end{aligned}$$

In words, we would like to minimize the total cost of opening fusion centers, and assigning sensors, subject to the constraints that *(i)* each sensor is either itself a fusion center, or is assigned to a neighboring fusion center, and *(ii)* if any sensor i is assigned to j, then j should be a fusion center.

The LP relaxation of this problem involves replacing the integer constraints with interval constraints $x_{ij}, y_j \in [0,1]$.

B. Applying BP

We now formulate a Markov random field (MRF) such that finding the MAP assignment of this MRF is equivalent to solving the integer program above. We then use BP as a distributed heuristic to find this MAP assignment. We also show how the message updates can be significantly simplified, involving two real numbers for every pair of nodes.

For every node i, we have a variable $s_i \in \{i \cup \mathcal{N}(i)\}$, which is to be interpreted as the *identity of the center that i is assigned to*. This center will be one of the neighbors of i, or i itself. Consider

$$p(s) \;\propto\; \prod_{(i,j)\in E} f_{ij}(s_i, s_j) \prod_i e^{-c_i s_i} \tag{7}$$

where f_{ij} checks consistency:

$$f_{ij} = \begin{cases} 0 & \text{if } s_i = j \text{ but } s_j \neq j \\ 0 & \text{if } s_i \neq i \text{ but } s_j = i \\ 1 & \text{else} \end{cases}$$

It is easy to see that $p(s) > 0$ only for those states corresponding to a valid solution, and that for these states it will be proportional to e^{-cost}. Thus maximizing $p(s)$ is equivalent to solving the IP above.

BP for this problem involves sensors iteratively sending messages to their neighbors. In standard form the message from i to j consists of k real numbers, where $k = |\mathcal{N}(j)| + 1$ is the number of possible values that s_j can take. However, these messages have tremendous amount of redundancy, and can be radically simplified to the following update rules, which we derive in the appendix:

Simplified Max-product for Facility Location

(o) Initially, $r^0_{i\to j} = a^0_{j\to i} = 0$ for all $(i,j) \in E$.

(i) The messages are updated as follows:

$$\begin{aligned} D &= \max\left\{ -c_{ii} + \sum_{k\neq j} r^t_{k\to i}\,,\; \max_{k\neq i,j} -c_{ik} + a^t_{k\to i} \right\} \\ r^{t+1}_{i\to j} &= \max\{0\,,\, -c_{ij} - D\} \\ a^{t+1}_{i\to j} &= -c_{ii} + \sum_{k\neq j} r^t_{k\to i} - D \end{aligned}$$

(ii) At time t, for each node i:

- i is a fusion center if and only if
$$\sum_k r_{k\to i} > \max_k a_{k\to i}$$
- i is assigned to a center at j if and only if
$$a_{j\to i} > \sum_k r_{k\to i} \quad \text{and} \quad a_{j\to i} > \max_{k\neq j} a_{k\to i}$$
- otherwise, the status of i at t is undecided.

Note that in the above algorithm we have deliberately not mentioned which edges are updated in any given iteration. While the version with full synchronous message updates is by far the most popular in the max-product literature, doing full synchronous updates in wireless sensor networks will lead to significant interference. Below we give an alternative asynchronous broadcast implementation. Furthermore, we also use damping, similar to the network formation section:

$$\begin{aligned} r^{t+1}_{i\to j} &= \lambda\left(\max\{0\,,\, -c_{ij} - D\}\right) + (1-\lambda) r^t_{i\to j} \\ a^{t+1}_{i\to j} &= \lambda\left(-c_{ii} + \sum_{k\neq j} r^t_{k\to i} - D\right) + (1-\lambda) a^t_{i\to j}. \end{aligned}$$

C. Broadcast Implementation

In the implementation of max-product given above, each node sends a separate message to every one of its neighbors. However, if the wireless medium is broadcast, so that any transmission by i is heard by all its neighbors, it is possible to significantly reduce the number and complexity of the transmissions. Suppose we want to update all the outgoing messages from node i. In the broadcast implementation, nodes transmit asynchronously. Node i broadcasts three numbers, which are received by each of its neighbors. These numbers are

$$\begin{aligned} s &= -c_{ii} + \sum_k r^t_{k\to i} \\ m_1 &= \max_{k\neq i} -c_{ik} + a^t_{k\to i} \\ m_2 &= \text{second}\max_{k\neq i} -c_{ik} + a^t_{k\to i}. \end{aligned}$$

Consider now a neighbor j of i which receives these three numbers. Suppose also that j: ***(a)*** knows the value of c_{ij}, its edge to i, and ***(b)*** has stored the values of $r^t_{j\to i}$ and $a^t_{j\to i}$, which are *its most recent messages to* i. Then, j can use the three numbers above to *compute* the new messages $r^{t+1}_{i\to j}$ and $a^{t+1}_{i\to j}$ that it should have received from i in a vanilla (i.e. non-broadcast) implementation.

To compute the new messages, j needs to be able to compute the value of D above. It is easy to see that

$$D \;=\; \max\left\{s - r^t_{j\to i}\,,\, u\right\},$$

where

$$u \;=\; \begin{cases} m_1 & \text{if } m_1 > -c_{ij} + a^t_{j\to i}, \\ m_2 & \text{if } m_1 = -c_{ij} + a^t_{j\to i}. \end{cases}$$

Note that $m_1 \geq -c_{ij} + a^t_{j\to i}$ always.

D. Empirical Observations

Empirically, BP performs very well with damping. Figures 7 and 8 show a typical example: here a 100-node network was optimally resolved within 20 iterations. We set the cost of an edge to be inversely proportional to the square of its length, and the cost of a fusion center to be the median cost of the edges coming out of it.

Based on numerical experiments, we make the following conjectures about the connection between the BP algorithm described in this section, and the LP relaxation of uncapacitated facility location.

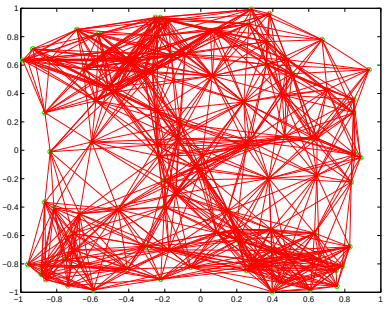

Fig. 7. Initial node placement for fusion center problem.

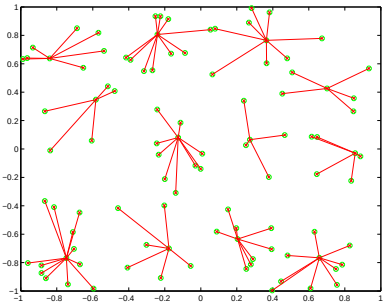

Fig. 8. Fusion center decision after 20 iterations.

1) In full-synchronous undamped BP, starting from the uninformative[5] initial condition, if node i is estimated to be a fusion center in two consecutive iterations, then $y_i^* = 1$ in the LP.
2) Any fixed point of BP corresponds to a valid solution. That is, for any node j, if any other node considers j to be its fusion center then $y_j = 1$ at the fixed point.
3) Any fixed point reached from the uninformative starting condition, and without damping, is the LP optimum (which in this case is also integral).

APPENDIX

A. Simplified message updates for MWbM

In this section we show how to reduce the standard factor-graph form max-product updates to obtain the simplified message updates in (2). The standard max-product updates are presented in the table below:

Max-Product for Weighted b-Matching

(o) Set $t = 0$ and initialize each message to 1.

(i) Iteratively compute new messages (factor to variable and variable to factor messages) until convergence as follows:

$$m_{e\to i}^{t+1}[x_e] \;=\; \exp(x_e w_e) \;\times\; m_{j\to e}^{t}[x_e] \qquad (8)$$

$$m_{i\to e}^{t+1}[x_e] \;=\; \max_{x_{E_i\setminus e}} \left\{ \psi_i(x_{E_i}) \prod_{e'\in E_i\setminus e} m_{e'\to i}^{t}[x_{e'}] \right\}$$

(ii) Also, at each t compute beliefs $n_e^t[x_e]$ and estimates for each edge e $\hat{x}^t \in \{0,1,?\}$ at time t:

$$\begin{aligned} n_e^t[x_e] &= \exp(w_e x_e) \;\times\; m_{i\to e}^t[x_e] \;\times\; m_{j\to e}^t[x_e], \text{ and} \\ \hat{x}_e^t &= 1 \quad \text{if } n_e^t[1] > n_e^t[0], \\ \hat{x}_e^t &= 0 \quad \text{if } n_e^t[1] < n_e^t[0], \\ \hat{x}_e^t &= ? \quad \text{if } n_e^t[1] = n_e^t[0]. \end{aligned}$$

[5]Uninformative IC means that all messages a, r are set to 0 initially.

We now mend them into a form suitable for sensor networks. First, instead of passing messages between edges and nodes we formulate a "node-to-node" protocol that is much more amenable to implementation in a distributed application. Let $e = (i,j)$ be the edge connecting the neighbors i and j, and $e' = (k,i)$. Combining $m_{i\to e}^{t+1}[x_e]$ with $m_{e\to j}^{t+1}[x_e]$, we have a node-to-node message update:

$$m_{i\to j}^{t+1}[x_e] = \exp(x_e w_e) \max_{x_{E_i\setminus e}} \left\{ \psi_i(x_{E_i}) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}^t[x_{e'}] \right\}.$$

For pedagogical purposes we derive the updates for simple matching first, and then extend them to b-matching. The constraint factor $\psi_i(x_{E_i})$ for MWM enforces that if $x_e = 1$ then $x'_e = 0$ for all $e' \in E_i\setminus e$. Hence

$$\log m_{i\to j}^{t+1}[1] = w_e + \sum_{k\in\mathcal{N}(i)\setminus j} \log m_{k\to i}^t[0].$$

However, if $x_e = 0$, then one of the neighboring edges can have value 1, hence,

$$\log m_{i\to j}^{t+1}[0] = \sum_{k\in\mathcal{N}(i)\setminus j} \log m_{k\to i}^t[0] + \max_k \left(\log m_{k\to i}^t[1] - \log m_{k\to i}^t[0]\right)_+ .$$

Now let

$$a_{i\to j}^t = \log\left(\frac{m_{i\to e}^t[0]}{m_{i\to e}^t[1]}\right) = \log\left(\frac{m_{i\to j}^t[0]}{m_{i\to j}^t[1]}\right) + w_{ij},$$

and after some simple algebra, we recover the simplified message updates in (2):

$$a_{i\to j}^{t+1} = \max_{k\in\mathcal{N}(i)\setminus j} \left(w_{ik} - a_{k\to i}^t\right)_+ .$$

The same sequence of reductions leads to the simplified algorithm for general b-matching. There, the constraint factor $\psi_i(x_{E_i})$ enforces that if $x_e = 1$ then we can set at most $b-1$ neighboring edges to 1, while if $x_e = 0$ then we can set b neighboring edges to 1. Hence, the difference between the two messages amounts to rank-b element in the set $\{(\log m_{k\to i}^t[1] - \log m_{k\to i}^t[0])_+ \mid k \in \mathcal{N}(i)\setminus j\}$, and the simplified update rule is given by:

$$a_{i\to j}^{t+1} \;=\; \text{rank}_b\left\{(w_{ik} - a_{k\to i}^t)_+ \mid k \in \mathcal{N}(i)\setminus j\right\}.$$

B. BP for Facility Location

In this section we derive the simplified message update rules given in Section III-B starting from the MRF $p(s)$ in (7). Recall that $m_{i\to j}(u)$ is the message from i to j for the case when $s_j = u$, i.e. the state of j is u. The standard max-product update equation is

$$m_{i\to j}^{t+1}(s_j) = \max_{s_i} f_{ij}(s_i, s_j)\, e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^t(s_i).$$

Now we consider the various values that s_j can take. First, if $s_j = i$, then $f_{ij}(s_i, s_j) = 1$ only when $s_i = i$, so

$$m_{i\to j}^{t+1}(i) \;=\; e^{-c_{ii}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^{t}(i).$$

On the other hand, if $s_j = j$ then $f_{ij}(s_i, s_j) = 1$ for any value of s_i, so

$$m_{i\to j}^{t+1}(j) \;=\; \max_{s_i} e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^{t}(s_i).$$

Finally, for any other value $s_j \neq i, j$, $f_{ij}(s_i, s_j) = 0$ only for $s_i = j$. Thus

$$m_{i\to j}^{t+1}(s_j \neq i, j) \;=\; \max_{s_i\neq j} e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^{t}(s_i).$$

Notice now that in the above equation, the value of $m_{i\to j}^{t+1}(s_j)$ is the *same* for all values of $s_j \neq i, j$. This is the crucial observation that allows us to obtain a much simpler update rule. We will denote this value by $m_{i\to j}^{t}(s_j \neq i, j)$ In particular, for each time t, define

$$r_{i\to j}^{t} \;=\; \frac{m_{i\to j}^{t}(j)}{m_{i\to j}^{t}(s_j \neq i, j)} \tag{9}$$

$$a_{i\to j}^{t} \;=\; \frac{m_{i\to j}^{t}(i)}{m_{i\to j}^{t}(s_j \neq i, j)}. \tag{10}$$

Consider first the update for r. Using the update rules for m we have that

$$\begin{aligned} r_{i\to j}^{t+1} &= \frac{m_{i\to j}^{t+1}(j)}{m_{i\to j}^{t+1}(s_j \neq i, j)} \\ &= \max\left\{1\,,\, \frac{e^{-c_{ij}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^{t}(j)}{\max_{s_i\neq j} e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} m_{k\to i}^{t}(s_i)}\right\} \\ &= \max\left\{1\,,\, \frac{e^{-c_{ij}}}{\max_{s_i\neq j} e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} \frac{m_{k\to i}^{t}(s_i)}{m_{k\to i}^{t}(j)}}\right\}. \end{aligned}$$

Now, as noted before, at time t we have that $m_{k\to i}^{t}(s_i)$ does not depend on the particular value of s_i, for all values $s_i \neq i, k$. Now $j \neq i, k$ so this means that $m_{k\to i}^{t}(j) = m_{k\to i}^{t}(s_i \neq i, k)$. Let us define

$$D \;=\; \max_{s_i\neq j}\; e^{-c_{is_i}} \prod_{k\in\mathcal{N}(i)-j} \frac{m_{k\to i}^{t}(s_i)}{m_{k\to i}^{t}(s_i \neq i, k)}, \tag{11}$$

so that $r_{i\to j} = \max\{1, \frac{e^{-c_{ii}}}{D}\}$. We now simplify D. Suppose first that the maximum in (11) is attained at $s_i = i$. Note that in this case

$$\frac{m_{k\to i}^{t}(s_i)}{m_{k\to i}^{t}(s_i \neq i, k)} \;=\; r_{k\to i}^{t}$$

by the definition (9) of r. Thus in this case $D = e^{-c_{ii}} \prod_{k\in\mathcal{N}(i)-j} r_{k\to i}^{t}$. Now suppose that the maximum in (11) is attained by some $s_i = k^* \neq i, j$. In this case

$$D \;=\; e^{-c_{ik^*}} \frac{m_{k^*\to i}^{t}(k^*)}{m_{k\to i}^{t}(s_i \neq i, k^*)} \prod_{k\in\mathcal{N}(i)-j-k^*} \frac{m_{k\to i}^{t}(k^*)}{m_{k\to i}^{t}(s_i \neq i, k)}.$$

Now for all $k \neq i, k^*$, we have that $\frac{m_{k\to i}^{t}(k^*)}{m_{k\to i}^{t}(s_i\neq i,k)} = 1$. Also, by the definition (10), $\frac{m_{k^*\to i}^{t}(k^*)}{m_{k\to i}^{t}} = a_{k^*\to i}^{t}$. Thus if the maximum in (11) is attained by some $k^* \neq i, j$ then $D = e^{-c_{ik^*}} a_{k^*\to i}^{t}$. Combining the above reasoning, we get that

$$D \;=\; \max\left\{ e^{-c_{ii}} \prod_{k\in\mathcal{N}(i)-j} r_{k\to i}^{t}\,,\, \max_{k\neq i,j} e^{-c_{ik}} a_{k\to i}^{t} \right\}. \tag{12}$$

The simplification of the update rule for a_{ij}^{t+1} follows along similar lines, yielding

$$a_{ij}^{t+1} \;=\; \frac{e^{-c_{ii}} \prod_{k\in\mathcal{N}(i)-j} r_{k\to i}^{t}}{D}.$$

The update rules given in Section III can be obtained by replacing a by $\log a$, and r by $\log r$.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *ISIT*, Sept. 2005, pp. 1763 – 1767.

[3] M. Bayati, C. Borgs, J. Chayes, R. Zecchina, "Belief-propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions," online at arxiv.org/abs/0709.1190

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. John Wiley and Sons, 1998.

[5] S. Sanghavi and D. Shah, "Tightness of LP via max-product belief propagation," online at arxiv.org/abs/cs/0508097

[6] A. Braunstein, M. Mezard, R. Zecchina "Survey propagation: an algorithm for satisfiability," in *Random Structures and Algorithms* 27, 201-226 (2005)

[7] J. Edmonds, "Paths, trees and flowers," *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965.

[8] B. Huang and T. Jebara, "Loopy belief propagation for bipartite maximum weight b-matching," in *Artificial Intelligence and Statistics (AISTATS)*, March 2007.

[9] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[10] X. Y. Li, P. J. Wan, Y. Wang, and O. Frieder, "Sparse power efficient topology for wireless networks," in *Proc. IEEE Hawaii Int. Conf. on System Sciences*, Jan. 2002.

[11] D. Malioutov, J. Johnson, and A. Willsky, "Walk-sums and belief propagation in Gaussian graphical models," *Journal of Machine Learning Research*, vol. 7, pp. 2031–2064, Oct. 2006.

[12] J. Pearl. *Probabilistic inference in intelligent systems*. Morgan Kaufmann, 1988.

[13] S. Sanghavi, D. Malioutov, and A. Willsky. Linear programming analysis of loopy belief propagation for weighted matching, *NIPS*, 2007.

[14] S. Sanghavi, D. Malioutov, and A. Willsky. Belief propagation and LP relaxation for weighted matching in general graphs, *submitted to IEEE Trans. Inf. Theory.*, 2008.

[15] S. Tatikonda and M. Jordan, "Loopy belief propagation and Gibbs measures," in *Uncertainty in Artificial Intelligence*, vol. 18, 2002, pp. 493–500.

[16] Y. Weiss and W. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.

[17] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring AI in the new millennium*, pages 239–269, 2003.

[18] B. Frey and D. Dueck, "Clustering by passing messages between data points," *Science* vol. 315, pp 972-976.

[19] G. Cornuejols, G. Nemhauser and L. Wosley, *The uncapacitated facility location problem* in P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, John Wiley and Sons, Inc., New York, 1990, pp 119-171

[20] D. Shmoys, E. Tardos and K. Aardal, *Approximation algorithms for facility location problems.* Proc. 29^{th} ACM symposium on Theory of Computing (1997), pp. 265-274.