Applications of Belief Propagation in CSMA Wireless Networks

Cai Hong Kai and Soung Chang Liew, *Fellow, IEEE*

***Abstract*—"Belief propagation" (BP) is an efficient way to solve "inference" problems in graphical models, such as Bayesian networks and Markov random fields. It has found great success in many application areas due to its simplicity, high accuracy, and distributed nature. This paper is a first attempt to apply BP algorithms in CSMA wireless networks. Compared to prior CSMA optimization algorithms such as ACSMA, which are *measurement-based*, BP-based algorithms are *proactive* and *computational*, without the need for network probing and traffic measurement. Consequently, BP-based algorithms are not affected by the temporal throughput fluctuations and can converge faster. Specifically, this paper explores three applications of BP. 1) We show how BP can be used to compute the throughputs of different links in the network given their access intensities, defined as the mean packet transmission time divided by the mean backoff countdown time. 2) We propose an inverse-BP algorithm to solve the reverse problem of how to set the access intensities of different links to meet their target throughputs. 3) We introduce a BP-adaptive CSMA algorithm to find the link access intensities that can achieve optimal system utility. The first two applications are NP-hard problems, and BP provides good approximations to them. The advantage of BP is that it can converge faster compared to prior algorithms like ACSMA, especially in CSMA networks with temporal throughput fluctuations. Furthermore, this paper goes beyond BP and considers a generalized version of it, GBP, to improve accuracy in networks with a loopy contention graph. The distributed implementation of GBP is nontrivial to construct. A contribution of this paper is to show that a "maximal clique" method of forming regions in GBP: 1) yields accurate results; and 2) is amenable to distributed implementation in CSMA networks, with messages passed between one-hop neighbors only. We show that both BP and GBP algorithms for all three applications can yield solutions within seconds in real operation.**

***Index Terms*—Belief propagation, CSMA, IEEE 802.11.**

I. Introduction

BELIEF propagation (BP) is a popular algorithm to solve "inference" problems in graphical models, such as Bayesian networks and Markov random fields (MRFs) [1]. Thanks to its simplicity, high accuracy, and distributed nature, BP has been widely used and has demonstrated empirical success in various areas such as artificial intelligence, information theory, and computer vision [2]–[4]. The Bayesian belief propagation [2], the iterative decoding algorithms for LDPC codes [3], the VISTA-Vision algorithm [4], and the transfer-matrix approach in physics [5] are special cases of BP algorithms discovered in different scientific communities.

The BP algorithm was first proposed by Judea Pearl in 1988, assuming that the underlying graphical model is a tree [2]. Specifically, given a treelike graphical model, we can obtain a natural iterative inference-making algorithm in which variable nodes pass messages among each other along edges of the graphical model. For tree graphs, it can be shown that the BP algorithm is an implementation of the Bayesian's formula, and it yields exact result with no approximation. The BP algorithm was applied to "loopy" graphical models with the simplifying assumption that the graphical model is still a tree (i.e., the same Bayesian formula for tree graphs is used in loopy graphs). Somewhat surprisingly, this BP heuristic has been found to work well empirically in terms of the approximate results it yields, as well as convergence [1], [2].

This paper considers three applications of BP in CSMA networks. Specifically, we use BP to solve two NP-hard problems in CSMA networks approximately and propose a new adaptive utility-optimization algorithm for CSMA networks. Compared to prior measurement-based algorithms for CSMA network control and optimization [6], BP-based algorithms are *proactive* and *computational*, without the need for network probing and traffic measurement. Consequently, BP algorithms are not affected by the temporal throughput fluctuations and can converge faster. Going beyond BP, we consider a generalized version of it, GBP, to improve accuracy in networks with a loopy contention graph. Importantly, we show how the BP and GBP algorithms can be implemented in a *distributed* manner in CSMA networks, making them useful in real operation. To the best of our knowledge, this is the first paper to use the BP framework to solve problems related to CSMA networks.

The first and the most direct application considered by us is to use BP to compute (infer) the throughputs of different links in a CSMA network given their access intensities. The access intensity of a link is the ratio of its average packet transmission time to its average backoff countdown time. Higher access intensity corresponds to higher aggressiveness of the link when it competes with other links for airtime under the CSMA protocol. The problem of computing link throughputs of a CSMA network is known to be NP-hard [7]. We show, however, that BP can obtain accurate approximate results within a short time, and its improved algorithm, GBP, can reduce the errors induced by loops significantly.

The second application is the reverse problem of computing link access intensities to meet some target link throughputs. We

Manuscript received October 18, 2010; revised July 17, 2011; accepted November 06, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Liu.This work was supported in part by the University Grants Committee of Hong Kong under Project No. AoE/E-02/08 and GRF Project No. 414911, the Chinese University of Hong Kong under Direct Grant Project No. 2050464, and the NSF of China under Project No. 60902016.

C. H. Kai is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the School of Computer and Information, Hefei University of Technology, Hefei 230009, China (e-mail: caihong.kai@gmail.com).

S. C. Liew is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: soung@ie.cuhk.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2177994

propose an Inverse Belief Propagation (IBP) algorithm for this purpose. IBP works well in terms of both speed and accuracy. Analogous to GBP, we propose Inverse GBP (IGBP) to reduce the errors in the access intensities found. Importantly, we show that IBP and IGBP can be easily implemented in a distributed manner, with messages passed between neighbors only. For a network of up to 200 links, both algorithms can yield solutions within seconds in real network operation.

The third application is on network utility optimization. We propose a BP-adaptive CSMA algorithm (BP-ACSMA) to achieve the optimal system utility. Compared to prior work, an advantage of BP-ACSMA is that it is a *proactive* computational algorithm without the need for network probing and traffic measurement. Thus, it will not be affected by the temporal throughput fluctuations in CSMA wireless networks [8]. We further propose a generalized version of BP-ACSMA, GBP-ACSMA, for higher accuracy in loopy graphs. Our simulation results indicate that the achieved aggregate throughputs and system utility are near-optimal.

Related Work: There have been numerous publications on throughput analysis and optimization of CSMA wireless networks (e.g., IEEE 802.11 networks). With respect to throughput analysis, [9] considered the throughput computation of CSMA wireless networks under the assumption of Poisson arrivals and exponential packet transmission durations. The authors of [9] showed that the link throughputs can be computed from a Markov chain, yielding a closed-form solution. Reference [10] made the same assumptions to investigate how the contention window in the CSMA protocol can be adapted to achieve a specific fairness target. Reference [11] studied interesting phenomena (e.g., border effects and phase transition) in large regular CSMA networks when the access intensity is high. Recent work [7] removed the Markovian assumption by showing that the throughput distribution is insensitive to the distributions of the packet transmission duration and backoff countdown time. In addition, [7] proposed a quick "back-of-the-envelope" (BoE) algorithm for link throughputs computation in CSMA wireless networks. As shown there, BoE could handle networks of up to 50 links with high accuracy and speed. Networks of larger size were left as an open issue. The BP/GBP algorithm proposed in this paper fills this gap.

With respect to network control and optimization, making use of an ideal carrier-sensing network model similar to that in [7] and this paper (see Section II), [6] proposed an elegant adaptive CSMA (ACSMA) algorithm to achieve optimal network utility. The ACSMA algorithm is a "probe and measure" scheme. Specifically, before a link adjusts its access intensity, a period of "smoothing" time is needed to measure the difference in the link's input traffic and output traffic. As will be shown in this paper, the required smoothing time can be quite excessive in networks that exhibit temporal starvation, resulting in very slow convergence. By contrast, the BP/GBP-based algorithms in this paper do not have this problem because they are computation-based rather than measurement-based.

BP as an inference-making methodology has been studied extensively. A good reference for BP is [1]. Reference [1] also presented GBP, without focusing on specific application domains. In particular, the treatment of GBP in [1] was general, and distributed implementation was not considered. An important contribution of our paper is to show that a "maximal clique" method of forming "regions" in GBP allows us to design *distributed* GBP algorithms for CSMA networks. Furthermore, this region-forming method yields good performance in CSMA networks.

In graphical models, there are two key inference problems: 1) the computation of marginal distributions of random variables; and 2) finding the maximum a posteriori probability. BP can be adapted to solve both algorithms, known as the "sum-product algorithm" and "max-product algorithm," respectively. The applications of BP in CSMA networks fall into the first category.

General necessary convergence conditions of loopy BP (LBP) to a unique fixed-point solution are still unknown, although sufficient conditions have been investigated extensively in the literature [1], [12]–[15]. References [1] and [15] related the convergence of LBP to the uniqueness of a sequence of Gibbs measures defined on the associated computation tree and derived sufficient conditions for convergence of BP. References [12] and [14] investigated the message update rules and examined whether the message update functions of LBP algorithms are contraction mapping. Message error propagation has been studied to understand how the performance of convergence will be affected [13]. In general, BP convergence is nontrivial to prove. References [1] and [12]–[15] made an assumption that the compatibility function (to be defined in Section III) is positive, which unfortunately does not hold in our BP in CSMA networks. With respect to the convergence of BP in this paper, we can only prove that IBP is a contraction mapping and guaranteed to converge. The rigorous convergence proofs of BP and BP-ACSMA, await future work. We remark that in our simulation studies, we observe convergence at all time.

To save space, some derivations and results are omitted in this paper. They can be found in our technical report [16].

The remainder of this paper is organized as follows. Section II introduces our system model and reviews its equilibrium analysis. Section III shows how to use BP for link throughput computation in large CSMA wireless networks. Section IV investigates the reverse problem: given the target link throughputs, how to find the link access intensities to meet them. Section V proposes BP-ACSMA for network utility optimization. Section VI shows how GBP can be used to solve the same problems as in Sections III–V, but with higher accuracy. Section VII concludes this paper.

II. System Model

In this section, we first review an idealized version of the CSMA network (ICN) to capture the main features of the CSMA protocol responsible for the interaction and dependency among links. The ICN model was used in several prior investigations [6], [7], [10], [11].[1]

A. ICN Model

In ICN, the carrier-sensing relationship among links is described by a contention graph $G = (V, E)$.[2] Each link is

[1]The correspondence between ICN and the IEEE 802.11 protocol [17] can be found in [7].

[2]Note that the pairwise carrier-sensing model is used in this paper. It is known that the cumulative interference model (or physical interference model) captures packet corruption more accurately compared to the pairwise interference model. However, it is possible to implement pairwise carrier sensing (as modeled by the contention graph) that can prevent collisions under the cumulative interference model [18]. In this case, hidden-node collisions can be effectively eliminated.

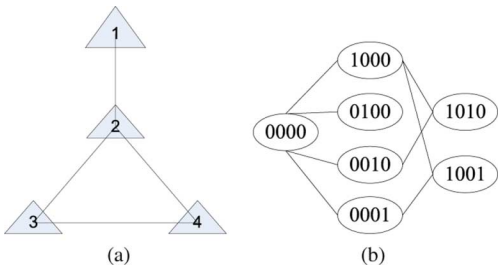

Fig. 1. (a) Example contention graph and (b) its state-transition diagram.

modeled as a vertex $i \in V$. Edges, on the other hand, model the carrier-sensing relationships among links. There is an edge between two vertices if the transmitters of the two associated links can sense each other. In this paper, we will use the terms "links" and "vertices" interchangeably.

At any time, a link is in one of two possible states, active or idle. A link is active if there is a data transmission between its two end nodes. Thanks to carrier sensing, any two links that can hear each other will refrain from being active at the same time. A link sees the channel as idle if and only if none of its neighbors is active.

In ICN, each link maintains a *backoff* timer C, the initial value of which is a random variable with an *arbitrary* distribution $f(t_{cd})$. The timer value of the link decreases in a continuous manner with $dC/dt = -1$ as long as the link senses the channel as idle. If the channel is sensed busy (due to a neighbor transmitting), the countdown process is frozen and $dC/dt = 0$. When the channel becomes idle again, the countdown continues with C initialized to the previous frozen value. When C reaches 0, the link transmits a packet. The transmission duration is a random variable with *arbitrary* distribution $g(t_{tr})$. After the transmission, the link resets C to a new random value according to the distribution $f(t_{cd})$, and the process repeats. We define the *access intensity* of a link as the ratio of its mean transmission duration to its mean backoff time: $\rho = E[t_{tr}]/E[t_{cd}]$.

Let $s_i \in \{0, 1\}$ denote the state of link i, where $s_i = 1$ if link i is active (transmitting) and $s_i = 0$ if link i is idle (actively counting down or frozen). The overall *system state* of ICN is $s = s_1 s_2 \cdots s_N$, where N is the number of links in the network. Note that s_i and s_j cannot both be 1 at the same time if links i and j are neighbors because: 1) they can sense each other; and 2) the probability of them counting down to zero and transmitting together is 0 under ICN (because the backoff time is a continuous random variable).

The collection of feasible states corresponds to the collection of independent sets of the contention graph. An independent set (IS) of a graph is a subset of vertices such that no edge joins any two of them [19].

As an example, Fig. 1(a) shows the contention graph of a network consisting of four links. In this network, link 1 can only sense link 2 while links 2–4 can sense each other. Fig. 1(b) shows the associated state-transition diagram under the ICN model. To avoid clutters, we have merged the two directional transitions between two states into one line in Fig. 1(b). Each transition from left to right corresponds to the beginning of the transmission of one particular link, while the reverse transition corresponds to the ending of the transmission of that link.

B. Equilibrium Analysis

This part is a quick review of the result in [7], and the reader is referred to [7] for details. If we assume that the backoff time and transmission time are exponentially distributed, then $s(t)$ is a *time-reversible* Markov process. For any pair of neighbor states in the continuous-time Markov chain, the transition from the left state to the right state occurs at rate $1/E[t_{cd}]$, and the transition from the right state to the left state occurs at rate $1/E[t_{tr}]$.

Let $\mathcal{S}$ denote the set of all feasible states, and n_s be the number of transmitting links when the system is in state $s = s_1 s_2 \cdots s_N$. The stationary distribution of state s can be shown to be

$$P_s = \frac{\rho^{n_s}}{Z} \forall\ s \in \mathcal{S}, \qquad \text{where } Z = \sum_{s \in \mathcal{S}} \rho^{n_s}. \tag{1}$$

The fraction of time during which link i transmits is $th_i = \sum_{s:s_i=1} P_s$, which corresponds to the normalized throughput of link i.

Reference [7] showed that (1) is in fact quite general and does not require the system state $s(t)$ to be a Markov process. In particular, (1) is insensitive to the distribution of the transmission duration $g(t_{tr})$ and the distribution of the backoff duration $f(t_{cd})$, given the ratio of their mean $\rho = E[t_{tr}]/E[t_{cd}]$.

Note that Z in (1) is a weighted sum of independent sets of G. In statistical physics, Z is referred to as the partition function, and the computation of Z is the crux of many problems, which is known to be NP-hard [19].

For the case where different links have difference access intensities, (1) can be generalized by replacing ρ^{n_s} with the $\prod_{i:s_i=1 \text{ in } s} \rho_i$, where ρ_i is the access intensity of link i.

Importantly, [7] showed that the contention graph associated with ICN is an MRF with respect to the probability distribution of its system states. This result suggests that the ICN model may allow us to generate useful network-control algorithms based on the BP framework. This is the main thrust of this paper.

C. Brief Discussion of Collision Effects in CSMA Networks

Here, we briefly discuss the collision effects in CSMA networks. Generally, in practical CSMA networks, there are two kinds of collision events: 1) hidden-node collisions caused by two nodes that cannot sense each other but yet can interfere with each other; 2) backoff collisions caused by two nodes (who can mutually sense each other) counting down to zero simultaneously in their backoff process and then transmitting together. We assume that the CSMA network is hidden-node-free. This can be achieved by properly setting the carrier-sensing power threshold as in [18].

Backoff collisions are unavoidable in practical networks, especially when the network is highly populated. The ICN model is an idealized model that models away such collisions by the adoption of a continuous-time countdown process. That is, it models a system in which the "minislot" used in the countdown backoff process is very small, and that carrier sensing is instantaneous. In [20], we proposed a generalized ICN (GICN) model for a perturbation analysis that tries to capture the effects of backoff collisions. However, as can be seen from [20, Section V], the effect of collisions is not significant as far as the link throughputs are concerned. That is, the original ICN model yields good approximations with errors

within 10% even though it does not incorporate the collision effect. Furthermore, the perturbation analysis based on GICN is topology-dependent, and a general mechanism to transform it to a general BP algorithm is lacking. For the above two reasons, in this paper, we stick to the original ICN model. We remark that the main goal of our BP algorithms is to compute results in the right ballpark so that we can use them in network design and control (e.g., our BP-ACSMA): Prevention of link starvation [7] is more important than exact throughputs. In that respect, we believe that ICN suffices.

III. Throughput Computation Using BP

This section describes a direct application of BP in CSMA networks: quick computation of the throughputs of links.

A. Motivation

As mentioned earlier, exact link throughput computation using (1) is an NP-hard problem. Thus, the problem can become intractable for large CSMA networks. As detailed in [7], for networks of more than 100 links, ICN computation can be rather time-consuming. An outstanding problem is to find quick and accurate approximate methods for large-scale CSMA wireless networks. This section is dedicated to this pursuit using BP.

B. Mapping ICN to BP

Under the framework of BP, the dependency between the states s_i and s_j of two neighbor vertices i and j is captured using a compatibility function $\psi_{ij}(s_i, s_j)$, defined as follows:

$$\psi_{ij}(s_i, s_j) = \begin{cases} 0, & \text{if } s_i = 1, s_j = 1 \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

In other words, the state $s_i = 1$ and the state $s_j = 1$ are not compatible because under CSMA, the two neighbor links cannot transmit together.

Recall that for an isolated link i, $p_i(s_i = 1) = \rho_i/(1 + \rho_i)$, and $p_i(s_i = 0) = 1/(1 + \rho_i)$. That is, $p_i(s_i = 1) \propto \rho_i$ and $p_i(s_i = 0) \propto 1$ for an isolated link. Thus, we give a weight to each possible state s_i as follows:

$$\varphi_i(s_i = 1) = \rho_i; \varphi_i(s_i = 0) = 1. \tag{3}$$

In particular, $\varphi_i(s_i = 1)$ and $\varphi_i(s_i = 0)$ in (3) capture the relative likelihoods of states $s_i = 1$ and $s_i = 0$ if link i were an isolated link without neighbors.

It is not difficult to verify that the stationary probability of the system state $s = s_1 s_2, \ldots, s_N$ in (1) can be rewritten as

$$P_s = \frac{\prod_{(i,j)\in E} \psi_{ij}(s_i, s_j) \prod_{i\in V} \varphi_i(s_i)}{Z} \qquad \forall\, s \in 2^N. \tag{4}$$

Note that in (4), if s is infeasible, a compatibility term $\psi_{ij}(s_i, s_j)$ will be zero and P_s will be zero.

In ICN, the normalized throughput of link i is the marginal probability $p_i(s_i = 1) = \sum_{s:s_i=1} P_s$. In the context of BP, $p_i(s_i)$, $s_i \in \{0, 1\}$, corresponds to the belief at vertex i, denoted by $b_i(s_i)$, $s_i \in \{0, 1\}$.

C. Message Update Rules in BP

With the stationary distribution expressed in the form of (4), we next show how to use the BP algorithm to solve for $p_i(s_i)$. The reader is referred to [1] for a general and detailed treatment of BP. Here, we focus on BP as applied to CSMA networks only. When applying BP to CSMA networks, each vertex i has an "intrinsic" belief of what the value of $p_i(s_i)$ should be. This intrinsic belief corresponds to the state probabilities of an isolated link. For an isolated link, we have $p_i(s_i) \propto \varphi_i(s_i)$.

In addition, each vertex i receives messages from its neighbors as to what they "think" $p_i(s_i)$ should be. Let N_i denote the neighbors of vertex i in G. Each neighbor $j \in N_i$ passes a message $m_{ji}(s_i)$ to vertex i as to its "belief" of $p_i(s_i)$. The beliefs of link i and all links $j \in N_i$ are then aggregated into an overall belief in the form of a product

$$b_i(s_i) = c_i \varphi_i(s_i) \prod_{j\in N_i} m_{ji}(s_i) \tag{5}$$

where c_i is a normalization constant so that $\sum_{s_i\in\{0,1\}} b_i(s_i) = 1$.

The messages are determined by the message update rule

$$\begin{aligned} m_{ji}(s_i) &\leftarrow \sum_{s_j\in\{0,1\}} \psi_{ij}(s_i, s_j)\varphi_j(s_j) \prod_{k\in N_j\setminus i} m_{kj}(s_j) \\ &= \sum_{s_j\in\{0,1\}} \psi_{ij}(s_i, s_j) b_j(s_j)/(c_j m_{ij}(s_j)). \end{aligned} \tag{6}$$

Note that $\varphi_j(s_j) \prod_{k\in N_j\setminus i} m_{kj}(s_j) \propto b_j(s_j)/m_{ij}(s_j)$. That is, it is proportional to the aggregated belief at vertex j with the message from i to j factored out. In tree graphs, this message update rule can also be understood as the expression of the Bayes' formula [1].

The BP algorithm iterates (6) over all vertices i. In each iteration, we could normalize the messages according to $\sum_{s_i\in\{0,1\}} m_{ji}(s_i) = 1$ for $\forall\ j$.[3] The iteration stops when $m_{ji}(s_i)$ converges or a maximum number of iterations is reached.

D. Complexity of BP

It is known that (5) and (6) give exact solutions in tree graphs. Furthermore, each message needs only be computed once before convergence in tree graphs (i.e., the number of iterations equals to the diameter of the contention graph [19]) [2]. Appendix A of our technical report [16] shows that in a tree graph, the BP messages can be interpreted as the partition functions of subgraphs. This interpretation reveals why BP can give exact link throughputs in networks with loop-free contention graphs.

For loopy graphs, assuming that the maximum vertex degree in the contention graph (i.e., the maximal number of neighbors of vertices in the graph) is $N_{\max}$, from (6) we know that one summation and two $(N_{\max} + 1)$ multiplications are required in each iteration. Furthermore, the number of message exchanges in each iteration of each link is bounded by $N_{\max}$. Then, the complexity of the BP algorithm largely depends on its convergence quality. Unfortunately, as explained in Section I, in general the convergence of the BP algorithm for loopy graphs is nontrivial to prove. Empirically, as can be seen from the simulation results later, we observed that BP algorithms converge in dozens of iterations for various network topology settings. The complexity of BP algorithms can be roughly bounded as L summation and $2L(N_{\max} + 1)$ multiplications for each link, where

[3] If we normalize the beliefs in (5) without normalizing the messages, the magnitudes of the messages may grow unbounded, but not the beliefs themselves. Thus, the algorithm may still be well-behaved if the beliefs converge quickly.

L is the empirically observed number of iterations. We find that L is below 100 in all the network settings we tested.

E. Distributed BP

BP can be easily implemented in a distributed manner, with messages exchanged between two neighbors who can mutually carrier-sense each other. The distributed implementation is rather obvious, and interested readers are referred to [16, Section III-D] for details. This paper will only present the less obvious distributed implementation of GBP.

F. BP in Loopy Contention Graphs

Although BP can often give good approximations, as pointed out in [2], if we apply BP in loopy contention graphs, the information may circulate indefinitely around loops, and BP may give inaccurate solutions and even not converge.

Consider a triangular graph consisting of three vertices. In BP, messages are passed between each pair of neighboring vertices. Vertex 1 gives certain information to vertex 2, some of which is included in the information from vertex 2 to vertex 3 and finally passed back to vertex 1, where it is regarded as a "new" incoming message. This message contains information correlated with the original information at vertex 1. The message update rule and the belief computation formula, however, do not take this correlation into account.

When the loop is large, the information a vertex i gives out vanishes along the cycle back to i, resulting in a smaller computation error. It can be shown that BP converges to the fixed point $b_i(s_i = 0) = (1+\sqrt{1+4\rho})/2\sqrt{1+4\rho}$ for each vertex in any N-vertex ring graph regardless of N (the derivation can be found in [16, Appendix D]). Reference [21, Section III-A] derived the equilibrium throughputs of links in a 1-D ring network. Using the results above, we can evaluate the errors of BP in a ring graph in terms of N. Given a value of $\rho = \rho_0 = 83/15.5$ (typical in 802.11 networks), the errors of BP for different N are the following:

- 8% for the 3-vertex ring;
- 0.1% for the 8-vertex ring;
- zero for the N-vertex ring as $N \to \infty$.

That is, the error of BP decreases as the length of the cycle increases.

From the ring example, we see that small loops cause more significant errors. This suggests that to contain errors in general graphs, we want to eliminate small loops in message propagation. This is the basic idea behind GBP.

The theoretical details of GBP will be presented in Section VI. For easy comparison, we next present simulation results on the performance of GBP together with BP first.

G. Experimental Evaluation

Reference [7] proposed the BoE algorithm for link throughputs computation in CSMA wireless networks. BoE could only handle networks of up to 50 links. Networks of larger size were left as an open issue. Therefore, the focus of our experiments here is on the accuracy and speed of BP and GBP for networks of more than 50 links.

For our experiments, we implement both algorithms in a centralized manner using MATLAB programs. The throughputs computed by BP and GBP are compared to that obtained from an ICN-simulator to examine their accuracy. Furthermore, we list the average number of iterations a link performs before convergence. This will be used to estimate the convergence time for distributed implementation in real networks. In our experiments, we define the minimum n such that $\max_j \left|th_j[n] - th_j^*\right| / th_j^* < 1\%$ is satisfied as the number of iterations required by BP or GBP to achieve convergence, where th_j^* is the final converged value.[4, 5]

TABLE I
MEAN LINK THROUGHPUT ERRORS AND NUMBERS OF ITERATIONS REQUIRED FOR CONVERGENCE WITH BP AND GBP FOR TOPOLOGY SETTINGS 1

# of links		50	100	200
Mean Throughput Errors	BP	5.3%	6.5%	7.0%
	GBP	0.3%	0.3%	0.6%
Number of Iterations	BP	14	15	15
	GBP	22	28	32

TABLE II
MEAN LINK THROUGHPUT ERRORS AND NUMBERS OF ITERATIONS REQUIRED FOR CONVERGENCE WITH BP AND GBP FOR TOPOLOGY SETTINGS 2

Mean Vertex Degree		2	4	6
Mean Throughput Errors	BP	4.7%	7.0%	7.2%
	GBP	0.2%	0.3%	0.3%
Number of Iterations	BP	13	15	16
	GBP	24	28	35

We generate two sets of network topologies. Specifically, for Topology Settings 1, we randomly generate networks of different numbers of links while maintaining the mean degree of links (number of neighbors per link) to around 4. For Topology Settings 2, we fix the number of links to 100 while varying the mean degree of links. The access intensities of all links are set to ρ_0. For each link, we calculate the error of the throughput obtained by BP and GBP relative to the simulated throughput obtained from the ICN simulator. The error is normalized by the maximum link throughput in the network. For each topology setting, we randomly generate ten different topologies and the experimental data are averaged over the ten networks.

Tables I and II show the simulation results of BP and GBP for the two sets of network topologies, respectively. As can be seen, for all networks tested, the error of BP is kept to 7.2% or below, while the error of GBP is consistently lower than 1%.

In Tables I and II, the access intensity ρ is set to ρ_0. A question is how well these algorithms work under different ρ. When ρ is large, two neighbor vertices become more tightly coupled, and the message passing within a loop may incur more computational errors. Table III shows the accuracy of BP and GBP under different ρ, for a 100-link network with mean vertex degree of 4. As can be seen, the mean error of BP increases with ρ. More impressive is GBP, whose mean error is very small even for $\rho = 4\rho_0$. This shows that GBP performs well over a large range of ρ.

For all the scenarios, both algorithms converge within dozens of iterations. That is, if implemented in a distributed manner in which each link passes a message every 0.1 s (e.g., if we use beacons for message passing in a 802.11 network [17]), both

[4]We use exponential averaging to smooth out the computed messages for GBP: i.e., $m_{ave}[n] = (1-\alpha)m_{ave}[n] + \alpha m[n]$, $0 < \alpha < 1$, where $m[n]$ is the newly computed message and α is the smoothing factor. $m[n]$ is recomputed in each iteration. For BP, we do not perform the averaging procedure because it converges smoothly even without it.

[5]We define convergence has occurred when $E\left[|th_j[n] - th_j[n-1]|\right] < 10^{-3}$ for $n \geq N$ for some N. According to this criteria, all the BP and GBP algorithms in this paper converge in our simulations.

TABLE III
MEAN LINK THROUGHPUT ERRORS AND NUMBERS OF ITERATIONS REQUIRED FOR CONVERGENCE WITH BP AND GBP FOR NETWORKS OF DIFFERENT ρ

ρ/ρ_0		2	3	4
Mean Throughput Errors	BP	10.6%	12.3%	13.5%
	GBP	0.2%	0.3%	0.3%
Number of Iterations	BP	32	57	76
	GBP	45	78	92

BP and GBP can obtain links throughputs within seconds in real operation.

IV. COMPUTATION OF LINK ACCESS INTENSITIES GIVEN TARGET LINK THROUGHPUTS

This section proposes an "inverse" belief propagation (IBP) algorithm to compute the link access intensities required to meet target link throughputs.

A. Motivation

In network design, an interesting problem is as follows. Given a network contention graph and a set of target link throughputs, how to set the link access intensities $\bar{\rho}$ to meet the target link throughputs.

For small networks, we can find $\bar{\rho}$ by solving (1) and $th_i = \sum_{s:s_i=1} P_s$. However, as throughput computation using (1), the computation becomes intractable when the network is large. IBP gives approximate solutions within a short time.

B. Definition of IBP

As described in Section III, the operation of BP is as follows. Given the network contention graph G and the access intensities of links $\bar{\rho}$, BP computes the throughputs of links. That is, $\bar{th} = BP(G, \bar{\rho})$.

Definition of IBP: We define $\bar{\rho} = IBP^{-1}(G, \bar{th})$ as the inverse operation of belief propagation for $\bar{th} = BP(G, \bar{\rho})$, where $\bar{th}$ is the vector of target link throughputs.

C. Message Update Rules of IBP

Recall from Section III-B that the belief at vertex j, $b_j(s_j = 1)$ corresponds to the link throughput, then the belief of link j, $b_j(s_j)$ is given in IBP.

From (6), we obtain the message update rule

$$m_{ji}(s_i) \leftarrow \sum_{s_j \in \{0,1\}} \psi_{ij}(s_i, s_j) b_j(s_j)/(c_j m_{ij}(s_j)) \quad (7)$$

and from (5), we have

$$\rho_j = \varphi_j(s_j = 1) = \frac{b_j(s_j = 1) \prod_{i \in N_j} m_{ij}(s_j = 0)}{b_j(s_j = 0) \prod_{i \in N_j} m_{ij}(s_j = 1)}. \quad (8)$$

The IBP algorithm iterates (7) over all vertices j. Similar to BP, in each iteration we could normalize the messages according to $\sum_{s_i \in \{0,1\}} m_{ji}(s_i) = 1, \forall\, j$. The iteration stops when $m_{ji}(s_i)$ converges or a maximum number of iterations is reached. The link access intensities $\bar{\rho}$ is then computed by (8).

Note that IBP, being an approximate algorithm, has computation errors that can potentially result in nonconvergence of the algorithm. As will be demonstrated in Section VI-D, we can resort to IGBP for more accurate computation. Another reason for nonconvergence is due to the problem formulation itself and not the fault of IBP. In the problem formulation, we require the target $\bar{th}$ to be feasible, and then seek the $\bar{\rho}$ to achieve $\bar{th}$. If the given $\bar{th}$ is not feasible, then no matter what algorithm we use, there is no solution. Formulating the problem as a system utility optimization problem as in Section V removes this difficulty.

TABLE IV
MEAN THROUGHPUT ERRORS AND NUMBERS OF ITERATIONS REQUIRED FOR CONVERGENCE WITH IBP AND IGBP FOR TOPOLOGY SETTINGS 1

# of links		50	100	200
Mean Throughput Error	IBP	5.40%	3.27%	6.2%
	IGBP	0.11%	0.07%	0.79%
Number of Iterations	IBP	29	38	47
	IGBP	42	44	54

TABLE V
MEAN THROUGHPUT ERRORS AND NUMBERS OF ITERATIONS REQUIRED FOR CONVERGENCE WITH IBP AND IGBP FOR TOPOLOGY SETTINGS 2

Mean Vertex Degree		2	4	6
Mean Throughput Error	IBP	2.56%	3.27%	3.69%
	IGBP	0.02%	0.07%	0.77%
Number of Iterations	IBP	2	38	58
	IGBP	11	44	74

D. Convergence of IBP

Theorem 1: If the target throughput is feasible in the sense that $\bar{th} = BP(G, \bar{\rho})$ for some $\bar{\rho}$, IBP defined by (7) and (8) is a contraction mapping and is guaranteed to converge to $\bar{\rho}$.

Proof: See [16, Appendix B]. ■

E. Experimental Evaluation

We implement IBP and IGBP using MATLAB programs. We use the same two sets of network topologies as in Tables I and II. The access intensity of link i, ρ_i, is randomly generated within the interval $[\rho_0, 4\rho_0]$. Next, we run the ICN simulator to get the link throughputs and set them to be the target throughputs of IBP and IGBP. Then, we find the corresponding access intensity $\bar{\rho}$ to meet the target link throughputs using IBP and IGBP. We then use the ICN-simulator to get the throughputs with the access intensities found. For each link, we calculate the error of the throughput obtained by the ICN-simulator relative to the target throughput. In our experiments, we define the minimum n such that $\max_j \left|\rho_j[n] - \rho_j^*\right| / \rho_j^* < 1\%$ is satisfied as the number of iterations for IBP and IGBP to achieve convergence, where ρ_j^* is the final converged value.

Tables IV and V show the simulation results of IBP and IGBP for Network Topology Settings 1 and 2, respectively. The error of IBP is kept to 6.2% or below, while the error of IGBP is within 1%. As for convergence speed, both IBP and IGBP converge within dozens of iterations. In a real network, if periodic beacons of one beacon per 0.1s are used for message passing, IBP and IGBP can output solutions within seconds for network of up to 200 links.

V. BP-ADAPTIVE CSMA (BP-ACSMA)

This section investigates solving the network utility optimization problem in CSMA networks using BP.

A. Motivation and Problem Formulation

In Section IV, the target link throughputs are given, and the corresponding link access intensities are computed. A problem is that in general we do not know whether the target link throughputs are feasible—computation of the feasible region is itself a tough problem for large networks.

A way to circumvent this problem is to focus on optimizing a system utility $\sum_j U_j(th_j)$ instead, where $U_j(th_j)$ is the utility of link j. The algorithm then iterates within the feasible region to identify the $\bar{\rho}$ that optimizes the utility, without finding the whole feasible region explicitly. This is the basic principle behind the ACSMA algorithm proposed in [6]. In the following, we briefly review the background leading to ACSMA. Then, in Section V-C, we introduce the alternative of using BP to solve the problem.

Recall that the feasible states of ICN are the independent sets of the contention graph. Define an indicator function x_j^s such that $x_j^s = 1$ if link j is transmitting in state s and $x_j^s = 0$ otherwise. Let u_s be the probability of state s (i.e., fraction of airtime dedicated to state s), then the throughput of link j can be expressed as $th_j = \sum_s u_s x_j^s$. Furthermore, let f_j denote the input rate of link j. Let $\bar{u}$ and $\bar{f}$ denote the vectors consisting of u_s for all s, and f_j for all j, respectively. Consider the following utility optimization problem:

$$\begin{aligned} \max_{\bar{u},\bar{f}} \quad & \sum_j U_j(f_j) \\ \text{s.t.} \quad & \sum_s u_s x_j^s \geq f_j \qquad \forall\, j \\ & u_s \geq 0 \,\forall\, s; \quad \sum_s u_s = 1. \end{aligned} \tag{9}$$

As explained in [6], when the system is in a state s (i.e., $x_j^s = 1$) but link j has no packet in its queue, link j will transmit a dummy packet. This accounts for the inequality $\sum_s u_s x_j^s \geq f_j$ (rather than equality).

The optimization problem as formulated in (9) has two problems. First, it is a difficult combinatorial optimization problem. Also, it is not clear how to implement a distributed algorithm to solve it. Second, even if a solution could be found, to realize it using CSMA, the $\bar{u}$ found would still have to be mapped to $\bar{\rho}$. That is, u_s must be equal to the stationary probability $P_s = \prod_{j:x_j^s=1} \rho_j / Z$ for CSMA networks.

To circumvent the above difficulties, [6] formulated an alternative optimization problem as follows:

$$\begin{aligned} \max_{\bar{u},\bar{f}} \quad & \beta \sum_j U_j(f_j) - \sum_s u_s \log u_s \\ \text{s.t.} \quad & \sum_s u_s x_j^s \geq f_j \qquad \forall\, j \\ & u_s \geq 0 \,\forall\, s; \quad \sum_s u_s = 1. \end{aligned} \tag{10}$$

Compared to (9), the objective function in (10) has an extra entropy term $-\sum_s u_s \log u_s$. When β is large, (10) asymptotically approaches (9). As shown below, the $\bar{u}$ found by (10) is CSMA realizable. Indeed, $r_j = \log(\rho_j)$ turns out to be the dual variable to the constraint $\sum_s u_s x_j^s \geq f_j$.

Let us define dual variable r_j for the constraint $\sum_s u_s x_j^s \geq f_j \,\forall\, j$, without assuming $r_j = \log(\rho_j)$ for the time being. A partial Lagrangian of problem (10) is

$$L(\bar{u},\bar{r},\bar{f}) = \beta \sum_j U_j(f_j) - \sum_s u_s \log u_s + \sum_j r_j \left(\sum_s u_s x_j^s - f_j \right). \tag{11}$$

Given $\bar{r}$ and $\bar{f}$, the optimal $\bar{u}$ to (10) can be shown to be

$$u_s^*(\bar{r}) = \exp\left(\sum_j r_j x_j^s \right) \Big/ \sum_s \exp\left(\sum_j r_j x_j^s \right) \forall\, s. \tag{12}$$

We see that (12) is just the stationary distribution of CSMA networks with $r_j = \log(\rho_j) \,\forall\, j$.

The optimal $\bar{f}$ is given by

$$\partial L(\bar{u},\bar{r},\bar{f}) / \partial f_j = \beta U_j'(f_j) - r_j = 0. \tag{13}$$

The optimal $\bar{r}$ is given by

$$\partial L(\bar{u},\bar{r},\bar{f}) / \partial r_j = \sum_s u_s x_j^s - f_j = 0. \tag{14}$$

Combining (12)–(14), we find that the optimal solution to (10) is given by a set of $\bar{r}$ and $\bar{f}$ that satisfy

$$\begin{aligned} \beta U_j'(f_j) - \log(\rho_j) &= 0 \\ f_j = \sum_s u_s x_j^s &= th_j \end{aligned} \tag{15}$$

In Section V-B, we briefly review how [6] solves the optimization problem using a distributed adaptive CSMA algorithm. We present an alternative method using the BP framework in Section V-C.

B. ACSMA Proposed in [6]

The joint scheduling and congestion control algorithm (ACSMA) proposed in [6] looks for the optimal solution to (10) by steepest ascent of $L\left(\bar{u},\bar{r},\bar{f}\right)$. According to (14), $\partial L(\bar{u},\bar{r},\bar{f}) / \partial r_j =$ output rate of link j − input rate of link j. The queue size of link j is a smoothed measure of the difference in the output rate and input rate. Thus, in each iteration, link j adjusts its ρ_j such that $r_i = \log(\rho_i)$ is proportional to its queue length. If the input rate of the queue is larger than the output rate, the queue builds up, leading to an increase in ρ_j, and vice versa. Note that ρ_j controls the output rate of link j. For the input rate f_j, link j adjusts f_j to satisfy $\beta U_j'(f_j) - \log(\rho_j) = 0$ in (13) based on the newly computed ρ_j. Before the next iterative update, link j waits for some time to examine whether the load f_j can be supported by the network under the current $\bar{\rho}$ (by examining its queue size). The iterations continue until the overall network finds a set of access intensities $\bar{\rho}$ that can support the loads $\bar{f}$ (with stable queue sizes). At that point, $th_j = f_j \forall\, j$.

ACSMA does not explicitly "compute" the link throughputs using (12). Rather, it makes use of actual data packets to probe the network and "measure" the link throughputs. To smooth out the measurement due to temporal throughput fluctuations, long smoothing intervals between successive iterations may be required. Reference [8] shows that CSMA networks are susceptible to the so-called temporal starvation, where links may undergo long intervals in which they receive near zero throughputs even through their equilibrium throughputs are acceptable. For networks with temporal starvation (e.g., Cayley tree networks [19]), long smoothing intervals are needed in ACSMA.

C. BP-ACSMA

The optimal network utility in (10) is achieved when (15) is satisfied. BP can be applied to ensure that.

In BP-ACSMA, the message update rule is

$$m_{ji}(s_i) \leftarrow \sum_{s_j \in \{0,1\}} \psi_{ij}(s_i, s_j)\varphi_j(s_j) \prod_{k \in N_j \setminus i} m_{kj}(s_j). \quad (16)$$

In each iteration, based on the received messages, vertex j computes belief $b_j(s_j)$ according to

$$b_j(s_j) = c_j \varphi_j(s_j) \prod_{i \in N_j} m_{ij}(s_j). \quad (17)$$

It then solves for ρ_j from (15) by setting $th_j = b_j(s_j = 1)$. Based on the new ρ_j, vertex j updates messages $m_{ji}(s_i) \forall\, i \in N_j$ according to (16) and broadcasts the messages to its neighbors.

In essence, BP replaces the network probing and throughput measurement in ACSMA by computation.

D. Experimental Evaluation

We implement BP-ACSMA, GBP-ACSMA, and ACSMA using MATLAB programs (details of GBP-ACSMA will be presented in Section VI-D). We consider the proportional fairness utility, $U_j(th_j) = \log(th_j)$, and set the weighting factor β to 1. For BP-ACSMA and GBP-ACSMA, the outputs are the converged link access intensities, $\bar{\rho}^*$. To evaluate the performance, we use the ICN simulator to get the throughputs of networks with the $\bar{\rho}^*$ found, and then obtain the network utility achieved from the throughputs.

For easy comparison between our algorithms and ACSMA, we use $r_j = \log(\rho_j)$ in our convergence test although the parameters being adjusted in our algorithms are $\rho_j, \forall\, j$. Let $r_j[n]$ be the value of r_j in iteration n. We define the number of iterations required for convergence in BP-ACSMA (GBP-ACSMA) as the minimum n such that $\max_j \left|r_j[n] - r_j^*\right| / r_j^* < 1\%$, where $r_j^* = \log(\rho_j^*)$ is the final converged r_j value.

In ACSMA, the parameters adjusted in iteration n are $r_j[n]$ and $f_j[n]\ \forall\, j$. If ACSMA converges, then $r_j[n]$ and $f_j[n]$ will asymptotically approach the target r_j^* and f_j^* as n increases. We define the minimum n such that $\max_j \left|r_j[n] - r_j^*\right| / r_j^* < 3\%$ is satisfied as the number of iterations for ACSMA to achieve convergence. Note that here we use a looser convergence test for ACSMA because, by nature, some fluctuations are unavoidable in ACSMA even after convergence because of its measurement approach. In our simulation, the update interval of ACSMA is set to 150 DATA packet times to guarantee convergence: We find that if the update interval is set to 125 DATA packet times, ACSMA does not converge in some networks tested.[6]

In the first set of experiments, we randomly generate networks with different numbers of links. The mean degree of links is around 4. In each simulation run, we gather the statistics of two metrics: 1) normalized total system throughput $Th = \sum_j th_j$; 2) system utility $U = \sum_j \log(th_j)$. Table VI shows the achieved throughputs and network utilities of BP-ACSMA, GBP-ACSMA, and ACSMA. As shown, BP-ACSMA has acceptable performance in terms of both throughputs and network utilities; and GBP-ACSMA has comparable performance to ACSMA. As for speed, BP-ACSMA and GBP-ACSMA output solutions after dozens of iterations, while ACSMA often requires hundreds of iterations.

[6]This brings up another issue with ACSMA. That is, we do not know how to set the update interval T in an optimal manner beforehand, and we need to run the algorithm to determine the minimum T required for each network. BP-ACSMA and GBP-CSMA, however, do not have this issue because the update interval is not related to measurement smoothing time.

TABLE VI
ACHIEVED AGGREGATE THROUGHPUTS, UTILITIES, AND NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE IN BP-ACSMA, GBP-ACSMA, AND ACSMA FOR NETWORKS IN WHICH EACH VERTEX HAS ON AVERAGE FOUR NEIGHBORS

# of links		25	50	75
BP-ACSMA	Th	8.63	17.16	26.28
	U	-34.24	-68.97	-95.75
	Number of Iterations	7	9	9
GBP-ACSMA	Th	8.38	16.25	25.26
	U	-31.85	-65.41	-90.76
	Number of Iterations	34	43	44
ACSMA	Th	8.12	15.94	24.74
	U	-30.58	-62.47	-89.14
	Number of Iterations	296	311	382

TABLE VII
ACHIEVED AGGREGATE THROUGHPUTS, UTILITIES, AND NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE IN BP-ACSMA, GBP-ACSMA AND ACSMA FOR NETWORKS OF 100 LINKS

Mean Vertex Degree		2	4	6
BP-ACSMA	Th	43.89	32.74	26.55
	U	-92.53	-152.64	-218.03
	Number of Iterations	9	9	11
GBP-ACSMA	Th	46.33	34.45	25.79
	U	-90.64	-139.70	-179.65
	Number of Iterations	39	52	64
ACSMA	Th	43.49	32.14	25.12
	U	-90.45	-121.31	-147.91
	Number of Iterations	228	405	413

In the second set of experiments, we randomly generate networks of 100 links with varying mean vertex degrees. Table VII compares the three algorithms. As the network becomes denser, more loops appear in the contention graph, resulting in higher computation error in BP-ACSMA. As shown in Table VII, BP-ACSMA loses accuracy when the mean vertex degree is set to 6. GBP-ACSMA continues to work well since it has removed loops in message passing. Table VII also shows that BP-ACSMA and GBP-ACSMA achieve higher aggregate throughputs than ACSMA does with some utility loss. As for convergence speed, BP-ACSMA and GBP-ACSMA are much faster than ACSMA.

E. Comparison of BP-ACSMA and GBP-ACSMA to ACSMA

Simulations in Section V-D show that both BP-ACSMA and GBP-ACSMA converge within dozens of iterations for a network of 100 links. ACSMA converges only after hundreds of iterations. For comparison, let us map the number of iterations to time needed for convergence in real network operation.

For BP-ACSMA and GBP-ACSMA, beacons could be used for message passing. In 802.11 networks, typically a beacon is broadcast every 0.1 s. For BP-ACSMA, from the results in Tables VI and VII, convergence is achieved within 11 iterations for all the scenarios we tested. Using beacons for message passing, it only needs $0.1 \times 11 = 1.1$ s to output solutions for networks of up to 100 links. For GBP-ASMA, convergence is achieved within 64 iterations for all the scenarios tested, corresponding to a convergence time of within 6.4 s. The convergence speed of both algorithms can be even faster if the messages are piggybacked on data packets rather than being carried on beacons. By contrast, ACSMA requires 413×150 ms ≈ 62 s for convergence, assuming the DATA packet duration is 1 ms—recall that we experimentally found that we need

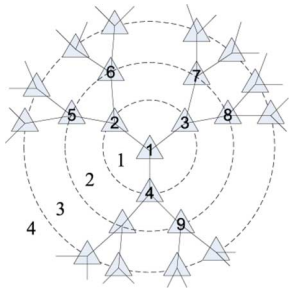

Fig. 2. Three-order Cayley tree network.

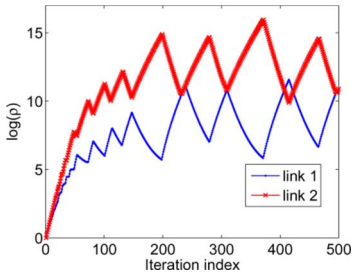

Fig. 3. Evolution of r_1 and r_2 in a 3×4 Cayley tree network for ACSMA of [6] with $T = 0.1$ s. Access intensities of other links exhibit similar fluctuations.

150 DATA packet times between successive measurements for convergence in ACSMA.

For networks that exhibit temporal starvation [8], even more time is needed for ACSMA within each iteration to smooth out the measurement. An example of a network that exhibits temporal starvation is Cayley tree network [19]. To illustrate our point, we perform simulations on a three-order four-layer Cayley tree. As shown in Fig. 2, each link in the Cayley tree has three neighbors. Emanating from link 1, all the links are arranged in shells around vertex 1. In our example, there are four such shells. We run ACSMA using the same parameters as in [6] except that the update interval T is set to 100 ms (we assume that DATA packet duration of 1 ms) and $\beta = 5$. Fig. 3 plots $r_i = \log(\rho_i)$, $i = 1, 2$ versus the iteration index, where link 2 is a neighbor of link 1. As can be seen, ACSMA cannot converge. This means that the update interval $T = 100$ ms is not long enough to accurately measure the link throughputs. Indeed, we find that ACSMA converges according to our convergence test $\max_j \left|r_j[n] - r_j^*\right| / r_j^* < 3\%$ only when T is set to a large value of 5700 ms. The number of iterations needed is 249. This means that ACSMA needs at least 5700×249 ms ≈ 23.66 min to converge.

Large update interval T is required to avoid oscillations of f_i and $\log(\rho_i)$ in the Cayley network because the temporal throughputs of links exhibit drastic fluctuations over time. Take link 1 as an example. As plotted in Fig. 4, its normalized temporal throughput, averaged over a window of $T = 100$ ms, alternates between 0 and 1 over time. To exactly measure the throughputs, each link needs to average its measured throughput over several 0–1 cycles, say 8–10 s. Thousands of DATA packet durations are needed in each iteration for accurate estimate of link throughputs under current network settings.

BP-ACSMA and GBP-ACSMA, however, do not require this real-time measurement and hence will not be affected by this

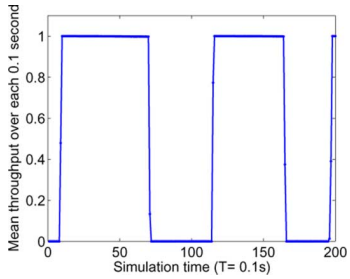

Fig. 4. Normalized throughputs of link 1 in a 3×4 Cayley tree measured over successive 0.1-s intervals when ACSMA is implemented. Throughputs of other links exhibit similar fluctuations.

temporal starvation phenomenon.[7] Indeed, for tree networks, distributed BP-ACSMA yields *exact* solutions with the number of iterations equal to the diameter of the graph [1]. The diameter of a 3×4 Cayley tree as in Fig. 2 is eight [19]. That is, distributed BP-ACSMA outputs exact solutions with eight iterations, corresponding to 0.8 s in real network operation, assuming beacons are used for message passing. Compared to ACSMA that converges after 23.66 min, BP-ACSMA is a lot faster.

Philosophical Interpretation of Convergence Rates: BP-ACSMA and GBP-ACSMA require one-hop message passing, while ACSMA does not require message passing. One may ponder why BP-ACSMA and GBP-ACSMA can converge faster than ACSMA. A way to look at the problem is as follows. In order for a link j to adjust its access intensity to achieve its "fair share" of throughput under the utility objective, it somehow has to acquire information about the network topology because "fair share" is a quantity that depends on the topology. To obtain the topology information that has an impact on the fair share, in a distributed algorithm, the links somehow have to communicate with each other.

In BP-ACSMA and GBP-ACSMA, the communication is in the form of "explicit" message passing. In ACSMA, however, the communication is achieved via "implicit" messages, in the following sense. In ACSMA, each time a link j transmits a regular data packet, it is actually conveying some information to the neighbor links. In particular, data packets transmitted by link j slow down the clearing of queues in neighbor links, who make use of their queue occupancies to adjust their access intensities. Because of the need for smoothing and the fact that these data packets are "indirect" messages, many more data packets than explicit messages are needed in order to convey the same information in ACSMA. This slows down the convergence rate of ACSMA.

The main potential drawback of BP-ACSMA and GBP-ACSMA is accuracy since the throughput dependencies on the access intensities are approximated. More precisely, both BP-ACSMA and GBP-ACSMA are only exact in tree topologies (e.g., Cayley tree networks) and may have errors in loopy graphs. The computation error may become unacceptable when the access intensities are extremely large (e.g., $1\ e+6$)

[7]We emphasize that we do not claim that BP-ACSMA and GBP-ACSMA can eliminate the temporal starvation phenomenon. Our point is that because BP-ACSMA and GBP-ACSMA obtain the equilibrium throughputs through computation, their convergence will not be slowed down by measurement. All the algorithms studied in this paper focus on controlling the equilibrium throughputs of links. However, given an acceptable equilibrium throughput, the temporal throughput of a link can still alternate between 0 and 1 in cycles of long durations. The reader is referred to [8] for a study on how to characterize temporal starvation and the possible remedies for it.

or the network is dense with large vertex degrees. In practice, we are unlikely to adopt very large access intensities because of implementation concerns such as finite size of time-slot (see of [22, Section III-B], where it was argued that access intensity cannot go beyond 530). Also, excessive access intensity is not desirable because the network may become susceptible to temporal starvation [8]. As for dense networks, we note that GBP-ACSMA can still achieve reasonably accurate results. Section VI details the theory behind GBP and how our specific implementation of GBP for CSMA networks attempts to remove small loops in the message passing construct; small loops are particularly detrimental to accuracy, as we have argued in Section III-F.

VI. Generalized Belief Propagation and Its Applications in CSMA Networks

In BP, all messages are from one vertex to another vertex. To reduce the error effects of loops, GBP allows messages to be passed from a group of vertices to another group. These groups of vertices are called *regions*, and the loops among the vertices in a region are subsumed into a region. The belief of a region corresponds to the joint probability of the states of the vertices within the region, and the intricate dependencies among the states of these vertices due to loops among them are captured by the joint probability. A region graph is constructed for message passing among the regions to capture the interdependencies among the regions.

A. Region Graph

The first step of GBP is to generate a region graph $\mathscr{G}$. In this paper, we use an algorithm similar to the cluster variation method introduced by Kikuchi and further developed in the physics literature [23]. GBP can be applied to various inference-making problems, and its general framework leaves open the issue of how to form regions out of the vertices. A contribution of this paper is to show that a "maximal clique" method of forming regions: 1) yields accurate results; and 2) is amenable to distributed implementation in CSMA networks.

A region $R = (V_R, E_R)$ is a subgraph of the original contention graph $G = (V, E)$ in which $V_R \subseteq V$, and $E_R \subseteq E$ are edges between the vertices in V_R. Regions are divided into different hierarchical levels. Each region belongs to one of the level. Fig. 5 gives an example demonstrating the construction of a region graph using the cluster variation method.

An important step is the forming of the set of regions at level 0, denoted by $\mathbf{R}_0$. Once $\mathbf{R}_0$ is formed, the regions at other levels are constructed based on $\mathbf{R}_0$. That is, the definitions of regions in other levels follow from $\mathbf{R}_0$.

Every vertex $i \in V$ and every edge $e \in E$ in the original graph $G = (V, E)$ must be included into at least one region $R \in \mathbf{R}_0$ [24]. A vertex can belong to more than one region in $\mathbf{R}_0$. However, no region $R \in \mathbf{R}_0$ could be a subregion of another region $R' \in \mathbf{R}_0$: that is, $R \not\subset R'$ for any two regions R, $R' \in \mathbf{R}_0$.

Within the above specification, there are many ways of forming $\mathbf{R}_0$. Different choices of $\mathbf{R}_0$ correspond to different implementations of GBP. Generally, there is a tradeoff between *complexity* and *accuracy*. Higher accuracy can be obtained if the regions in $\mathbf{R}_0$ are large, but the computation complexity will also be higher.

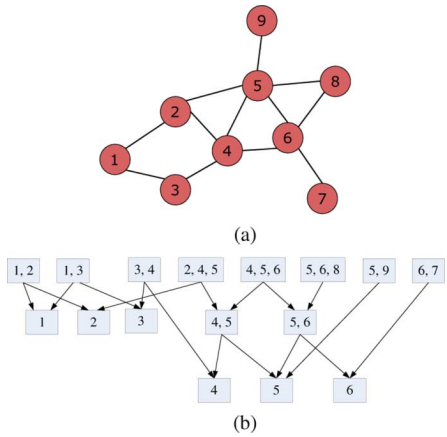

Fig. 5. Example of construction of a region graph. (a) Contention graph. (b) Region graph of (a).

As discussed in Section III-F, when BP messages are passed around a small loop, computation errors will be incurred. In GBP, we try to include loops in the original contention graph into a region in $\mathbf{R}_0$ to negate their effects. In our GBP for CSMA networks, we generate $\mathbf{R}_0$ by making each maximal clique in G a region in $\mathbf{R}_0$.[8] This ensures that each vertex and each edge in G are included into at least one region. The definition of $\mathbf{R}_0$ based on maximal cliques are motivated by two reasons. First, error-inducing small loops in BP consisting of only three vertices are guaranteed to be subsumed into a region. Although larger loops may not be subsumed into a region, the intuition is that they induce smaller errors anyway. Simulation results in the preceding sections have borne out our method of forming regions in $\mathbf{R}_0$ under various contention graphs and parameter settings. Second, as will be detailed later, maximal-clique $\mathbf{R}_0$ turns out to be a key that enables a distributed implementation of GBP in CSMA networks because it guarantees that the vertices of a region are one-hop neighbors of each other.

For notational simplicity, in the following, we sometimes write R in terms of its vertices without listing its edges. In Fig. 5, the maximal cliques are $\{1,2\}$, $\{1,3\}$, $\{3,4\}$, $\{2,4,5\}$, $\{4,5,6\}$, $\{5,6,8\}$, $\{5,9\}$, and $\{6,7\}$, all of which are included in $\mathbf{R}_0$ on the top row of Fig. 5(b).

After the construction of $\mathbf{R}_0$, we then construct the set of regions at level 1, $\mathbf{R}_1$, from the intersections of the regions in $\mathbf{R}_0$. We discard from $\mathbf{R}_1$ any intersection region that is a strict subregion of another intersection region. Specifically, to construct $\mathbf{R}_1$, we first form the set $\mathbf{S}_1 = \{R | R = R_i \cap R_j, \forall\ R_i \in \mathbf{R}_0, R_j \in \mathbf{R}_0, i \neq j\}$. We then discard from $\mathbf{S}_1$ any region $R \in \mathbf{S}_1$ where $R \subset R' \in \mathbf{S}_1$. In Fig. 5, for example, $\mathbf{R}_1$ consists of $\{1\}$, $\{2\}$, $\{3\}$, $\{4,5\}$, and $\{5,6\}$. Note that although $\{5\}$

[8]It is important to note that the identification of maximal cliques here is not NP-hard if the vertex degree is limited. In practical CSMA wireless networks, the degree of a vertex does not grow with the network size, thanks to geographical constraints. Typically, a vertex has at most 5–6 neighbors regardless of the number of vertices in the graph. Let K be the maximum degree of vertices in the contention graph and N be the number of links in the network. For each vertex the complexity of finding maximal cliques containing it is of order $O(2^K)$. Hence, the complexity of finding all the maximal cliques is of order $O(N2^K)$, which increases linearly with N. For distributed implementation, the computation-time complexity is of order $O(2^K)$.

is the intersection of $\{2,4,5\}$ and $\{5,6,8\}$, $\{5\}$ is not included in $\mathbf{R}_1$ because it is a strict subregion of $\{4,5\}$ and $\{5,6\}$.

Similarly, we construct the set of regions $\mathbf{R}_2$ from the intersections of the regions in $\mathbf{R}_0 \cup \mathbf{R}_1$. In addition to discarding intersection regions that are subregions of other intersection regions in $\mathbf{R}_2$, we also discard intersection regions that have already appeared in $\mathbf{R}_1$.

General Procedure for Constructing $\mathbf{R}_k$ and Edges to It: In general, to construct $\mathbf{R}_k$, we first form the set

$$\mathbf{S}_k = \bigcup_{n=0}^{k-1} \{R | R = R_i \cap R_j, \forall\, R_i \in \mathbf{R}_{k-1}, R_j \in \mathbf{R}_n, i \neq j\}.$$

We then discard from $\mathbf{S}_k$ any region $R \in \mathbf{S}_k$ where $R \subset R'$ for some $R' \in \mathbf{S}_k$; and any region $R \in \mathbf{S}_k$ where $R \in \mathbf{R}_n$ for some $n \leq k-1$ (i.e., also discard any region in $\mathbf{S}_k$ that already appears at an upper level). We stop forming new regions at the next level when no more new intersection regions can be identified.

For each region R, we draw a directed edge from each of its super-regions to it, except for those regions that are super-regions of other super-regions of region R. For example, in Fig. 5 there is no direct edge from $\{2,4,5\}$ to $\{4\}$ since region $\{2,4,5\}$ is the super-region of region $\{4,5\}$, which is also a super-region of $\{4\}$.

Observation 1: Since regions below $\mathbf{R}_0$ are formed from the intersections of regions in $\mathbf{R}_0$, and the regions in $\mathbf{R}_0$ are cliques, all regions in the overall regional graph must be cliques.

In the resulting region graph $\mathscr{G}$, an edge connects a "parent region" P and a "child region" R. If there is a directed path from region R' to region R, we say that R' is an ancestor of R, and R is a descendant of R'. We denote the region graph by $\mathscr{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of regions and $\mathbf{E}$ is the set of edges. Note that in this paper, to avoid confusion, the bold fonts $\mathbf{V}$ and $\mathbf{E}$ are used to refer to the "regions" and "edges between regions," and V and E refer to the "vertices" and "edges" in the original contention graph.

B. Belief Computation and Message-Update Rules

This paper adopts the Parent-to-Child algorithm [24] for message updates in GBP. In this algorithm, messages are passed from parent regions to their child regions only. Let $s_R = s_1 s_2 \cdots s_i s_j \cdots s_{|V_R|}$, $i, j \in V_R$, be the state of a region R. In GBP, the belief of R, $b_R(s_R)$, is to be estimated.

In GBP, the "intrinsic" belief of R is given by $\prod_{(i,j)\in E_R} \psi(s_i, s_j) \prod_{i \in V_R} \varphi_i(s_i)$. This would be proportional to the joint probability of the states of the vertices in R if there were no other vertices in the overall network (i.e., if R were the overall network itself). In general, $b_R(s_R)$ is found to be the product of the intrinsic belief and external messages from other regions, as explained in the next paragraph.

Let $\mathbf{D}_R \subseteq \mathscr{G}$ be the subgraph consisting of a region R and all its descendants. In GBP, the update equation for $b_R(s_R)$ has to incorporate all "external messages" passed to the regions in $\mathbf{D}_R$, not just those to R. In Fig. 5, for example, $\mathbf{D}_{\{5,6\}} = \{\{5,6\}, \{5\}, \{6\}\}$, and the following external messages are passed into $\mathbf{D}_{\{5,6\}}$: $m_{\{4,5,6\}\to\{5,6\}}$, $m_{\{5,6,8\}\to\{5,6\}}$, $m_{\{4,5\}\to\{5\}}$, $m_{\{5,9\}\to\{5\}}$, $m_{\{6,7\}\to\{6\}}$ (note that $m_{\{5,6\}\to\{5\}}$ and $m_{\{5,6\}\to\{6\}}$ are not external messages). In general, let $\mathrm{Parents}(R')$ denote the parents of a region R'. The belief at R is the product of its intrinsic belief and external messages:

$$b_R(s_R) \propto \prod_{(i,j)\in E_R} \psi(s_i, s_j) \prod_{i \in V_R} \varphi_i(s_i) \cdot \prod_{R' \in \mathbf{D}_R} \prod_{R'' \in \mathrm{Parents}(R') \setminus \mathbf{D}_R} m_{R''\to R'}(s_{R'}). \quad (18)$$

Note that in the above, s_R is a particular state of R, and the state of $R' \subseteq R$, $s_{R'}$, is induced from s_R.

The message-update rules are obtained by requiring consistency of the beliefs between parent and child regions. In Fig. 5(b), let us focus on the region $\{4,5,6\}$ and its child $\{5,6\}$. The belief at region $\{4,5,6\}$ is given by

$$b_{\{4,5,6\}}(s_4 s_5 s_6) \propto \prod_{i,j\in\{4,5,6\}, i\neq j} \psi(s_i, s_j) \prod_{i\in\{4,5,6\}} \varphi_i(s_i) \cdot m_{\{2,4,5\}\to\{4,5\}}(s_4 s_5) m_{\{5,6,8\}\to\{5,6\}}(s_5 s_6) \, m_{\{3,4\}\to\{4\}}(s_4) m_{\{5,9\}\to\{5\}}(s_5) m_{\{6,7\}\to\{6\}}(s_6)$$

and the belief at region $\{5,6\}$ is given by

$$b_{\{5,6\}}(s_5 s_6) \propto \psi(s_5, s_6)\varphi_5(s_5)\varphi_6(s_6) \cdot m_{\{4,5,6\}\to\{5,6\}}(s_5 s_6) m_{\{5,6,8\}\to\{5,6\}}(s_5 s_6) \, m_{\{4,5\}\to\{5\}}(s_5) m_{\{5,9\}\to\{5\}}(s_5) m_{\{6,7\}\to\{6\}}(s_6).$$

Using the marginalization constraint $b_{\{5,6\}}(s_5 s_6) = \sum_{s_4} b_{\{4,5,6\}}(s_4 s_5 s_6)$, we obtain a relation between messages

$$1 = \frac{\sum_{s_4} \psi(s_4, s_5)\psi(s_4, s_6)\varphi_4(s_4) m_{\{2,4,5\}\to\{4,5\}}(s_4 s_5) m_{\{3,4\}\to\{4\}}(s_4)}{m_{\{4,5,6\}\to\{5,6\}}(s_5 s_6) m_{\{4,5\}\to\{5\}}(s_5)}$$

from which we obtain

$$m_{\{4,5,6\}\to\{5,6\}}(s_5 s_6) = \frac{\sum_{s_4} \psi(s_4, s_5)\psi(s_4, s_6)\varphi_4(s_4) m_{\{2,4,5\}\to\{4,5\}}(s_4 s_5) m_{\{3,4\}\to\{4\}}(s_4)}{m_{\{4,5\}\to\{5\}}(s_5)}. \quad (19)$$

Note that on the right-hand side (RHS) of (19), with reference to Fig. 5, only those external messages flowing into $\mathbf{D}_{\{4,5,6\}}$ that are not also external messages flowing into $\mathbf{D}_{\{5,6\}}$ are retained in the numerator, and only the "internal" messages from $\mathbf{D}_{\{4,5,6\}} \setminus \mathbf{D}_{\{5,6\}}$ to $\mathbf{D}_{\{5,6\}}$ are retained in the denominator. Similar relations can be obtained between each pair of parent and child regions.

In general, the belief of a parent region P can be written as

$$b_P(s_P) \propto \prod_{(i,j)\in E_P} \psi(s_i, s_j) \prod_{i \in V_P} \varphi_i(s_i) \cdot \prod_{P' \in \mathbf{D}_P} \prod_{P'' \in \mathrm{Parents}(P') \setminus \mathbf{D}_P} m_{P''\to P'}(s_{P'}). \quad (20)$$

The marginalization constraint for a child region R with respect to the specific parent P is

$$b_R(s_R) = \sum_{s_{V_{P\setminus R}}} b_P(s_P). \quad (21)$$

Combining (18), (20), and (21), and canceling common items on the left-hand side (LHS) and RHS of (21), the message from a parent P to a child R can be written as (22), shown at the bottom of the next page.

Note that the term $\prod_{R' \in \mathbf{D}_P \setminus \mathbf{D}_R} \prod_{R'' \in \text{Parents}(R') \setminus \mathbf{D}_P} m_{R'' \to R'}(s_{R'})$ in the numerator consists of the "external" messages into $\mathbf{D}_P$ but not $\mathbf{D}_R$; the term $\prod_{R' \in \mathbf{D}_R} \prod_{R'' \in \text{Parents}(R') \cap \mathbf{D}_P \setminus \mathbf{D}_R} m_{R'' \to R'}(s_{R'})$ in the denominator consists of the "internal" messages from $\mathbf{D}_P \setminus \mathbf{D}_R$ to $\mathbf{D}_R$. Although not necessary mathematically, in each update, we also impose the normalization constraint $\sum_{s_R} m_{P \to R}(s_R) = 1$ to contain the numerical errors.

C. Distributed GBP

To implement GBP in a distributed manner, for each message from a parent region P to a child region R, $m_{P \to R}(s_R)$, we need to identify a particular vertex to be responsible for its update and dissemination. We will refer to the responsible vertex as the *message agent*.

We propose to let a vertex that is in both P and R, $j \in V_{P \cap R}$, to be the message agent for $m_{P \to R}(s_R)$. In general, $V_{P \cap R}$ could contain more than one vertex, in which case we elect the vertex with the lowest ID to be the message agent. As to what to use for ID, we note that each node in the CSMA network usually has a unique ID (e.g., MAC address). Each vertex is a link consisting of a transmitter and a receiver. We can simply choose the transmitter ID to be the link ID. If we have an infrastructure network, the AP ID can be the link ID.

Features for Correct Operation of Distributed GBP: The following lists three features of our distributed GBP that enable its correct operation.

Feature 1: Each vertex j could collect enough information to construct a local region graph $\mathscr{G}_j$ for the purpose of distributed computation of beliefs and messages. The local region graph $\mathscr{G}_j$ is a subgraph of the complete region graph $\mathscr{G}$. In particular, $\mathscr{G}_j$ is consistent with $\mathscr{G}$ in that each region appearing in $\mathscr{G}_j$ also appears in $\mathscr{G}$, and each edge appearing in $\mathscr{G}_j$ also appears in $\mathscr{G}$.

Feature 2: Each vertex j could: i) identify all regions to which it belongs from $\mathscr{G}_j$ and randomly select one of them, say R, for its throughput computation; ii) collect the information needed to compute the region belief $b_R(s_R)$ according to (18). Then, by taking marginal probability, it can compute its throughput: $th_j = b_j(s_j = 1) = \sum_{s_R : s_j = 1} b_R(s_R)$.

Feature 3: Each vertex j could: i) identify the messages for which it is the message agent from $\mathscr{G}_j$; and ii) for each such message $m_{P \to R}(s_R)$, collect the information needed to update $m_{P \to R}(s_R)$ according to (22).

Next, we describe the part of our distributed GBP that enables Feature 1: i.e., the construction of a consistent local region graph $\mathscr{G}_j$. In our implementation, a vertex j would first construct a local contention graph $G_j \subseteq G$, which it constructs based on the broadcast information from its neighbor vertices. From G_j, vertex j would then construct $\mathscr{G}_j$.

Assumption 1: We assume that a vertex j can hear the broadcasts of all its neighbors N_j in the contention graph G. The reader is referred to [16, Section III-D] on how to realize this assumption in real implementation.

Broadcast of Vertices to Enable Construction of G_j: Define $N_j^{(1)} \triangleq N_j \cup \{j\}$. Each vertex j in the network broadcasts three kinds of information in its neighborhood: 1) its link ID ID_j; 2) its access intensity ρ_j; 3) a local contention graph, denoted by $G_j^{(1)}$, consisting of all the vertices in $N_j^{(1)}$ and the edges between them (i.e., all edges (i,k) such that $i, k \in N_j^{(1)}$). Conceptually, this information is embodied in a 3-tuple $(\text{ID}_j, \rho_j, G_j^{(1)})$. For ease of exposition, we assume $\text{ID}_j = j$ and the broadcast information is a 3-tuple $(j, \rho_j, G_j^{(1)})$. Strictly speaking, the intensity ρ_j is not needed for the construction of local contention graphs and will be used only for the computations of beliefs and messages in Features 2 and 3. Thus, in the following, we focus on the 2-tuple $(j, G_j^{(1)})$ that can be extracted from the 3-tuple.

Construction of Local Contention Graph G_j: Consider a vertex j. Each neighbor $i \in N_j$ broadcasts the 2-tuple $(i, G_i^{(1)})$. Vertex j constructs a local region graph G_j based on $(i, G_i^{(1)})$ from all $i \in N_j$.

Initially $G_i^{(1)} = (i, \emptyset)$ and is not accurate. However, at least all $i \in N_j$ could be identified by vertex j after one round of broadcast by the neighbors. In the next round, each vertex $i \in N_j$, based on what it hears from its own neighbors in the last round, can deduce the set of edges $\{(i,k) | k \in N_i^{(1)}\}$. Vertex i will then broadcast $(i, G_i^{(1)})$ with $G_i^{(1)} = (N_i^{(1)}, \{(i,k) | k \in N_i^{(1)}\})$. Specifically, $G_i^{(1)}$ will have the correct vertices, but only edges between i and its neighbors appear; but not those between neighbors. After one more round, however, this will be fixed, and $G_i^{(1)} = (N_i^{(1)}, \{(k,l) | k,l \in N_i^{(1)}, (k,l) \in E\})$ where E are the edges in the complete contention graph $G = (V, E)$. Thus, three rounds of broadcast will make sure the broadcast 2-tuple is correct.

Then, vertex j constructs a local contention graph consisting of the union of its own $G_j^{(1)}$ and the $G_i^{(1)}$ in its neighborhood: $G_j = \bigcup_{i \in N_j^{(1)}} G_i^{(1)}$.

Property of G_j: G_j contains all vertices within two hops of vertex j. From G_j, vertex j can identify all the maximal cliques to which it belongs (within the overall contention graph G), as

$$m_{P \to R}(s_R) \propto \frac{\sum_{s_{V_P \setminus R}} \prod_{(i,j) \in E_P \setminus E_R} \psi(s_i, s_j) \prod_{i \in V_P \setminus V_R} \varphi_i(s_i) \prod_{R' \in \mathbf{D}_P \setminus \mathbf{D}_R} \prod_{R'' \in \text{Parents}(R') \setminus \mathbf{D}_P} m_{R'' \to R'}(s_{R'})}{\prod_{R' \in \mathbf{D}_R} \prod_{R'' \in \text{Parents}(R') \cap \mathbf{D}_P \setminus \mathbf{D}_R} m_{R'' \to R'}(s_{R'})}. \quad (22)$$

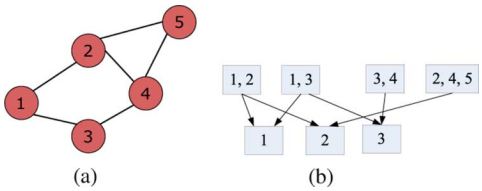

Fig. 6. Example of construction of a local region graph. (a) Local contention graph. (b) Local region graph.

well as all the maximal cliques to which each of its neighbors belongs. That is, all maximal cliques containing at least one vertex in $N_j^{(1)}$ can be identified from G_j.

Construction of Local Region Graph $\mathscr{G}_j$: Based on G_j, vertex j then constructs a local region graph $\mathscr{G}_j$ using the cluster variation method described in Section VI-A, with a small modification, as described in the next paragraph. As in Section VI-A, the first step is to form the set of regions at level 0, denoted by $\mathbf{R}_0^{(j)}$, from the maximal cliques in G_j that contains at least one vertex in $N_j^{(1)}$ (note that the Property of G_j above guarantees that $\mathbf{R}_0^{(j)}$ can be generated correctly from G_j). After the construction of $\mathbf{R}_0^{(j)}$, we then perform the same procedure as in Section VI-A to construct the regions in lower levels.

The modification is that we will discard all regions that do not contain any vertex in $N_j^{(1)}$(i.e., $V_R \cap N_j^{(1)} = \emptyset$). The discarded regions will have no bearing on the local computation to be performed to realize Features 2 and 3. In Fig. 5, for example, let us look at vertex 1. We draw the local contention graph G_1 in Fig. 6(a). By forming maximal cliques, $\mathbf{R}_0^{(1)}$ includes regions $\{1,2\}$, $\{1,3\}$, $\{3,4\}$ and $\{2,4,5\}$. At level 1, the intersections of regions in $\mathbf{R}_0^{(1)}$ are generated: $\{1\}$, $\{2\}$, $\{3\}$, and $\{4\}$. We discard region $\{4\}$ from level 1 since the sole vertex it contains, vertex 4, is two hops away from vertex 1 and not in $N_j^{(1)}$.

As in Section VI-A, for each remaining region R, we draw a directed edge from each of its super-regions to it, except for those regions that are super-regions of a super-region of region R. We denote the local region graph of vertex j by $\mathscr{G}_j = \left(\mathbf{V}^{(j)}, \mathbf{E}^{(j)}\right)$. Note that $\mathbf{V}^{(j)}$ here are regions and $\mathbf{E}^{(j)}$ are the directed edges between regions.

Consistency of Local Region Graph: We restate Feature 1 more rigorously here.

Feature 1: The local region graph $\mathscr{G}_j$ constructed from G_j is consistent with the complete region graph $\mathscr{G}$ in that each region in $\mathscr{G}_j$ is also a region in $\mathscr{G}$, and each edge in $\mathscr{G}_j$ is also an edge in $\mathscr{G}$. That is: 1)$\forall\, R \in \mathbf{V}^{(j)}$, $R \in \mathbf{V}$; 2) $\forall\, e \in \mathbf{E}^{(j)}$, $e \in \mathbf{E}$.

The rigorous proof of Feature 1 is given in [16, Appendix C]. The outline of the proof is as follows. We first show that each region in $\mathbf{R}_0^{(j)}$ must also be a region in $\mathbf{R}_0$. We then show that, in general, each region appearing in $\mathscr{G}_j$ must also appear in $\mathscr{G}$, and each edge between two regions in $\mathscr{G}_j$ must also be an edge in $\mathscr{G}$.

Feature 1 here means there are no extraneous regions or extraneous edges in $\mathscr{G}_j$. Feature 1′ below is sort of a converse to Feature 1. It states that the portion of the region graph $\mathscr{G}$ needed for the local computations of beliefs and messages by vertex j is exactly duplicated in $\mathscr{G}_j$. That is, $\mathscr{G}_j$ contains enough information to enable Features 2 and 3.

Feature 1′: Any region R in $\mathscr{G}$ that contains at least one vertex in $N_j^{(1)}$ must also be a region R in $\mathscr{G}_j$. Furthermore, consider two regions R and R', both of which have at least one vertex in $N_j^{(1)}$. If there is an edge between R and R' in $\mathscr{G}$, there is also an edge between R and R' in $\mathscr{G}_j$.

We refer interested readers to [16, Appendix C] for a rigorous proof of Feature 1′. Features 1 and 1′ above are critical to enabling Features 2 and 3 because Features 2 and 3 require the knowledge of all the regions in $\mathscr{G}$ that have at least one vertex in $N_j^{(1)}$, as well as the edges between them (this can be seen in the proofs of Features 2 and 3).

Based on $\mathscr{G}_j$, we proceed to implement the other procedures of our distributed GBP that enable Features 2 and 3.

Self-Identification of Message Agents, Message Computation, and Message Broadcast: Recall that in GBP a message is passed from a parent region P to a child region R. For each parent–child pair in $\mathscr{G}_j$, there is a message $m_{P\to R}(s_R)$ between them. Based on the local region graph $\mathscr{G}_j$, vertex j first identifies all the messages satisfying $j \in V_{P\cap R}$. For each such message, vertex j checks to see if it is the vertex with the lowest ID in $V_{P\cap R}$; if so, it will elect itself to be the message agent for $m_{P\to R}(s_R)$. This procedure enables Feature 3(i).

Vertex j will compute message $m_{P\to R}(s_R)$ for which it is responsible using (22), and then broadcast the message to its neighbors. Note from the RHS of(22) that, in general, other messages are required for the computation of $m_{P\to R}(s_R)$. Thus, vertex j must be able to collect these messages in our distributed implementation. The key observation is that in our maximal-clique $\mathbf{R}_0$ implementation, the message agents for these messages must be either vertex j itself or one-hop neighbors of vertex j. The reader is referred to [16, Appendix C] for a detailed proof for this property.

Belief Computation by Vertices: Each vertex j can choose a region R to which it belongs from $\mathscr{G}_j$ and computes the beliefs $b_R(s_R)$ according to (18). It then obtains its throughput by taking marginal probability $th_j = \sum_{s_R:s_j=1} b_R(s_R)$. Essentially, as with computation of messages, our proof of Feature 2 in [16, Appendix C] shows that vertex j will be able to hear the broadcast of the messages required in (18) by their message agents.

Periodic Update to Track Dynamic Network Topology: In practice, the network contention graph may change dynamically with new nodes joining and existing nodes leaving the network. Even among the existing nodes, they may become idle when their users are not actively using the network. To track the variations of the network topology, the local contention graph G_j needs to be refreshed periodically, as well as the local region graph $\mathscr{G}_j$.

The overall pseudocode of distributed GBP can be found in [16, Section VI-C].

D. IGBP and GBP-ACSMA

Analogous to IBP and BP-ACSMA, we can adapt GBP for the access intensities computation to meet the target throughput distribution and the resource allocation problem as in (10),

respectively. We refer the interested readers to [16] for more details.

VII. Conclusion

This paper is a first attempt to apply belief propagation to the analysis and design of CSMA wireless networks. In particular, we investigate three applications of belief propagation and generalized belief propagation: 1) computation of link throughputs given link access intensities; 2) computation of link access intensities required to meet target link throughputs; and 3) optimization of network utility.

We show how the BP and GBP algorithms for all three applications can be implemented in a distributed manner, making them useful in practical network operation. BP works well in terms of speed, and it yields exact results in networks with tree contention graphs. For loopy contention graphs, GBP improves accuracy at the cost of longer but still manageable convergence time.

With regard to 1) and 2), we use BP to solve two NP-hard problems in CSMA networks. In loopy graphs, BP yields solutions with accuracy higher than 90% under various contention-graph and access-intensity settings. GBP can cap the mean error to below 1% for networks of up to 200 links within dozens of iterations.

Among the three applications, of particular interest are distributed and adaptive algorithms to 3). A solution, ACSMA was first proposed in [6], in which no message passing is needed. The operation in ACSMA is based on "probe and measure." Specifically, before a link adjusts its access intensity in an iteration, a period of smoothing time is needed to measure the difference of its input and output traffic in the last iteration. As shown in this paper, the required smoothing time can be quite excessive in networks that exhibit temporal starvation [8], resulting in very slow convergence. BP-ACSMA and GBP-ACSMA do not have this problem because they are computation-based rather than measurement-based. One-hop message passing, however, is required.

BP has found empirical success in numerous applications (e.g., decoding of LDPC and turbo codes). Typically, the convergence of the BP algorithms in these applications is nontrivial to prove (except for tree graphs). Such is the case with belief propagation in CSMA networks as well. For all scenarios tested, our experiments indicate that both BP and GBP algorithms can converge quickly with accurate computed results. Convergence proofs, however, await future work.

References

[1] J. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Proc. IJCAI*, 2001.

[2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

[3] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[4] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int. J. Comput. Vision*, vol. 40, no. 1, pp. 25–47, 2000.

[5] M. Mezard, G. Parisi, and M. Virasor, *Spin Glass Theory and Beyond*. Singapore: World Scientific, 1987.

[6] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[7] S. Liew, C. Kai, J. Leung, and B. Wong, "Back-of-the-envelope computation of throughput distributions in CSMA wireless networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, pp. 1319–1331, Sep. 2010.

[8] C. Kai and S. Liew, "Temporal starvation in CSMA wireless networks," 2010 [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1009/1009.3415.pdf

[9] R. Boorstyn and A. Kershenbaum, "Throughput analysis in multihop CSMA packet radio networks," *IEEE Trans. Commun.*, vol. COM-35, no. 3, pp. 267–274, Mar. 1987.

[10] X. Wang and K. Kar, "Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks," in *Pro. IEEE INFOCOM*, 2005, vol. 1, pp. 23–34.

[11] M. Durvy, O. Dousse, and P. Thiran, "Border effects, fairness, and phase transition in large wireless networks," in *Proc. IEEE INFOCOM*, 2008, pp. 601–609.

[12] X. Shi, D. Schonfeld, and D. Tuninetti, "Message error analysis of loopy belief propagation for the sum-product algorithm," 2010 [Online]. Available: http://arxiv.org/PS_cache/arxiv/pdf/1009/1009.2305v1.pdf

[13] A. Ihler, J. Fisher, and A. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *J. Mach. Learn. Res.*, vol. 6, pp. 905–936, 2005.

[14] J. Mooij and H. Kappen, "Sufficient conditions for convergence of the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4422–4437, Dec. 2007.

[15] T. Heskes, "On the uniqueness of loopy belief propagation fixed points," *Neural Comput.*, vol. 16, no. 11, pp. 2379–2413, Nov. 2004.

[16] C. Kai and S. Liew, "Applications of belief propagation in CSMA wireless networks," 2010 [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1007/1007.5218.pdf

[17] *IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE 802.11-1997, 1997.

[18] L. Fu, S. Liew, and J. Huang, "Effective carrier sensing in CSMA networks under cumulative interference," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[19] R. Diestel, *Graph Theory*. Heidelberg, Germany: Springer-Verlag, 2010.

[20] C. Kai and S. Liew, "Throughput computation in CSMA wireless networks with collision effects," 2011 [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1107/1107.1633.pdf

[21] S. Liew, C. Chau, J. Zhang, and M. Chen, "Analysis of frequency-agile CSMA wireless networks," 2010 [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1007/1007.5255.pdf

[22] M. Chen, S. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[23] *Foundations and Applications of Cluster Variation Method and Path Probability Method*, T. Morita, M. Suzuki, K. Wada, and M. Kaburagi, Eds. Kyoto, Japan: PTP, 1994, vol. 115, Progress of Theoretical Physics: Supplement.

[24] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.

Cai Hong Kai received the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2010.

She is now an Associate Professor with the School of Computer and Information, Hefei University of Technology, Hefei, China. Her research interests are in IEEE 802.11-like multiaccess protocols and performance analysis of wireless networks.

Soung Chang Liew (S'87–M'88–SM'92–F'11) received the Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, in 1988.

From March 1988 to July 1993, he was with Bellcore (now Telcordia), Piscataway, NJ. He has been a Professor with the Department of Information Engineering, the Chinese University of Hong Kong, Hong Kong, since 1993. His research interests include wireless communications and networking, Internet protocols, multimedia communications, and packet switch design.

Prof. Liew is a Fellow of IET and HKIE.