Linköping Electronic Articles in
Computer and Information Science
Vol. n(2002): nr nn

Computing Kikuchi approximations by Cluster BP

Taisuke Sato

Department of Computer and Information Science
Tokyo Institute of Technology / CREST
Tokyo, Japan

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/2002/nnn/

Published on February nn,, 2002 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

© 2002 Taisuke Sato
Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. n(2002): nr nn.
http://www.ep.liu.se/ea/cis/2002/nnn/. *February nn,, 2002.*

This URL will also contain a link to the author's home page.

The publishers will keep this article on-line on the Internet (or its possible replacement network in the future) for a period of 25 years from the date of publication, barring exceptional circumstances as described separately.

The on-line availability of the article implies a permanent permission for anyone to read the article on-line, to print out single copies of it, and to use it unchanged for any non-commercial research and educational purpose, including making copies for classroom use. This permission can not be revoked by subsequent transfers of copyright. All other uses of the article are conditional on the consent of the copyright owner.

The publication of the article on the date stated above included also the production of a limited number of copies on paper, which were archived in Swedish university libraries like all other written works published in Sweden. The publisher has taken technical and administrative measures to assure that the on-line version of the article will be permanently accessible using the URL stated above, unchanged, and permanently equal to the archived printed copies at least until the expiration of the publication period.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/
or by conventional mail to the address stated above.

Abstract

In this paper, we investigate *loopy BP* (*belief propagation*) which is at the crossroads of Bayesian networks, statistical physics and error correction decoding. It is a method of computing exact and approximate marginals based on a minimization of variational free energy, and offers a quite powerful means for otherwise computationally intractable problems. We re-examine its theoretical background, the Kikuchi approximation, and propose *Cluster BP*, a mild generalization of loopy BP that can compute some class of Kikuchi approximations. We also propose *Cluster CCCP* which is a convergent version of Cluster BP to obtain local minima of Kikuchi approximations.

1 Introduction

In this paper, we investigate[1] *loopy BP* (*belief propagation*) which is at the crossroads of Bayesian networks, statistical physics and error correction decoding. It is a method of computing exact and approximate marginals based on a minimization of variational free energy, and offers a quite powerful means for otherwise computationally intractable problems.

Loopy BP came out of the study of Bayesian networks[2] [3, 2, 4]. Pearl proposed the BP (belief propagation) algorithm for computing marginal distributions [10]. It is a message passing algorithm that propagates messages (probability distributions) over a Bayesian network. When the network is singly connected, i.e. no loop appears when viewed as an undirected graph, it computes exact marginals in time linear in the size of the network [10].[3] As BP is just a local message passing algorithm, it is applicable even when the network contains loops (hence the name "loopy BP"), though at the risk of non-convergence. Historically while what loopy BP does was obscure in the case of loopy networks, experiments revealed that it is very versatile and effective, and sometimes can give remarkably good approximate marginals especially when used for error correction decoding [7, 8, 15]. It was not until recently that Yedidia et al. discovered that what loopy BP actually does is computing a stationary point of a certain quantity known as the Bethe approximation to the free energy in statistical physics [16]. The Bethe approximation is just one of possible approximations to the free energy. Yedidia et al. proposed *GBP* (generalized BP), a new message passing algorithm that yields, when it converges, a stationary point of the Kikuchi approximation which is a more complicated but more accurate than the Bethe approximation [16].[4] Soon later Yuille proposed an alternative to GBP, *CCCP double-loop algorithm*[5], that is guaranteed to compute local minima of the Bethe and Kikuchi approximations based on his new approach to optimizing functions which are decomposable into a sum of convex and concave functions [18].

While loopy BP is only able to compute stationary points of Bethe approximations, it is simple and efficient. Contrastingly GBP and CCCP for Kikuchi approximations are quite general but rather complex. In this paper, we exploit the simplicity of loopy BP and extend it to the *Cluster BP* algorithm by allowing messages to contain more than one variable. We clarify a condition which guarantees the computation of Kikuchi approximations. Our approach is a mild generalization of loopy BP along

[1] Our investigation is motivated by PRISM, a general symbolic-statistical modeling language which we have been developing for years [11, 13, 14]. It is based on a rigorous mathematical semantics and enables one to build a complex statistical model as a PRISM program and efficiently learn parameters embedded in the program through the graphical EM algorithm [12, 14]. The computational efficiency is achieved however by restricting programming style so that it complies with assumptions of probability computation employed by PRISM. We are seeking for a general yet robust method of probability computation to realize more freedom PRISM programming.

[2] A Bayesian network is a finite direct acyclic graph in which a conditional distribution associated with each vertex. It defines a joint distribution as a product of those conditional distributions.

[3] For Bayesian networks which are not singly connected, the junction tree algorithm [6] is available to efficiently compute marginal distributions.

[4] Descriptions of the Bethe and Kikuchi approximations in this paper are largely due to [16].

[5] "CCCP" stands for the Concave Convex Procedure.

the line of Aji and McEliece [1]. Furthermore, to remove the possibility of non-convergence, we derive the *Cluster CCCP* algorithm following Yuille's convex-concave decomposition which always converges and gives the same result as the Cluster BP algorithm.

In the sequel, we first briefly review the problem to be solved. We then introduce the Kikuchi approximation in a slightly generalized form by dropping the requirement that clusters used in the approximation must be closed under intersection. After preparations, two new algorithms, *Cluster BP* and *Cluster CCCP*, are proposed which run on cluster graphs that is a generalization of junction trees. They compute stationary points of (generalized) Kikuchi approximations but the latter is guaranteed to converge and compute the local minima. The reader is assumed to be familiar with Bayesian networks and the junction tree algorithm [3, 2, 4].

2 Preliminaries

2.1 Variable free energy

We assume a discrete random vector $X = (X_1, \ldots, X_n)$ of n dimension is given and has a joint distribution $p(x) = p(x_1, \ldots, x_n)$ such that

- $p(x)$ is always positive, $p(x) > 0$ for any x.
- $p(x)$ is presented as a product of functions called *potentials* $\psi_\alpha(x_\alpha)$

$$
\begin{aligned}
p(x) &= \kappa \prod_{\alpha \in \mathbf{P}} \psi_\alpha(x_\alpha) && (1)\\
&= \frac{1}{Z} e^{-E(x)} &&
\end{aligned}
$$

where

$$
\begin{aligned}
E(x) &\stackrel{\text{def}}{=} \sum_{\alpha \in \mathbf{P}} -\ln \psi_\alpha(x_\alpha) && (2)\\
Z &\stackrel{\text{def}}{=} \kappa^{-1} && (3)\\
&= \sum_x \prod_{\alpha \in \mathbf{P}} \psi_\alpha(x_\alpha) &&
\end{aligned}
$$

Here κ is a normalizing constant and Z is a partition function. $E(x)$ is considered as the "energy" of a state x. α denotes a sub-vector of $(1, \ldots, n)$ and $\mathbf{P}$ is a collection of such sub-vectors. x_α stands for a sub-vector of $(x_1, \ldots, x_n)$ corresponding to α. For instance, if $x = (x_1, x_2, x_3, x_4)$ and $\alpha = (2, 4)$, we have $x_\alpha = (x_2, x_4)$. For convenience, we identify the sub-vector α with a set consisting of its components. Therefore $\alpha = (2, 4)$ is considered $\alpha = \{2, 4\}$. Because both notations are mutually recoverable, they are equivalent and hence such treatment does not cause confusion.

A non-empty subset α of $\{1, \ldots, n\}$ is called a *cluster* and if α is in $\mathbf{P}$, it is referred to as a *potential cluster*. When α and β are clusters, so are $\alpha \cap \beta$, $\alpha \cup \beta$ and $\alpha \setminus \beta$. Operations on clusters reflect on vectors. For example, put $\alpha = \{1, 2, 3\}$ and $\beta = \{2, 3, 4\}$. Then $x_{\alpha \cap \beta} = x_{\{2,3\}} = (x_2, x_3)$ and $x_{\alpha \setminus \beta} = x_{\{1\}} = x_1$. $A \subset B$ means A is a proper subset of B while $A \subseteq B$ means $A \subset B$ or $A = B$.

We define the *variational free energy* $F(b)$ w.r.t. energy $E(x)$ by

$$
F(b) \stackrel{\text{def}}{=} (\text{average energy}) - (\text{entropy})
$$

$$= \sum_x b(x)E(x) + \sum_x b(x)\ln b(x) \tag{4}$$

where $b(\cdot)$ is a test distribution. Because $F(b) = \sum_x b(x)\ln\left(\frac{b(x)}{p(x)}\right) - \ln Z$ and the Kullback-Leibler divergence $\sum_x b(x)\ln\left(\frac{b(x)}{p(x)}\right)$ is non-negative and zero only when $b(\cdot) = p(\cdot)$, $F(b)$ takes a minimum value if and only if $b(\cdot) = p(\cdot)$. In other words, we have an equation $p = \mathrm{argmin}_b F(b)$.

Suppose a distribution $p(x)$ is given by (1). For the variational free energy $F(b)$, we can define its approximation, the *Kikuchi approximation* $F_K(\{b_\alpha\})$ such that $F(b) \approx F_K(\{b_\alpha\})$ with respect to $\{b_\alpha\}$.[6] Combining the Kikuchi approximation with $p = \mathrm{argmin}_b F(b)$, we obtain an approximation scheme for marginal distributions $\{p_\alpha(x_\alpha)\}$ of $p(x)$:

$$\{p_\alpha\} \approx \mathrm{argmin}_{\{b_\alpha\}} F_K(\{b_\alpha\}).$$

We read this equation from right to left and calculate approximate marginal distributions $\{p_\alpha\}$ by minimizing the functional $F_K(\{b_\alpha\})$ with $\{b_\alpha\}$ as variational variables. In view of the significance of computing marginals in many fields including pattern recognition, natural language processing, robotics, bioinformatics, coding theory etc on one hand and the difficulty of their exact computation on the other hand, it is quite important to develop efficient algorithms for computing $\mathrm{argmin}_{\{b_\alpha\}} F_K(\{b_\alpha\})$. The objective of this paper is to propose such algorithms while keeping the simplicity of BP.

2.2 The Kikuchi approximation

In this subsection, we introduce the Kikuchi approximation in a slightly generalized form. Let $\mathbf{P}$ be a set of potential clusters appearing in (1). For $\mathbf{P}$ we introduce another set $\mathbf{U}$ of clusters such that

- for any α in $\mathbf{P}$, there exists the smallest $\beta \in \mathbf{U}$ that includes α, i.e. $\beta \supseteq \alpha$ and if $\beta' \supseteq \alpha$ then $\beta' \supseteq \beta$ for any $\beta' \in \mathbf{U}$.[7]

$\mathbf{U}$ is called *a set of clusters for* $\mathbf{P}$. We consider $\mathbf{U}$ as a partially ordered set ordered by set inclusion ordering. Let $\mathbf{B}$ be the set of maximal clusters in $\mathbf{U}$. If $\mathbf{U}$ is generated from $\mathbf{B}$ by taking all possible non-empty intersections of elements in $\mathbf{B}$, i.e. $\mathbf{U} = \{\alpha_1 \cap \ldots \cap \alpha_h \mid \alpha_i \in \mathbf{B}, 0 < h, 1 \le i \le h\}$, a cluster in $\mathbf{U}$ is called a *Kikuchi cluster*, and $\mathbf{U}$ *a set of Kikuchi clusters for* $\mathbf{P}$. We associate an *overcounting number* a_α with a cluster α in $\mathbf{U}$ which is inductively defined from maximal clusters as follows.

$$\begin{cases} a_\alpha \stackrel{\mathrm{def}}{=} 1 & \text{if } \alpha \text{ is maximal in } \mathbf{U} \\ a_\alpha \stackrel{\mathrm{def}}{=} 1 - \sum_{\beta:\beta\supset\alpha,\beta\in\mathbf{U}} a_\beta & \text{o.w.} \end{cases}$$

By definition

$$\sum_{\alpha:\alpha\in\mathbf{U},\alpha\supseteq\beta} a_\alpha = 1 \text{ for } \forall\beta \in \mathbf{U}.$$

Introduce $E^+_\alpha(x_\alpha) \stackrel{\mathrm{def}}{=} -\ln\psi_\alpha(x_\alpha)$ and transform the average energy $\sum_x b(x)E(x)$ using the above property of overcounting numbers as follows.

[6] $b_\alpha(x_\alpha)$ (resp. $p_\alpha(x_\alpha)$) denotes a marginal distribution of $b(x)$ (resp. $p(x)$) marginalized to x_α.

[7] This condition is trivially satisfied by putting $\mathbf{U} = \mathbf{P}$, but for the sake of freedom of approximations, we prefer to introduce $\mathbf{U}$ independently of $\mathbf{P}$.

$$\begin{aligned}
\sum_x b(x)E(x) &= \sum_x b(x) \sum_{\beta:\beta\in\mathbf{P}} E^+_\beta(x_\beta) \\
&= \sum_{\beta:\beta\in\mathbf{P}} \sum_{x_\beta} b_\beta(x_\beta) E^+_\beta(x_\beta) \\
&= \sum_{\beta:\beta\in\mathbf{P}} \left(\sum_{\alpha\in\mathbf{U},\alpha\supseteq\beta} a_\alpha \right) \sum_{x_\beta} b_\beta(x_\beta) E^+_\beta(x_\beta) \\
&= \sum_{\alpha\in\mathbf{U}} a_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) E_\alpha(x_\alpha) \qquad (5)
\end{aligned}$$

Here $b_\beta(x_\beta) \stackrel{\text{def}}{=} \sum_{x_{[n]\setminus\beta}} b(x)$ ($[n] = \{1,\ldots,n\}$) is a marginal distribution and $E_\alpha(x_\alpha) \stackrel{\text{def}}{=} \sum_{\beta:\beta\in\mathbf{P},\beta\subseteq\alpha} E^+_\beta(x_\beta)$ is the energy of cluster α. Now the average energy is expressed as a linear combination of average cluster energies. It is unfortunate that a similar transformation of the entropy $S = -\sum_x b(x)\ln b(x)$ is impossible because of non-linearity of the entropy function. Yet we can derive its approximation as a linear combination of cluster entropies. First introduce S_α, the entropy of a cluster α by

$$S_\alpha \stackrel{\text{def}}{=} -\sum_{x_\alpha} b_\alpha(x_\alpha)\ln b_\alpha(x_\alpha).$$

Inductively define S^+_α from minimal elements of $\mathbf{U}$ by

$$\begin{cases} S^+_\alpha \stackrel{\text{def}}{=} S_\alpha & \text{if } \alpha \text{ is minimal in } \mathbf{U} \\ S^+_\alpha \stackrel{\text{def}}{=} S_\alpha - \sum_{\beta:\beta\subset\alpha,\beta\in\mathbf{U}} S^+_\beta & \text{o.w.} \end{cases}$$

For every $\alpha\in\mathbf{U}$, we have

$$S_\alpha = \sum_{\beta:\beta\subseteq\alpha,\beta\in\mathbf{U}} S^+_\beta.$$

Accordingly if we *assume* $S \approx \sum_{\beta\in\mathbf{U}} S^+_\beta$, we see

$$\begin{aligned}
S &\approx \sum_{\beta\in\mathbf{U}} S^+_\beta \\
&= \sum_{\beta\in\mathbf{U}} \left(\sum_{\alpha\in\mathbf{U},\alpha\supseteq\beta} a_\alpha \right) S^+_\beta \\
&= \sum_{\alpha\in\mathbf{U}} a_\alpha S_\alpha. \qquad (6)
\end{aligned}$$

Putting (5) and (6) together, the variational free energy $F(b)$ is approximated as follows.

$$\begin{aligned}
F(b) &= \sum_x b(x)E(x) - S \\
&\approx \sum_{\alpha\in\mathbf{U}} a_\alpha \left(\sum_{x_\alpha} b_\alpha(x_\alpha)E_\alpha(x_\alpha) + \sum_{x_\alpha} b_\alpha(x_\alpha)\ln b_\alpha(x_\alpha) \right).
\end{aligned}$$

We define the *generalized Kikuchi approximation* $\tilde{F}_K(\{b_\alpha\})$ for $F(b)$ by

$$\begin{aligned}
\tilde{F}_K(\{b_\alpha\}) &\stackrel{\text{def}}{=} \sum_{\alpha\in\mathbf{U}} a_\alpha \left(\sum_{x_\alpha} b_\alpha(x_\alpha) E_\alpha(x_\alpha) + \sum_{x_\alpha} b_\alpha(x_\alpha) \ln b_\alpha(x_\alpha) \right) \\
&= \sum_{\alpha\in\mathbf{U}} a_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln \left(\frac{b_\alpha(x_\alpha)}{\phi_\alpha(x_\alpha)} \right) \qquad (7)
\end{aligned}$$

where

$$\begin{aligned}
\phi_\alpha(x_\alpha) &\stackrel{\text{def}}{=} e^{-E_\alpha(x_\alpha)} \\
&= \exp\left(\sum_{\beta:\beta\in\mathbf{P},\beta\subseteq\alpha} -E^+_\beta(x_\beta) \right) \\
&= \prod_{\beta:\beta\in\mathbf{P},\beta\subseteq\alpha} \psi_\beta(x_\beta). \qquad (8)
\end{aligned}$$

When $\mathbf{U}$ is a set of Kikuchi clusters for $\mathbf{P}$, $\tilde{F}_K(\{b_\alpha\})$ is identical to the original Kikuchi approximation [16].

3 Cluster graphs

Just like Pearl's BP runs on Bayesian networks and the junction tree algorithm runs on junction trees, our new algorithms, *Cluster BP* and *Cluster CCCP*, will run on *cluster graph*s which can be seen as a generalization of junction trees to graphs or that of "junction graphs" [1] to ones where clusters other than $\mathbf{P}$ are used. From here on for brevity, we interchangeably use $n_1 \cdots n_h$ as cluster $\{n_1, \ldots, n_h\}$ like 123 for $\{1, 2, 3\}$.

Let $\mathbf{U}$ be a set of clusters for $\mathbf{P}$. An undirected graph $\mathcal{G}_U$ is said to be a *cluster graph on* $\mathbf{U}$ if vertices and edges are labeled by clusters in $\mathbf{U}$ as follows and if there is such a graph, we say $\mathbf{U}$ has a cluster graph.

- Every maximal element in $\mathbf{U}$ appears exactly once in $\mathcal{G}_U$ as a label of a vertex.
- Suppose vertices v_1 and v_2 are respectively labeled α_1 and α_2. The edge connecting v_1 and v_2, if any, is labeled $\beta \subseteq \alpha_1 \cap \alpha_2$ ($\beta \neq \emptyset$). When the equality $\beta = \alpha_1 \cap \alpha_2$ holds for every label on an edge, we say $\mathcal{G}_U$ is *regular*.
- The *tree condition* is satisfied: for every $\alpha \in \mathbf{U}$, a subgraph of $\mathcal{G}_U$ comprised of vertices and edges whose label include α is a tree.

Please note that we allow the same cluster to occur (as a label) more than once in a cluster graph. So vertices and edges may be labeled by a common cluster and so may be distinct edges. By the way it is obvious that junction trees are cluster graphs whose nodes are cliques and they satisfy the tree condition.

We have a look at examples of cluster graphs. The left graph in Figure 1 is a cluster graph on $\mathbf{U} = \{1234, 2356, 589, 1478, 168, 23, 14, 18, 5, 8, 1\}$. The maximal clusters are $\{1234, 2356, 589, 1478, 168\}$. Associated overcounting numbers are $a_{1234} = \cdots = a_{168} = 1$ for the maximal clusters, $a_{23} = a_5 = a_8 = a_{14} = a_{18} = -1$ and $a_1 = 0$ for the rest. We check the tree condition. For example the subgraph comprised of vertices and edges that include cluster 23 is a tree. Similarly, the one containing 1, though it does not

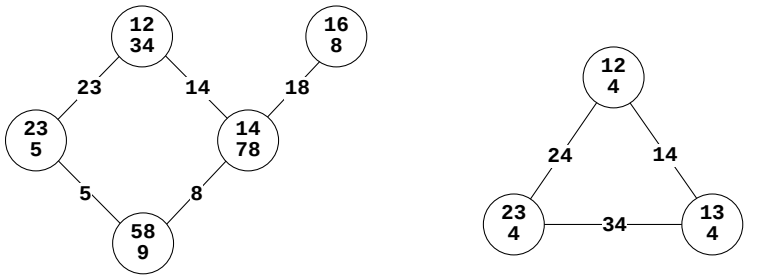

Figure 1: The left graph satisfies the tree condition but the right graph does not.

appear as a label, is also a tree and so on. So the left graph satisfies the tree condition. The right graph with $\mathbf{U} = \{124, 234, 134, 24, 34, 14, 4\}$ however does not satisfies the tree condition because all vertices and edges include 4 and they form a loop.

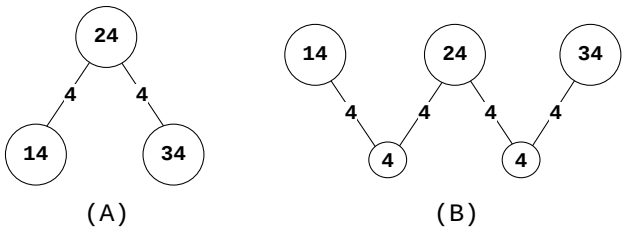

Figure 2: Cluster graphs with a multiple occurrence of the same label.

In Figure 2 both graphs in Figure 2 are on the same set of clusters $\mathbf{U} = \{14, 24, 34, 4\}$ and satisfy the tree condition.

We next prove a proposition that states the relationship between the tree condition and overcounting numbers.

Proposition 3.1
Let $\mathcal{G}_U$ be a cluster graph on $\mathbf{U}$ and a_α an overcounting number of $\alpha \in \mathbf{U}$. Also let $V(\alpha)$ and $E(\alpha)$ respectively be

$$V(\alpha) \stackrel{\text{def}}{=} \text{the number of vertices whose label is } \alpha$$
$$E(\alpha) \stackrel{\text{def}}{=} \text{the number of edges whose label is } \alpha$$

We have $a_\alpha = V(\alpha) - E(\alpha)$.

(Proof) By induction the size of clusters. Suppose α is a maximal cluster. Then there is exactly one vertex labeled α whereas there is no edge labeled α from the tree condition. So the r.h.s. $= 1 = a_\alpha$. Now suppose the proposition holds for all $\beta \supset \alpha$.

$$\begin{aligned} a_\alpha &= 1 - \sum_{\beta : \alpha \subset \beta \in \mathbf{U}} a_\beta \\ &= 1 - \sum_{\beta : \alpha \subset \beta \in \mathbf{U}} (V(\beta) - E(\beta)) \end{aligned}$$

$$= \left\{\begin{array}{l}\text{the number of edges}\\ \text{whose label properly}\\ \text{contain } \alpha\end{array}\right\} - \left\{\begin{array}{l}\text{the number of vertices}\\ \text{whose label properly}\\ \text{contain } \alpha\end{array}\right\} + 1$$

Recalling the following holds by the tree condition

$$V(\alpha) + \left\{\begin{array}{l}\text{the number of vertices}\\ \text{whose label properly}\\ \text{contain } \alpha\end{array}\right\} = E(\alpha) + \left\{\begin{array}{l}\text{the number of edges}\\ \text{whose label properly}\\ \text{contain } \alpha\end{array}\right\} + 1$$

$$= V(\alpha) - E(\alpha) \qquad Q.E.D.$$

Proposition 3.2
Suppose $\mathbf{U}$ *has a cluster graph.*
Then for every $\alpha \in \mathbf{U}$, $a_\alpha \leq 0$ *if* α *is not maximal in* $\mathbf{U}$.

(Proof) Let α be a non-maximal element in $\mathbf{U}$. There is a maximal $\beta \in \mathbf{U}$ such that $\alpha \subset \beta$. If no vertex bears α as a label, then $a_\alpha \leq 0$ by Proposition 3.1. Suppose otherwise and a vertex v is labeled by α. Due to the tree condition, there is a path connecting v and w where w is the vertex labeled by β. Let u, β' and γ respectively be a vertex on the path adjacent to v, its label and the label of the edge connecting u and v. We have $\alpha \subseteq \beta'$ and $\alpha \subseteq \gamma$ because all vertices and edges on the path have a label containing α. On the other hand, due to the labeling condition, $\gamma \subseteq \alpha \cap \beta' = \alpha$. Hence $\gamma = \alpha$. We may assume $\beta' \neq \alpha$ (o.w. we merge u and v and their labels β'). So there is a map from the vertex v labeled α and its edge labeled α connecting to an adjacent node whose label is not α. Consequently, from Proposition 3.1, we conclude $a_\alpha \leq 0$. Q.E.D.

We remark that "overcounting numbers of non-maximal clusters are not positive" is a necessary condition for the tree condition but not a sufficient condition.

4 Cluster BP

In this section, we derive an iterative algorithm, *Cluster BP*, running on cluster graphs which computes a stationary point of the generalized Kikuchi approximation by exchanging messages between vertices in a graph.

4.1 Deriving Cluster BP

We restate our assumptions and notational conventions.

- A joint distribution $p(x) = \kappa \prod_{\alpha \in \mathbf{P}} \psi_\alpha(x_\alpha)$ is specified by using potential clusters $\mathbf{P}$ and potentials $\psi_\alpha(x_\alpha)$. Potentials are always positive. We aim is to efficiently compute approximate marginals $\{p_\alpha(x_\alpha)\}$ by minimizing the generalized Kikuchi approximation $\tilde{F}_K(\{b_\alpha\})$ defined by (7).
- $\mathbf{U}$ is a set of clusters for $\mathbf{P}$. $\phi_\alpha(x_\alpha)$ is the α's potential defined as $\phi_\alpha(x_\alpha) \stackrel{\text{def}}{=} \prod_{\beta \in \mathbf{P}, \beta \subseteq \alpha} \psi_\alpha(x_\alpha)$.
- $\mathcal{G}_U$ is a cluster graph on $\mathbf{U}$. We denote by V (resp. E) the set of vertices (reps. edges) in $\mathcal{G}_U$, and by $\underline{v}$ (resp. by $\underline{e}$) the cluster labeling a vertex $v \in V$ (resp. the cluster labeling an edge $e \in E$).

Now back to the generalized Kikuchi approximation $\tilde{F}_K(\{b_\alpha\})$. We rewrite it using Proposition 3.1 as follows.

$$\begin{aligned}
\tilde{F}_K(\{b_\alpha\}) &= \sum_{\alpha\in\mathbf{U}} a_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln\left(\frac{b_\alpha(x_\alpha)}{\phi_\alpha(x_\alpha)}\right) \\
&= \sum_{\alpha\in\mathbf{U}} (V(\alpha)-E(\alpha)) \sum_{x_\alpha} b_\alpha(x_\alpha) \ln\left(\frac{b_\alpha(x_\alpha)}{\phi_\alpha(x_\alpha)}\right) \\
&= \sum_{\alpha\in\mathbf{U}} V(\alpha) \sum_{x_\alpha} b_\alpha(x_\alpha) \ln\left(\frac{b_\alpha(x_\alpha)}{\phi_\alpha(x_\alpha)}\right) \\
&\qquad - \sum_{\alpha\in\mathbf{U}} E(\alpha) \sum_{x_\alpha} b_\alpha(x_\alpha) \ln\left(\frac{b_\alpha(x_\alpha)}{\phi_\alpha(x_\alpha)}\right) \\
&= \sum_{v\in V} \sum_{x_{\underline{v}}} b_{\underline{v}}(x_{\underline{v}}) \ln\left(\frac{b_{\underline{v}}(x_{\underline{v}})}{\phi_{\underline{v}}(x_{\underline{v}})}\right) - \sum_{e\in E} \sum_{x_{\underline{e}}} b_{\underline{e}}(x_{\underline{e}}) \ln\left(\frac{b_{\underline{e}}(x_{\underline{e}})}{\phi_{\underline{e}}(x_{\underline{e}})}\right)
\end{aligned}$$

Note the last equation is expressed by marginal distributions $\{b_{\underline{v}}(x_{\underline{v}}), b_{\underline{e}}(x_{\underline{e}})\}$ that correspond to labels of vertices and edges in $\mathcal{G}_U$.

We minimize $\tilde{F}_K(\{b_\alpha\})$ as a functional over $\{b_{\underline{v}}(x_{\underline{v}}), b_{\underline{e}}(x_{\underline{e}})\}$ to obtain approximate marginals but the minimization must be carried out under two constraints.

Distribution: $\sum_{x_{\underline{v}}} b_{\underline{v}}(x_{\underline{v}}) = 1$ for every vertex v.

Consistency: $\sum_{x_{\underline{v}\setminus\underline{e}}} b_{\underline{v}}(x_{\underline{v}}) = b_{\underline{e}}(x_{\underline{e}})$ for every vertex v and its edge e.[8]

So we introduce a Lagrangian $\mathcal{L} = F_K(\{b_{\underline{v}}(x_{\underline{v}}), b_{\underline{e}}(x_{\underline{e}})\}) + \mathcal{C}$ with a constraint term $\mathcal{C}$.

$$\begin{aligned}
\mathcal{C} \stackrel{\text{def}}{=} \sum_{e\in E} \sum_{x_{\underline{e}}} \Bigg\{ & \lambda^{\underline{e}}_{\underline{v}_1}(x_{\underline{e}}) \left(\sum_{x_{\underline{v}_1\setminus\underline{e}}} b_{\underline{v}_1}(x_{\underline{v}_1}) - b_{\underline{e}}(x_{\underline{e}}) \right) \\
& + \lambda^{\underline{e}}_{\underline{v}_2}(x_{\underline{e}}) \left(\sum_{x_{\underline{v}_2\setminus\underline{e}}} b_{\underline{v}_2}(x_{\underline{v}_2}) - b_{\underline{e}}(x_{\underline{e}}) \right) \Bigg\} + \sum_{v\in V} \lambda_{\underline{v}} \left(\sum_{x_{\underline{v}}} b_{\underline{v}}(x_{\underline{v}}) - 1 \right)
\end{aligned}$$

One thing must be noted here. Setting the derivatives of $\mathcal{L}$ with respect to $\{b_{\underline{v}}(x_{\underline{v}}), b_{\underline{e}}(x_{\underline{e}})\}$ equal to zero does not give an algorithm which exchanges messages among vertices because $\mathcal{L}$'s variational variables correspond to labels of vertices, not to vertices themselves (some vertices may have a common label). We therefore inflate variational variables by introducing new variables $b_v(x_{\underline{v}}), b_e(x_{\underline{e}})$ which have one-to-one correspondence to vertices v and edges e.[9] However this change does not affect the solution because the tree condition ensures that those vertices or edges with a common label will have the same distribution. So our final Lagrangian becomes

[8] For any clusters β, α ($\beta \subset \alpha$), the tree condition guarantees the existence of a path connecting β and α regardless of whether they label a vertex or not. Hence, if the consistency holds along this path, $\sum_{x_{\alpha\setminus\beta}} b_\alpha(x_\alpha) = b_\beta(x_\beta)$ must hold transitively.

[9] If $v \neq v'$, $b_v(x_{\underline{v}})$ and $b_{v'}(x_{\underline{v}'})$ are independent variables. If they have a common label, i.e. $\underline{v} = \underline{v}'$ however, $b_v(x_{\underline{v}}) = b_{v'}(x_{\underline{v}'})$ must hold. The same applies to edges.

$$
\begin{aligned}
\mathcal{L}' &= \tilde{F}'_K + \mathcal{C}' \\
\tilde{F}'_K &= \sum_{v\in V}\sum_{x_{\underline{v}}} b_v(x_{\underline{v}}) \ln\left(\frac{b_v(x_{\underline{v}})}{\phi_{\underline{v}}(x_{\underline{v}})}\right) - \sum_{e\in E}\sum_{x_{\underline{e}}} b_e(x_{\underline{e}}) \ln\left(\frac{b_e(x_{\underline{e}})}{\phi_{\underline{e}}(x_{\underline{e}})}\right) \quad (9) \\
&= \sum_{v\in V}\sum_{x_{\underline{v}}} b_v(x_{\underline{v}}) \ln\left(\frac{b_v(x_{\underline{v}})}{\phi'_{\underline{v}}(x_{\underline{v}})}\right) - \sum_{e\in E}\sum_{x_{\underline{e}}} b_e(x_{\underline{e}}) \ln b_e(x_{\underline{e}}) \quad (10) \\
\mathcal{C}' &= \sum_{e\in E}\sum_{x_{\underline{e}}} \left\{ \lambda^e_{v_1}(x_{\underline{e}}) \left(\sum_{x_{\underline{v}1\setminus\underline{e}}} b_{v_1}(x_{\underline{v_1}}) - b_e(x_{\underline{e}}) \right) \right. \\
&\qquad \left. + \lambda^e_{v_2}(x_{\underline{e}}) \left(\sum_{x_{\underline{v}2\setminus\underline{e}}} b_{v_2}(x_{\underline{v_2}}) - b_e(x_{\underline{e}}) \right) \right\} \\
&\qquad + \sum_{v\in V} \lambda_v \left(\sum_{x_{\underline{v}}} b_v(x_{\underline{v}}) - 1 \right) \quad (11)
\end{aligned}
$$

Here $\lambda^e_{v_i}(x_{\underline{e}})$ $(i = 1, 2)$ is a Lagrange multiplier for the vertex v_i connected by an edge e. It is introduced to ensure the consistency of marginal distributions associated with v_i and e. In the transition from (9) to (10), we performed *an adjustment of potentials* to displace potentials from edges, which is performed nondeterministically as follows (there is a lot of freedom).

1. For each edge e, choose a vertex v from those connected by e. We write $v = \theta(e)$.
2. Let $\phi_{uv}(x_{\underline{v}})$ be the potential of v and $\phi_{\underline{e}}(x_{\underline{e}})$ the potential of e. Define the *adjusted potential for* v by
$$\phi'_v(x_{\underline{v}}) = \frac{\phi_{\underline{v}}(x_{\underline{v}})}{\prod_{e:\theta(e)=v} \phi_{\underline{e}}(x_{\underline{e}})}$$
We say $\phi_{\underline{e}}(x_{\underline{e}})$ *is moved to* v.

We show examples. In Figure 3, suppose the potential of edge 23 is moved to vertex 234, that of edge 4 to vertex 145, that of edge 1 to vertex 123 and that of edge 14 to vertex 145. The adjusted potentials are respectively $\phi'_{123}(x_{123}) = \frac{\phi_{123}(x_{123})}{\phi_1(x_1)}$, $\phi'_{234}(x_{234}) = \frac{\phi_{234}(x_{234})}{\phi_{23}(x_{23})}$, $\phi'_{145}(x_{145}) = \frac{\phi_{145}(x_{145})}{\phi_4(x_4)\phi_{14}(x_{14})}$ and $\phi'_{14}(x_{14}) = \phi_{14}(x_{14})$.

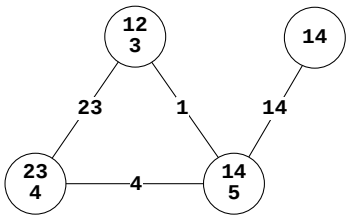

Figure 3: An adjustment of potentials

Now we calculate the derivatives of $\mathcal{L}'$ with respect to $\{b_v(x_{\underline{v}}), b_e(x_{\underline{e}})\}$ and put them equal to 0. We here change notations and use i, j, k for vertices, (ij) for the edge connecting i and j and $\underline{ij}$ for the cluster labeling (ij). The results are

$$
\begin{aligned}
b_i(x_{\underline{i}}) &= \kappa\, \phi'_i(x_{\underline{i}}) \prod_{k \in N(i)} e^{-\lambda_k^{(ki)}(x_{\underline{ki}})} \\
b_{(ij)}(x_{\underline{ij}}) &= \kappa'\, e^{-\lambda_i^{(ij)}(x_{\underline{ij}})} e^{-\lambda_j^{(ij)}(x_{\underline{ij}})}
\end{aligned}
$$

where $N(i)$ is a set of the vertices adjacent to i and κ, κ' are normalizing constants. Applying the distribution and consistency conditions to the pairs of $b_i(x_{\underline{i}})$ and $b_{(ij)}(x_{\underline{ij}})$ and introducing

$$
m_{i \triangleright j}(x_{\underline{ij}}) \stackrel{\text{def}}{=} \kappa\, e^{-\lambda_i^{(ij)}(x_{\underline{ij}})}
$$

we reach the *Cluster BP algorithm* in Figure 4 that computes stationary points of generalized Kikuchi approximations. A multi-variate function $m_{i \triangleright j}(x_{\underline{ij}})$ is considered as a message from vertex i to vertex j. $\phi'_i(x_{\underline{i}})$ is an adjusted potential associated with i. $N(i) \setminus j$ denotes the set $N(i)$ with the vertex j deleted.

$$
m_{i \triangleright j}(x_{\underline{ij}}) = \kappa \sum_{x_{\underline{i} \setminus \underline{ij}}} \phi'_i(x_{\underline{i}}) \prod_{k \in N(i) \setminus j} m_{k \triangleright i}(x_{\underline{ki}}) \tag{12}
$$

$$
b_i(x_{\underline{i}}) = \kappa'\, \phi'_i(x_{\underline{i}}) \prod_{k \in N(i)} m_{k \triangleright i}(x_{\underline{ki}}) \tag{13}
$$

Figure 4: Cluster BP

This algorithm starts with initial messages $\{m_{i \triangleright j}(x_{\underline{ij}}) = 1\}$ and iteratively updates $\{m_{i \triangleright j}(x_{\underline{ij}})\}$ by substituting the right hand side of (12) for the left hand side infinitely (sufficiently) many times. After all $\{m_{i \triangleright j}(x_{\underline{ij}})\}$ converge, i.e. they reach a fixed point satisfying (12), we substitute them for $\{m_{k \triangleright i}(x_{\underline{ki}})\}$ in (13) and obtain approximate marginals $\{b_i(x_{\underline{i}})\}$ of $p(x)$.

4.2 Special cases

As cluster graphs are rather arbitrarily defined, we should not expect accurate approximations without appropriate restriction on $\mathbf{U}$. Proposition 4.1 is obvious from the derivation of Cluster BP.

Proposition 4.1
Suppose $\mathbf{U}$ *is a set of Kikuchi clusters and there is a cluster graph* $\mathcal{G}_U$ *on* $\mathbf{U}$. *Cluster BP running on* $\mathcal{G}_U$ *computes a stationary point of the Kikuchi approximation.*

We now look into cases where $\mathcal{G}_U$ is a tree. In this case observe that $b_i(x_{\underline{i}})$ is uniquely determined by (12) and by (13) because starting from (13) and replacing the l.h.s. $m_{i \triangleright j}(x_{\underline{ij}})$ of (12) with the r.h.s., we obtain, in time linear in the size of the tree, $b_i(x_{\underline{i}})$ expressed purely in terms of the sum and product of adjusted potentials.

Proposition 4.2
Suppose $\mathbf{U}$ *is closed under intersection*[10] *and has a cluster graph* $\mathcal{G}_U$ *on* $\mathbf{U}$ *which is a tree. Then marginal distributions computed by Cluster BP using* $\mathcal{G}_U$ *are exact.*

(Proof) First of all $\mathcal{G}_U$ satisfies the so-called junction tree property.[11] This is because if z is in some cluster $\alpha \in \mathbf{U}$, there is the smallest cluster $\beta \in \mathbf{U}$ containing z as $\mathbf{U}$ is closed under intersection. The tree condition ensures that vertices and edges whose label include β form a tree. As a result the junction tree property is satisfied.

Second, $\mathcal{G}_U$ must be regular because otherwise there are vertices labeled α and β respectively, and connected by an edge e whose label is γ such that $\gamma \subset \alpha \cap \beta$. As $\alpha \cap \beta \in \mathbf{U}$, there is a path comprised of vertices and edges whose label include $\alpha \cap \beta$. The path does not contain the edge e however, which means that there is a loop containing γ, contradicting the tree condition.

Thirdly, we can deduce using $\sum_{\alpha:\alpha\in\mathbf{U},\alpha\supseteq\beta} a_\alpha = 1$ that

$$\begin{aligned}
\prod_{\alpha\in\mathbf{P}} \psi_\alpha(x_\alpha) &= \prod_{\alpha\in\mathbf{U}} (\phi_\alpha(x_\alpha))^{a_\alpha} \\
&= \prod_{\alpha\in\mathbf{U}} \frac{\phi_\alpha(x_\alpha)^{V(\alpha)}}{\phi_\alpha(x_\alpha)^{E(\alpha)}} \\
&= \frac{\prod_{v\in V} \phi_{\underline{v}}(x_{\underline{v}})}{\prod_{e\in E} \phi_{\underline{e}}(x_{\underline{e}})} \\
&= \prod_{v\in V} \phi'_{\underline{v}}(x_{\underline{v}})
\end{aligned}$$

where $\phi_\alpha(x_\alpha)$ is a potential for the cluster α, $\phi'_v(x_{\underline{v}})$ an adjusted potential for v, and V is the set of vertices in $\mathcal{G}_U$.

From $\prod_{\alpha\in\mathbf{P}} \psi_\alpha(x_\alpha) = \prod_{v\in V} \phi'_{\underline{v}}(x_{\underline{v}})$ and the fact that $\mathcal{G}_U$ is a regular cluster tree, it is straightforward to prove that marginal distributions $\{b_i(x_{\underline{i}})\}$ defined by (12) and (13) are exact (details omitted). Q.E.D.

We state some remarks below. When $\mathcal{G}_U$ is a junction tree (considering separators as clusters labeling edges), Cluster BP coincides with the celebrated junction tree algorithm [6]. Proposition 4.2 covers however also cases where some clusters are subsets of others thereby $\mathcal{G}_U$ being not a junction tree.

Aji and McEliece introduced junction graphs and the Bethe-Kikuchi approximation in [1]. They derived an iterative message passing algorithm called the generalized distribution law which is identical to Cluster BP for computing Bethe-Kikuchi approximations [1]. However the nature of the Bethe-Kikuchi approximation was unclear and under what condition the generalized distribution law algorithm computes Kikuchi approximations has been left open.

In junction graphs vertices are labeled only by potential clusters from $\mathbf{P}$ and it is required that for any variable z (singleton cluster), edges and

[10] If $\alpha, \beta \in \mathbf{U}$, then $\alpha \cap \beta \in \mathbf{U}$ if $\alpha \cap \beta \neq$. Note that this condition is less restrictive than the condition that $\mathbf{U}$ is a set of Kikuchi clusters. For example $\mathbf{U} = \{12, 1, 2\}$ is not a set of Kikuchi clusters but closed under intersection.

[11] If a variable z occurs in the labels of vertices v and v', there is a path connecting v and v' such that every vertex and edge on the path contains z.

vertices whose label contains z form a tree. Cluster graphs are similar to junction graphs but allow clusters other than $\mathbf{P}$ as labels of vertices and we further require that the tree condition holds for all Kikuchi clusters as well as for singleton clusters. The Bethe-Kikuchi approximation therefore is considered as a generalized Kikuchi approximation where $\mathbf{U}$ includes all singletons $\{1, \ldots, n\}$ and has a cluster graph $\mathcal{G}_U$ whose vertices are labeled only by potential clusters, but the tree condition is dropped for clusters other than singletons $\{1, \ldots, n\}$.

Lastly, suppose $\mathbf{U}$ consists of potential clusters and singletons, i.e. $\mathbf{U} = \mathbf{P} \cup \{1, \ldots, n\}$. If $\mathcal{G}_U$ is a bipartite graph with $\mathbf{P}$ on one side and $\{1, \ldots, n\}$ on the other side, connecting k in $\{1, \ldots, n\}$ with $\alpha \in \mathbf{P}$ containing k, Cluster BP is reduced to the sum-product algorithm [5].

5 Cluster CCCP

Loopy BP (resp. GBP) can compute stationary points of Bethe approximations (resp. Kikuchi approximations) but does not necessarily converge. Yuille recently proposed a new approach to the sure minimization of Bethe and Kikuchi approximations based on the convex-concave decomposition of a target function [18]. His algorithm is called the *CCCP double-loop algorithm* and computes local minima of Bethe and Kikuchi approximations.

The basic idea is as follows. Suppose a multi-variate function $f(x)$ is represented as a sum of a convex function and a concave function. For convenience however, we regard this sum, *convex-concave decomposition*, as a difference of convex functions $g(x)$ and $h(x)$ like $f(x) = g(x) - h(x)$ (note that $-h(x)$ is concave). Then due to the property of convex functions, we have $(1 - r)g(x) + rg(x + d) \geq g(x + rd)$ for any real vector d and any real number r $(0 < r < 1)$. By taking the limit $r \to 0$, we know $g(x + d) - g(x) \geq (d \cdot \bigtriangledown g(x))$ where $\cdot$ denotes an inner product. Hence we have

$$\begin{aligned} g(x_n) &\geq g(x_{n+1}) + ((x_n - x_{n+1}) \cdot \bigtriangledown g(x_{n+1})) \\ h(x_{n+1}) &\geq h(x_n) + ((x_{n+1} - x_n) \cdot \bigtriangledown h(x_n)) \end{aligned}$$

from which we obtain a recurrence formula generating a series $\{x_n\}$ that locally minimizes $f(x)$.

$$\begin{aligned} f(x_n) = g(x_n) - h(x_n) \;\geq\; f(x_{n+1}) = g(x_{n+1}) - h(x_{n+1}) \\ \text{if } \bigtriangledown g(x_{n+1}) = \bigtriangledown h(x_n) \end{aligned}$$

When the problem is a minimization of $f(x)$ under linear constrains $(a_i \cdot x) = b_i$ $(1 \leq i \leq h)$, a similar treatment is possible and we can derive a series that leads to a solution.

$$\begin{aligned} &f(x_n) \geq f(x_{n+1}) \text{ if} \\ &\qquad \bigtriangledown g(x_{n+1}) = \bigtriangledown h(x_n) + \textstyle\sum_{i=1}^{h} c_i a_i \text{ for some } c_i \; (1 \leq i \leq h) \end{aligned}$$

Figure 5: Minimizing $f(x)$ under linear constraints

Since Bethe approximations (resp. Kikuchi approximations) are representable as a difference of two convex functions (of the form $x \ln \left(\frac{x}{a}\right)$ and $x \ln x$), Yuille applied the above method and derived an iterative algorithm

that is guaranteed to converge and guaranteed to compute local minima of Bethe approximations (resp. Kikuchi approximations) [18].

We here follow [18] and apply his decomposition method to minimize the generalized Kikuchi approximation $\tilde{F}_K(\{b_\alpha\})$ in (7) or equivalently $\tilde{F}'_K$ in (10) and derive a new algorithm called *Cluster CCCP*. We however carefully choose a decomposition of $\tilde{F}_K(\{b_\alpha\})$ so that the resulting Cluster CCCP becomes a convergent generalization of Cluster BP.

We assume as before a distribution $p(x) = \kappa \prod_{\alpha \in \mathbf{P}} \psi_\alpha(x_\alpha)$ and a cluster graph $\mathcal{G}_U$ on a set of clusters $\mathbf{U}$ for $\mathbf{P}$ and use V for the set of vertices and E for the set of edges in $\mathcal{G}_U$. When $i \in V$ is a vertex, $\underline{i}$ is a cluster labeling i. Also when $(ij) \in E$ is an edge connecting i and j, $\underline{ij}$ is a cluster labeling the edge (ij). $\phi'_i(x_{\underline{i}})$ is an adjusted potential of i. We rewrite F'_K in (10) as follows.

$$
\begin{aligned}
F'_K &= \sum_{i \in V} \sum_{x_{\underline{i}}} b_i(x_{\underline{i}}) \ln \left(\frac{b_i(x_{\underline{i}})}{\phi'_i(x_{\underline{i}})} \right) - \sum_{(ij) \in E} \sum_{x_{\underline{ij}}} b_{(ij)}(x_{\underline{ij}}) \ln b_{(ij)}(x_{\underline{ij}}) \quad (14) \\
&= \left\{ \sum_{i \in V} \sum_{x_{\underline{i}}} b_i(x_{\underline{i}}) \ln \left(\frac{b_i(x_{\underline{i}})}{\phi'_i(x_{\underline{i}})} \right) + \sum_{(ij) \in E} \sum_{x_{\underline{ij}}} b_{(ij)}(x_{\underline{ij}}) \ln b_{(ij)}(x_{\underline{ij}}) \right\} \\
&\quad -2 \sum_{(ij) \in E} \sum_{x_{\underline{ij}}} b_{(ij)}(x_{\underline{ij}}) \ln b_{(ij)}(x_{\underline{ij}}) \quad (15)
\end{aligned}
$$

We apply the convex-concave decomposition to (15) and derive the Cluster CCCP algorithm that locally minimizes the generalized Kikuchi approximation. The result is displayed in Figure 6.[12]

$$
\begin{aligned}
e^{2\lambda^j_{(ij)}(x_{\underline{ij}})} &= \frac{\sum_{x_{\underline{i}\backslash\underline{j}}} \phi'_i(x_{\underline{i}})\, e^{-\sum_{h \in N(i)\backslash j} \lambda^h_{(ih)}(x_{\underline{ih}})}}{\{b_{(ij)}(x_{\underline{ij}})\}^2\, e^{\lambda^i_{(ij)}(x_{\underline{ij}})}} \left(\frac{e^{-(1+\lambda_i)}}{e} \right) \quad (16) \\
e^{2\lambda^i_{(ij)}(x_{\underline{ij}})} &= \frac{\sum_{x_{\underline{j}\backslash\underline{i}}} \phi'_j(x_{\underline{j}})\, e^{-\sum_{h \in N(j)\backslash i} \lambda^h_{(jh)}(x_{\underline{jh}})}}{\{b_{(ij)}(x_{\underline{ij}})\}^2\, e^{\lambda^j_{(ij)}(x_{\underline{ij}})}} \left(\frac{e^{-(1+\lambda_j)}}{e} \right) \quad (17) \\
e^{(1+\lambda_i)} &= \sum_{x_{\underline{i}}} \phi'_i(x_{\underline{i}}) e^{-\sum_{h \in N(i)} \lambda^h_{(ih)}(x_{\underline{ih}})} \quad (18)
\end{aligned}
$$

(Inner loop)

$$
b'_{(ij)}(x_{\underline{ij}}) = \{b_{(ij)}(x_{\underline{ij}})\}^2\, e^{\lambda^i_{(ij)}(x_{\underline{ij}}) + \lambda^j_{(ij)}(x_{\underline{ij}}) + 1} \quad (19)
$$

(Outer loop)

Figure 6: Cluster CCCP

Cluster CCCP is executed as follows. First we initialize $\{b_i(x_{\underline{i}}), \lambda_i, \lambda^j_{(ij)}(x_{\underline{ij}})\}$ to 1 and sort edges $\{(ij)\}$. We then iterate the inner loop (to satisfy linear constraints). Take an edge (ij) serially and for each (ij) update

[12] It is possible to apply the convex-concave decomposition to (14) but the resulting algorithm won't be as elegant as the Cluster CCCP.

$\{\lambda^j_{(ij)}(x_{\underline{ij}}), \lambda^i_{(ij)}(x_{\underline{ij}}), \lambda_i\}$ using (16), (17) and (18) *in this order*. When updates for all $\{(ij)\}$ are finished, we enter next cycle of the inner loop. We repeat the inner loop until all values of $\{\lambda_i, \lambda^j_{(ij)}(x_{\underline{ij}})\}$ converge (the convergence is assured). After a the convergence of the inner loop, we execute the outer loop (19) once to update $b_{(ij)}(x_{\underline{ij}})$ to $b'_{(ij)}(x_{\underline{ij}})$. At each update of the outer loop, the variable free energy decreases.

We iterate this inner loop–outer loop process enough times so that the outer loop converges. After the convergence of the outer loop (again the convergence is assured), we calculate

$$b'_i(x_{\underline{i}}) \quad = \quad e^{-(1+\lambda_i)} \phi'_i(x_{\underline{i}}) \, e^{-\sum_{h \in N(i)} \lambda^h_{(ih)}(x_{(ih)})} \tag{20}$$

to obtain approximate marginals $b'_i(x_{\underline{i}})$ for vertex i.

We check the relationship between Cluster CCCP and Cluster BP. When the outer loop converges, we have $b'_{(ij)}(x_{\underline{ij}}) = b_{(ij)}(x_{\underline{ij}})$, from which it follows

$$b_{(ij)}(x_{\underline{ij}}) = e^{-\lambda^i_{(ij)}(x_{\underline{ij}}) - \lambda^j_{(ij)}(x_{\underline{ij}}) - 1}.$$

By substituting this into (16) and (17), and replacing $e^{-\lambda^i_{(ij)}(x_{\underline{ij}})}$ with $m_{i \triangleright j}(x_{\underline{ij}})$, we reproduce Cluster BP in (4). One may say therefore that Cluster CCCP is a convergent version of Cluster BP which takes advantage of information about neighbor points of a stationary point.

6 Related work and discussion

One of the purposes of this paper is to clarify and generalize the junction graph approach proposed by Aji and McEliece [1]. Our graphs, cluster graphs, are similar to junction graphs, but more restrictive in that all clusters are required to satisfy the tree condition though more general in that clusters are not limited to potential clusters. We proved that Cluster BP, which coincides with the generalized distribution law algorithm as an algorithm, computes Kikuchi approximations using cluster graphs on a set of Kikuchi clusters. We further proposed Cluster CCCP as a convergent generalization of Cluster BP though it is not in a style of message passing.

(Generalized) Kikuchi approximations which Cluster BP and Cluster CCCP can compute are however a proper subclass of those computed by a more general method such as GBP [16]. By Proposition (3.2) Cluster BP does not work for Kikuchi approximations which have positive overcounting numbers for nonmaximal elements. For instance look at the right graph (1) in Figure 7. It is on a set of Kikuchi clusters $\mathbf{U} = \{124, 234, 134, 24, 34, 14, 4\}$ which correspond to a distribution $p(x_{1234}) = \kappa \, \psi(x_{124})\psi(x_{234})\psi(x_{134})$, and the overcounting number of the minimum element, 4, is 1. So Cluster BP is not applicable to the Kikuchi approximation for the $p(x_{1234})$. Nevertheless a reasonable approximation by Cluster BP is possible by transforming the graph so that the tree condition is satisfied like the one illustrated in Figure 7.[13]

[13] What we lose compared to the Kikuchi approximation is the consistency of marginals between b_{234} and b_{134}. We cannot expect $\sum_{x_2} b_{234}(x_{234}) = \sum_{x_1} b_{134}(x_{134})$ because the clusters 234 and 134 have no path conveying information about the correlation of x_3 and x_4.

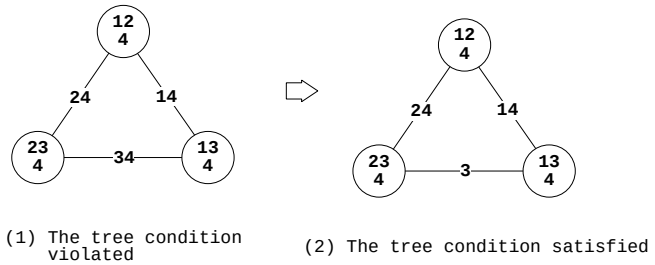

Figure 7: Turning into a cluster graph

In this case we have a generalized Kikuchi approximation on a set of clusters $\mathbf{U} = \{124, 234, 134, 24, 14, 3\}$ which is computable by the sum and product over six messages using Cluster BP. GBP on the other hand can compute a more accurate approximation, the Kikuchi approximation, but requires the sum, product and division over nine one-way messages some of which are listed below.

$$
\begin{aligned}
m_{124\to 14}(x_{14}) &= \kappa_1 \frac{\sum_{x_2} \psi(x_{124}) m_{234\to 24}(x_{24})}{m_{24\to 4}(x_4)} \\
m_{234\to 24}(x_{24}) &= \kappa_2 \frac{\sum_{x_3} \psi(x_{234}) m_{134\to 34}(x_{34})}{m_{34\to 4}(x_4)} \\
m_{134\to 34}(x_{34}) &= \kappa_3 \frac{\sum_{x_1} \psi(x_{134}) m_{124\to 14}(x_{14})}{m_{14\to 4}(x_4)} \\
&\dots
\end{aligned}
$$

As indicated by the above example, it seems that there is a trade-off between the complexity of the algorithm and the accuracy of the approximation. We leave it as a future work to investigate how to balance the computational efforts against the approximation accuracy.[14]

Pakzad and Anantharam generalized the Kikuchi approximation by allowing Kikuchi clusters to be just a partially ordered set represented by a Hasse diagram [9]. They proposed to vary the degree of approximation by a number k in such a way that for very cluster $|\alpha| \leq k$, $\sum_{\beta:\beta\subseteq\alpha} a_\beta = 1$.

7 Conclusion

We have proposed cluster graphs to compute generalized Kikuchi approximations by Cluster BP using multi-variate messages. It has been shown that Cluster BP computes when converged a subclass of Kikuchi approximations for which cluster graphs exist. We also have proposed Cluster CCCP which is more complicated than Cluster BP but is assured to compute local minima of generalized Kikuchi approximations.

[14] In a more recent publication Yedidia et al. proposed the region graph method [17]. Region graphs are a generalization of junction graphs with additional structures, a partial ordering over regions and a distinction between variable nodes and factor nodes like factor graphs for the sum-product algorithm [5]. In the Appendix E, they derive a two-way message passing algorithm for computing a stationary point of the region free energy, and suggest a sufficient condition for the two-way algorithm to be reduced to the standard BP, which seems equivalent to the tree condition though there is a subtlety due to the fact that cluster graphs admit of a multiple occurrence of the same label.

References

[1] A.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Trans. Inform. Theory*, 46(2):325–343, 2000.

[2] E. Castillo, J. M. Gutierrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer-Verlag, 1997.

[3] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996.

[4] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[5] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transaction on Information Theory*, 47(2):498–519, 2001.

[6] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society, B*, 50:157–224, 1988.

[7] D.J.C. MacKay, R.J. McEliece, and J.-F. Cheng. Turbo decoding as an instance of pearl's "belief propagation" algorithm. *IEEE Journal of Selected Areas of Communication*, pages 140–152, 1998.

[8] Kevin Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference (UAI-1999)*, pages 467–475, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[9] P. Pakzad and V. Anantharam. Belief Propagation and Statistical Physics. In *Proceedings of 2002 Conference on Information Sciences and Systems*, 2002.

[10] J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:357–369, 1986.

[11] T. Sato. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 715–729, 1995.

[12] T. Sato, S. Abe, Y. Kameya, and K. Shirai. A separate-and-learn approach to EM learning of PCFGs. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLRPS2001)*, pages 255–262, 2001.

[13] T. Sato and Y. Kameya. PRISM: a language for symbolic-statistical modeling. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1330–1335, 1997.

[14] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[15] Y. Weiss and W.T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

[16] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. Technical Report MERL TR2001-16, Mitsubishi Electric Research Laboratories, 2001.

[17] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report MERL TR2002-35, Mitsubishi Electric Research Laboratories, 2002.

[18] A.L. Yuille. CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. In *NIPS2001*, pages –, 2001.