Belief Propagation for Min-Cost Network Flow: Convergence & Correctness

Citation

As Published
http://dl.acm.org/citation.cfm?id=1873625

Publisher
Society for Industrial and Applied Mathematics

Version
Final published version

Accessed
Wed Mar 16 08:24:20 EDT 2016

Citable Link
http://hdl.handle.net/1721.1/73521

Terms of Use
Article is made available in accordance with the publisher’s policy and may be subject to US copyright law. Please refer to the publisher’s site for terms of use.

Detailed Terms

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
Belief Propagation for Min-cost Network Flow: Convergence & Correctness

David Gamarnik∗ Devavrat Shah† Yehua Wei ‡

Abstract

We formulate a Belief Propagation (BP) algorithm in the context of the capacitated minimum-cost network flow problem (MCF). Unlike most of the instances of BP studied in the past, the messages of BP in the context of this problem are piecewise-linear functions. We prove that BP converges to the optimal solution in pseudo-polynomial time, provided that the optimal solution is unique and the problem input is integral. Moreover, we present a simple modification of the BP algorithm which gives a fully polynomial-time randomized approximation scheme (FPRAS) for MCF. This is the first instance where BP is proved to have fully-polynomial running time.

1 Introduction

The Markov random field or graphical model provides a succinct representation for capturing the dependency structure between a collection of random variables. In the recent years, the need for large scale statistical inference has made graphical models the representation of choice in many statistical applications. There are two key inference problems for a graphical model of interest. The first problem is the computation of marginal distribution of a random variable. This problem is (computationally) equivalent to the computation of partition function in statistical physics or counting the number of combinatorial objects (e.g., independent sets) in computer science. The second problem is that of finding the mode of the distribution, i.e., the assignment with maximum likelihood (ML). For a constrained optimization problem, when the constraints are modeled through a graphical model and probability is proportional to the cost of the assignment, an ML assignment is the same as an optimal solution. Both of these questions, in general, are computationally hard.

Belief Propagation (BP) is an “umbrella” heuristic designed for these two problems. Its version for the first problem is known as the “sum-product algorithm” and for the second problem is known as the “max-product algorithm”. Both versions of the BP algorithm are iterative, simple and message-passing in nature. In this paper, our interest lies in the correctness and convergence properties of the max-product version of BP when applied to the minimum-cost network flow problems (MCF), an important class of linear (or more generally convex) optimization problems. In the rest of the paper, we will use the term BP to refer to the max-product version unless specified otherwise.

The BP algorithm is essentially an approximation of the dynamic programming assuming that underlying graphical model is a tree [8], [22], [16]. Specifically, assuming that the graphical model is a tree, one can obtain a natural parallel iterative version of the dynamic programming in which variable nodes pass messages between each other along edges of the graphical model. Somewhat surprisingly, this seemingly naive BP heuristic has become quite popular in practice [2], [9], [11], [17]. In our opinion, there are two primary reasons for the popularity of BP. First, like dynamic programming, it is generically applicable, easy to understand and implementation-friendly due to its iterative, simple and message-passing nature. Second, in many practical scenarios, the performance of BP is surprisingly good [21] [22]. On one hand, for an optimist, this unexpected success of BP provides a hope for its being a genuinely much more powerful algorithm than what we know thus far (e.g., better than primal-dual methods). On the other hand, a skeptic would demand a systematic understanding of the limitations (and strengths) of BP, in order to caution a practitioner in using it. Thus, irrespective of the perspective of an algorithmic theorist, rigorous understanding of BP is very important.

Prior work. Despite these compelling reasons, only recently have we witnessed an explosion of research for theoretical understanding of the performance of the BP algorithm in the context of various combinatorial optimization problems, both tractable and intractable (NP-hard) versions. To begin with, Bayati, Shah and Sharma [6] considered the performance of BP for the problem of finding maximum weight matching in a bipartite graph. They established that BP converges in pseudo-polynomial time to the optimal solution when the optimal solution is unique [6]. Bayati et al. [4] as well as Sanghavi et al. [18] generalized this result...
by establishing correctness and convergence of the BP algorithm for b-matching problem when the linear programming (LP) relaxation corresponding to the node constraints has a unique integral optimal solution. Note that the LP relaxation corresponding to the node constraints is not tight in general, as inclusion of the odd-cycle elimination constraints [20] is essential. Furthermore, [4] and [18] established that the BP does not converge if this LP relaxation does have a non-integral solution. Thus, for a b-matching problem BP finds an optimal answer when the LP relaxation can find an optimal solution. In the context of maximum weight independent set problem, a one-sided relation between LP relaxation and BP is established [19]; if BP converges then it is correct and LP relaxation is tight. We also take note of the work on BP for Steiner tree problem [5], special instances of quadratic/convex optimization problems [13], [12], Gaussian graphical models [10], and the relation between primal and message-passing for LP decoding [3].

2 BP for Linear Programming Problem

Suppose we are given an LP problem in the standard form:

\[(LP) \quad \begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0, x \in \mathbb{R}^n
\end{align*}\]

where \(A\) is a real \(m \times n\) matrix, \(b\) is a real vector of dimension \(m\), and \(c\) is a real vector of dimension \(n\). We define its factor graph, \(F_{LP}\), to be a bipartite graph, with factor nodes \(v_1, v_2, \ldots, v_m\) and variable nodes \\(e_1, e_2, \ldots, e_m\), such that \(e_i\) and \(v_j\) are adjacent if and only if \(a_{ij} \neq 0\). For example, the graph shown in Figure 1, is the factor graph of the LP problem:

\[
\begin{align*}
\min & \quad 4x_1 + x_2 - 3x_4 + x_5 + 2x_6 - x_7 \\
\text{s.t.} & \quad x_1 + 9x_3 - x_4 + 5x_7 = 10 \\
& \quad -2x_1 + x_2 + x_3 - 3x_4 + x_5 - x_6 + x_7 = 3 \\
& \quad x_1 + x_6 = 2 \\
& \quad x \geq 0, x \in \mathbb{R}^7.
\end{align*}
\]

In \(F_{LP}\), for every \(1 \leq i \leq m\), let \(E_i = \{e_j | a_{ij} \neq 0\}\); and let \(V_j = \{v_i | a_{ij} \neq 0\}\). For each factor node \(v_i\), define the factor function \(\psi_i : \mathbb{R}^{|E_i|} \rightarrow \mathbb{R}\) (\(\mathbb{R} = \mathbb{R} \cup \infty\)) as:

\[
\psi_i(z) = \begin{cases} 
0 & \text{if } \sum_{j \in E_i} a_{ij} z_j = b_j \\
\infty & \text{otherwise}
\end{cases}
\]

Essentially, \(\psi_i\) represents the \(i\)-th equality constraints in \(LP\). If we let \(x_{E_i}\) be the subvector of \(x\) with indices in \(E_i\), then \(\psi_i(x_{E_i}) < \infty\) if and only if \(x\) does not violate the \(i\)-th equality constraint in \(LP\). Also, for any variable node \(e_j\), define variable function \(\phi_j : \mathbb{R} \rightarrow \mathbb{R}\) as:

\[
\phi_j(z) = \begin{cases} 
c_j z & \text{if } \sum_{j \in E_i} a_{ij} z_j = b_j \\
\infty & \text{if } \sum_{j \in E_i} a_{ij} z_j = b_j
\end{cases}
\]

Notice that problem \(LP\) is equivalent to \(\min_{x \in \mathbb{R}^n} \sum_{i=1}^m \psi_i(x_{E_i}) + \sum_{j=1}^n \phi_j(x_j)\), which is defined on \(F_{LP}\). As BP is defined on graphical models, we can now formulate the BP algorithm as an iterative message-passing algorithm on \(F_{LP}\). It is formally presented below as Algorithm 1. The idea of the algorithm is that at the \(t\)-th iteration, every factor vertex \(v_i\) sends a message function \(m_{v_i \rightarrow e_j}(z)\) to each one of its neighbors \(e_j\), where \(m_{v_i \rightarrow e_j}(z)\) is \(v_i\)’s estimate of the cost if \(x_j\) takes value \(z\), based on the messages \(v_i\) received at time \(t\) from its neighbors. Additionally, every variable vertex \(e_j\) sends a message function \(m_{e_j \rightarrow v_i}(z)\) to each one of its neighbors \(v_i\), where \(m_{e_j \rightarrow v_i}(z)\) is \(e_j\)’s estimate of the cost if \(x_j\) takes value \(z\), based on the messages \(e_j\) received at time \(t-1\) from all of its neighbors except \(v_j\). After \(N\) iterations, the algorithm calculates the “belief” at variable node \(e_j\) using the messages sent to vertex \(e_j\) at time \(N\) for each \(j\).

In Algorithm 1, one basic question is the computation procedure of message functions of the form \(m_{v_i \rightarrow e_j}(z)\) and \(m_{e_j \rightarrow v_i}(z)\). We claim that every message function is a convex piecewise-linear function, which allows us to encode it in terms of a finite vector describing the break points and slopes of its linear pieces. In subsection 3.1, we describe in detail how \(m_{v_i \rightarrow e_j}(z)\) can be computed.

In general, the max-product behaves badly when solving an LP instance without a unique optimal solution [6], [4]. Yet, even with the assumption that an
Algorithm 1 BP for LP

1: Initialize $t = 0$, and for each variable node $e_j$, create factor to variable message $m^{t}_{v_i \rightarrow e_j}(z) = 0$, $\forall e_j \in E_i$.
2: for $t = 1, 2, \ldots, N$ do
3: \hspace{1em} $\forall e_j, v_i$, where $v_i \in V_j$, set $m^{t}_{e_j \rightarrow v_i}(z)$ to
4: \hspace{3em} $\phi_j(z) + \sum_{v_r \in V_j \setminus v_i} m^{t-1}_{v_r \rightarrow e_j}(z)$,
5: \hspace{1em} $\forall z \in \mathbb{R}$.
6: \hspace{1em} $\forall e_j, v_i$, where $e_j \in E_i$, set $m^{t}_{v_i \rightarrow e_j}(z)$ to
7: \hspace{3em} $\min_{z \in \mathbb{R}^{E_i \cup \bar{V}_j}, z_j = z} \left\{ \psi_i(z) + \sum_{e_j \in E_i \setminus e_j} m^{t}_{e_j \rightarrow v_i}(z_j) \right\}$,
8: \hspace{1em} $\forall z \in \mathbb{R}$.
9: \hspace{1em} $t := t + 1$
10: end for

$\hat{x}^N$ in Algorithm 1 does not converge to the unique optimal solution as $N$ grows large. One such instance (LP-relaxation of an instance of the maximum-weight independent set problem) which was earlier described in [19]:

$$\min \sum_{i=1}^{3} 2x_i - \sum_{j=1}^{3} 3y_j$$

s.t. $x_i + y_j + z_{ij} = 1, \forall 1 \leq i \leq 3, 1 \leq j \leq 3$

$x, y, z \geq 0$

3 BP Algorithm for Min-Cost Network Flow Problem

We start off this section by defining the capacitated min-cost network flow problem ($\mathcal{MCF}$). Given a directed graph $G = (V, E)$, let $V, E$ denote the set of vertices and arcs respectively, and let $|V| = n$, $|E| = m$. For any vertex $v \in V$, let $E_v$ be the set of arcs incident to $v$, and for any $e \in E_v$, let $\Delta(e) = 1$ if $e$ is an out-arc of $v$ (i.e. arc $e = (v, w)$, for some $w \in V$), and $\Delta(e) = -1$ if $e$ is an in-arc of $v$ (i.e. arc $e = (w, v)$, for some $w \in V$). $\mathcal{MCF}$ on $G$ is formulated as follows [7],[1]:

$$\min \sum_{e \in E} c_ew, \quad (\mathcal{MCF})$$

$$\Delta(v, e)x_e = b_v, \forall v \in V \quad \text{(demand constraints)}$$

$$0 \leq x_e \leq u_e, \forall e \in E \quad \text{(flow constraints)}$$

where $c_e \geq 0$, $u_e \geq 0$, $c_e \in \mathbb{R}$, $u_e \in \mathbb{R}$, for each $e \in E$, and $b_v \in \mathbb{R}$ for each $v \in V$. $\mathcal{MCF}$ is a special class of LP, we can view each arc $e \in E$ as a variable vertex and each $v \in V$ as a factor vertex, and define functions $\psi_v$, $\phi_e$, $\forall v \in V, \forall e \in E$:

$$\psi_v(z) = \begin{cases} 0 & \text{if } \sum_{e \in E_v} \Delta(v, e)z_e = b_v \\ \infty & \text{otherwise} \end{cases}$$

$$\phi_e(z) = \begin{cases} c_ez & \text{if } 0 \leq z \leq u_e \\ \infty & \text{otherwise} \end{cases}$$

Algorithm 2 BP for Network Flow

1: Initialize $t = 0$. For each $e \in E$, suppose $e = (v, w)$.
2: for $t = 1, 2, 3, \ldots, N$ do
3: \hspace{1em} $\forall e \in E$, let $e = (v, w)$, update messages:
4: \hspace{3em} $m^t_{e \rightarrow v}(z) = \phi_e(z)$
5: \hspace{3em} $+ \min_{z \in \mathbb{R}^{E_w \cup \bar{V}_v}, z_v = z} \left\{ \psi_w(\bar{z}) + \sum_{\bar{e} \in E_w \setminus e} m^{t-1}_{\bar{e} \rightarrow w}(\bar{z}_{\bar{e}}) \right\}$,
6: \hspace{3em} $\forall z \in \mathbb{R}$
7: \hspace{3em} $m^t_{e \rightarrow w}(z) = \phi_e(z)$
8: \hspace{3em} $+ \min_{z \in \mathbb{R}^{E_v \cup \bar{V}_w}, z_w = z} \left\{ \psi_v(\bar{z}) + \sum_{\bar{e} \in E_v \setminus e} m^{t-1}_{\bar{e} \rightarrow v}(\bar{z}_{\bar{e}}) \right\}$,
9: \hspace{3em} $\forall z \in \mathbb{R}$.
10: \hspace{1em} $t := t + 1$
11: end for

12: $\forall e \in E$, let $e = (v, w)$, and set the “belief” function:
13: \hspace{1em} $n^N_e(z) = m^N_{e \rightarrow v}(z) + m^N_{e \rightarrow w}(z) - \phi_e(z)$
14: \hspace{1em} $n^N_e(z)$ is the guess of the optimal solution of $\mathcal{MCF}$.
Clearly, solving $\text{MCF}$ is equivalent to solving\[ \min_{x \in \mathbb{R}^{|E|}} \{ \sum_{v \in V} \psi_v(x_{E_v}) + \sum_{e \in E} \phi_e(x_e) \}. \]We can then apply BP algorithm for LP (Algorithm 1) for $\text{MCF}$. Because of the special structure of $\text{MCF}$, each variable node is adjacent to exactly two factor nodes. This allows us to skip the step of update messages $m^e_{v \rightarrow c}$, and present a simplified version of BP algorithm for $\text{MCF}$, which we refer to as Algorithm 2.

To understand Algorithm 2 at a high level, each arc can be thought of as an agent, which is trying to figure out its own flow while meeting the conservation constraints at its endpoints. Each link maintains an estimate of its “local cost” as a function of its flow (thus this estimate is a function, not a single number). At each time step an arc updates its function as follows: the cost of assigning $x$ units of flow to link $e$ is the cost of pushing $x$ units of flow through $e$ plus the minimum-cost way of assigning flow to neighboring edges (w.r.t. the functions they computed last time step) to restore flow conservation at the endpoints of $e$.

Before formally stating theorem of convergence of BP for $\text{MCF}$, we first give the definition of a residual network \([1]\). Define $G(x)$ to be the residual network of $G$ and flow $x$ as: $G(x)$ has the same vertex set as $G$, and $\forall e = (v, w) \in E$, if $x_e < u_e$, then $e$ is an arc in $G(x)$ with cost $c^e_x = c^e$, also, if $x_e > 0$, then there is an arc $e' = (w, v)$ in $G(x)$ with cost $c^{e'}_x = -c^e$, now, let $\delta(x) = \min_{C \in \mathcal{C}} (c^e(C) = \sum_{e \in C} c^e_x)$, where $\mathcal{C}$ is the set of directed cycles in $G(x)$. Note that if $x^*$ is a unique optimal solution of $\text{MCF}$ with directed graph $G$, then $\delta(x^*) > 0$ in $G(x^*)$.

**Theorem 3.1.** Suppose $\text{MCF}$ has a unique optimal solution $x^*$. Define $L$ to be the maximum cost of a simple directed path in $G(x^*)$. Then for any $N \geq (\frac{L}{2\delta(x^*)} + 1)n$, we have $\bar{x}^N = x^*$.

Thus, by Theorem 3.1, the BP algorithm finds the unique optimal solution of $\text{MCF}$ in at most $(\frac{L}{2\delta(x^*)} + 1)n$ iterations.

### 3.1 Computing/Encoding Message Functions

First, we formally define a convex piecewise-linear function:

**Definition 3.1.** A function $f$ is a convex piecewise-linear function if for some finite set of reals, $a_0 < a_1 < ... < a_n$, (allowing $a_0 = -\infty$ and $a_n = \infty$), we have:
\[ f(z) = \begin{cases} c_1(z - a_1) + f(a_1) & \text{if } z \in [a_0, a_1] \\ c_{i+1}(z - a_i) + f(a_i) & \text{if } z \in (a_i, a_{i+1}] \end{cases} \]
where $c_1 < c_2 < ... < c_n$ and $f(a_1) \in \mathbb{R}$. We define $a_0, a_1, ..., a_n$ as the vertices of $f$, $p(f) = n$ ($p(f)$ denotes the number of linear pieces in $f$) and $c_i(z - a_{i-1}) + f(a_{i-1})$ for $z \in [a_{i-1}, a_i]$ the $i$th linear piece of $f$.

Clearly, if $f$ is a convex piecewise-linear function, then we can store all the “information” about $f$ in a finite vector of size $O(p(f))$.

**Property 3.1.** Let $f_1(x)$, $f_2(x)$ be two convex piecewise-linear functions. Then, $f_1(ax + b)$, $c_1f_1(x) + df_2(x)$ are also convex piecewise-linear functions, for any real numbers $a$, $b$, $c$ and $d$, where $c \geq 0, d \geq 0$.

**Definition 3.2.** Let $S = \{f_1, f_2, ..., f_k\}$ be a set of convex piecewise-linear functions, and let $\Psi_t : \mathbb{R}^k \rightarrow \mathbb{R}$ be:
\[ \Psi_t(x) = \begin{cases} 0 & \text{if } \sum_{i=1}^k x_i = t \\ \infty & \text{otherwise} \end{cases} \]
We say $I_S(t) = \min_{x \in \mathbb{R}^k} \{ \psi_t(x) + \sum_{i=1}^k f_i(x_i) \}$ is a interpolation of $S$.

**Lemma 3.1.** Let $f_1$, $f_2$ be two convex piecewise-linear functions, and suppose $z_1^*, z_2^*$ are vertices of $f_1$, $f_2$ and $z_i^* = \arg \min_i f_i(z)$. Let $n = \{f_1, f_2\}$, then $I_S(t)$ (the interpolation of $S$) is a convex piecewise-linear function and it can be computed in $O(p(f_1) + p(f_2))$ operations.

**Proof.** We prove this lemma by describing a procedure to construct $I_S(t)$. The idea behind construction of $I_S(t)$ is essentially to “stitch” together the linear piece
of \( f_1 \) and \( f_2 \). Specifically, let \( g(t) \) be the function which is defined only at \( z_1^* + z_2^* \), with \( g(z_1^* + z_2^*) = f_1(z_1^*) + f_2(z_2^*) \), and let \( L_1 = z_1^*, L_2 = z_2^*, U_1 = z_1^*, U_2 = z_2^* \). We will construct \( g \) iteratively and eventually have \( g(t) = I_S(t) \). The construction is described as follows: at each iteration, let \( X_1 \) (and \( X_2 \)) be the linear piece of \( f_1 \) (and \( f_2 \)) at the left side of \( L_1 \) (and \( L_2 \)). Then choose the linear piece with the larger slope from \( \{X_1, X_2\} \), and “stitch” this piece onto the left side of the left endpoints of \( g \). If \( P_i \) is chosen, update \( L_i \) to the vertex which is on the right side of \( P_i \). As an example, consider \( f_1 \) and \( f_2 \) in Figure 2, then \( z_1^* = 1 \) and \( z_2^* = 0 \) are vertices of \( f_1 \) and \( f_2 \) such that \( z_1^* = \arg \min f_1(z), z_2^* = \arg \min f_2(z) \). Note that the linear piece \( X_1 \) in the procedure is labeled as \( P1 \) on the graph, while \( X_2 \) does not exist (since there is no linear piece for \( f_2 \) on the right side of \( z_2 \)). Hence, we “stitch” \( P1 \) to the left side of \( g \), and update \( L_1 \) to 0.

Similarly, let \( Y_1 \) (\( Y_2 \)) be the linear piece of \( f_1 \) (\( f_2 \)) at the right side of \( U_1 \) (\( U_2 \)), then choose the linear piece with the smaller slope and “stitch” this piece onto the right side of the right endpoints of \( g \). If \( Q_i \) is chosen, update \( U_i \) to the vertex which is on the right side of \( P_i \). Again, we use \( f_1 \) and \( f_2 \) in Figure 2 as an illustration. Then the linear piece \( Y_1 \) in the procedure is labeled as \( P2 \), while \( Y_2 \) is labeled as \( P3 \). As \( P2 \) has a lower slope than \( P3 \), we “stitch” \( P2 \) to the right side of \( g \) and update \( U_1 \) to 2.

Repeat this procedure until both \( L_1 \) (and \( L_2 \)) and \( U_1 \) (and \( U_2 \)) are the leftmost (and rightmost) endpoints of \( f_1 \) (and \( f_2 \)), or both endpoints of \( g \) are infinity. See Figure 2 and Figure 3 as an illustration of interpolation of two functions.

Note that the total number of iterations is bounded by \( O(p(f_1) + p(f_2)) \), and each iteration takes at most constant number of operations. By construction, it is clear that \( g \) is a convex piecewise-linear function. Also, \( g(z_1^* + z_2^*) = f_1(z_1^*) + f_2(z_2^*) \), and by the way we constructed \( g \), we must have \( g(t) \leq \{ \Psi_i(x) + f_1(x_1) + f_2(x_2) \} \) for any \( t \in \mathbb{R} \). Therefore, we have \( g = I_S \), and we are done.

Remark. In the case where either of \( f_1 \) or \( f_2 \) do not have global minimal, the interpolation can be still obtained in \( O(p(f_1) + p(f_2)) \) operations, but it involves tedious analysis of different cases. As reader will notice, the implementation of Algorithm 2 does not require this case and hence we don’t discuss it here.

**Theorem 3.2.** Given a set \( S \) consisting of convex piecewise-linear functions, \( I_S(t) \) is a convex piecewise-linear function. Moreover, suppose \( |S| = k \), and \( P = \sum_{i \in S} p(f) \). Then, \( I_S(t) \) can be computed in \( O(P \cdot \log k) \) operations.

**Proof.** For the sake of simplicity, assume \( k \) is divisible by 2. Let \( S_1 = \{f_1, f_2, ..., f_k\} \), and \( S' = \{f_{k-1}, f_k\} \). Then one can observe that \( I' = I_S \), by definition of \( I_S \). By Lemma 3.1, each function \( S' \) is piecewise-linear, and \( S' \) can be computed in \( O(P) \) operations. Consider changing \( S \) to \( S' \) as a procedure of decreasing the number of convex piecewise-linear functions. This procedure reduces the number by a factor of 2 each time, and consumes \( O(P) \) operations. Hence, it takes \( O(\log k) \) procedures to reduce set \( S \) into a single convex piecewise-linear function. And hence, computing \( I_S(t) \) takes \( O(P \cdot \log k) \) operations.

**Definition 3.3.** Let \( S = \{f_1, f_2, ..., f_k\} \) be a set of convex piecewise-linear functions, \( a \in \mathbb{R}^k \), and let \( \Psi : \mathbb{R}^k \rightarrow \mathbb{R} \) be:

\[
\Psi_i(x) = \begin{cases} 
0 & \text{if } \sum_{i=1}^{k} a_i x_i = t, \\
\infty & \text{otherwise}
\end{cases}, \quad \forall v \in V
\]

We say \( I^S_S(t) = \min_{x \in \mathbb{R}^k} \{ \psi_i(x) + \sum_{i=1}^{k} f_i(x_i) \} \) is a scaled interpolation of \( S \).

**Corollary 3.1.** Given any set of convex piecewise-linear functions \( I^S_S(t) \) is a convex piecewise-linear function. Suppose \( |S| = k \), and \( P = \sum_{i \in S} p(f) \), then, the corners and slopes of \( I^S_S(t) \) can be found in \( O(P \cdot \log k) \) operations.

**Proof.** Let \( S = \{f_1(x), f_2(x), ..., f_k(x)\} \), and let \( S' = \{f'_1(x) = f_1(a_1x), f'_2(x) = f_2(a_2x), ..., f'_k(x) = f_k(a_kx)\} \). Then we have \( I^S_{S'}(t) = I^S_S(t) \), and the result follows immediately from Theorem 3.2.

**Corollary 3.2.** For any nonnegative integer \( t, e \in E \) such that \( e = (v, w) \) (or \( e = (w, v) \)), then \( m_{e-v} \), a message function at \( t \)-th iteration of Algorithm 2, is a convex piecewise-linear function.

**Proof.** We show this by doing induction on \( t \). For \( t = 0 \), \( m_{e-v} \) is a convex piecewise-linear function by definition. For \( t > 0 \), let us remind ourselves that \( m_{e-v} \) is \( \phi_e(z) + \min_{z \in \mathbb{R} \cap E_{w \setminus \{e\}}} \{ \psi_{\tilde{w}}(\tilde{z}) + \sum_{\tilde{e} \in E_{w \setminus \{e\}}} m_{\tilde{e}-w}^{-1}(\tilde{z} \circ \tilde{e}) \} \), for \( z \in \mathbb{R} \).

By induction hypothesis, any \( m_{e-v} \) is a piecewise-linear convex function. Suppose \( S = \{m_{\tilde{e}-w}^{-1}(\tilde{z})\} \), \( e \in E_{w \setminus \{e\}} \), and \( a = \mathbb{R}^{E_{w \setminus \{e\}}} \). Then, \( g(z) = \min_{z \in \mathbb{R} \cap E_{w \setminus \{e\}}} m_{e-w}^{-1}(z) \) is equal to \( I_S(z) \) for some real number \( c \) and \( d \) (i.e., a shifted scaled interpolation of \( S \)). By Corollary 3.1, \( g(z) \) is a convex piecewise-linear function, and \( \phi_e \) is a convex piecewise-linear function. Therefore, we have that \( m_{e-v} = g + \phi_e \) is a convex piecewise-linear function.
Corollary 3.3. Suppose the cost vector $c$ in $\text{MCF}$ is integral, then at iteration $t$, for any defined message function $m_{e\rightarrow v}^t(z)$, let $\{s_1, s_2, \ldots, s_k\}$ be the slopes of $m_{e\rightarrow v}^t(z)$, then $-tc_{\text{max}} \leq s_i \leq tc_{\text{max}}$, and $s_i$ is integral for each $1 \leq i \leq k$.

Proof. We prove the corollary using induction. For $t = 0$, the statement is clearly true. For $t \geq 1$, we recall that $m_{e\rightarrow v}^t(z) - \phi_e(z) = \operatorname{min}_{z \in \mathbb{R}^{|\mathcal{E}|}} \{\psi_w(z) + \sum_{e \in \mathcal{E} \setminus \mathcal{E}_e} m_{e\rightarrow w}^{t-1}(z_{\mathcal{E}}')\}$ is a scaled interpolation of message functions in the form of $m_{e\rightarrow w}^{t-1}$ (possibly shifted). Since $\Delta(v, e) = \pm 1$, the absolute values of the slopes for the linear pieces of $m_{e\rightarrow v}^t - \phi_e$ is same as the absolute values of the slopes for the linear pieces of message functions $m_{e\rightarrow w}^{t-1}$ which was used for interpolation. Therefore, by induction hypothesis, the absolute values of the slopes of $m_{e\rightarrow v}^t - \phi_e$ are integral and bounded by $(t-1)c_{\text{max}}$. This implies the absolute values of slopes of $m_{e\rightarrow v}^t$ are integral and bounded by $tc_{\text{max}}$.

Corollary 3.4. Suppose the vectors $b$ and $u$ are integral in $\text{MCF}$, then at iteration $t$, for any message function $m_{e\rightarrow v}^t$, the vertices of $m_{e\rightarrow v}^t$ are also integral.

Proof. Again, we use induction. For $t = 0$, the statement is clearly true. For $t \geq 1$, as $m_{e\rightarrow v}^t - \phi_e$ is a piecewise-linear function obtained from a shifted scaled interpolation of functions $m_{e\rightarrow w}^{t-1}$. Since $\Delta(v, e) = \pm 1$, and $b$ is integral, the shift of the interpolation is integral, and as $\Delta(v, e) = \pm 1$, the scaled factor is $\pm 1$. As all vertices in any function $m_{e\rightarrow w}^{t-1}$ are integral, we deduce that all vertices of $m_{e\rightarrow v}^t - \phi_e$ must also be integral. Since $u$ is integral, all vertices of $\phi_e$ are integral, and therefore, all vertices of $m_{e\rightarrow v}^t$ are integral.

Corollary 3.1 and Corollary 3.2 shows that at each iteration, every message function can be encoded in terms of a finite vector describing the corners and slopes of its linear pieces in finite number of iterations. Using similar argument, we can see that the message functions in Algorithm 1 are also convex piecewise-linear functions, and thus can also be encoded in terms of a finite vector describing the corners and slopes of its linear pieces. We would like to note the result that message functions $m_{e\rightarrow v}^t$ are convex piecewise-linear functions can also be shown by sensitivity analysis of LP (see Chapter 5 of [7]).

3.2 BP on Special Min-cost Flow Problems Although we have showed in Section 3.1 that each message functions can be computed as piecewise-linear convex functions, those functions tend to become complicated to compute and store as the number of iterations grows. When an instance of $\text{MCF}$ has $b$, $u$, integral, then $m_{e\rightarrow v}^t$ (for some $e = (u, v)$, $t \in \mathbb{N}^+$) is a piecewise-linear function with at most $u_e$ linear pieces. Thus, if $u_e$ is bounded by some constant for all $e$, while $b$ and $u$ are integral, then the message functions at each iteration are much simpler. Next, we present a subclass of $\text{MCF}$, which can be solved quickly using BP compare to the more general $\text{MCF}$. Given a directed graph $G = (V, E)$, this subclass of problem is defined as follows:

$$\min \sum_{e \in E} c_e x_e \quad (\text{MCF}^t)$$

$$\sum_{e \in E_v} \Delta(v, e)x_e = b_v, \quad \forall v \in V$$

$$\sum_{e \in \delta(v)} x_e \leq \tilde{u}_v, \quad \forall v \in V, \delta(v) = \{(u, v) \in E\}$$

$$0 \leq x_e \leq u_e, \quad \forall e \in E$$

c, $u$, and $\tilde{u}$ are all integral, and all of their entries are bounded by a constant $K$.

To see $\text{MCF}^t$ is indeed a $\text{MCF}$, observe that for each $v \in V$, if we split $v$ into two vertices $v_{in}$ and $v_{out}$, where $v_{in}$ is incident to all in-arcs of $v$ with $b_{v_{in}} = 0$, and $v_{out}$ is incident to all out-arcs of $v$ with $b_{v_{out}} = b_v$; while create an arc from $v_{in}$ to $v_{out}$, set the capacity of the arc to be $\tilde{u}_v$, and cost of the arc to be 0. Let the new graph be $G'$, then, the $\text{MCF}$ on $G'$ is equivalent to $\text{MCF}^t$. Although we can use Algorithm 2 to solve the $\text{MCF}$ on the new graph, we would like to define functions $\psi_v, \phi_v, \forall v \in V, \forall e \in E$:

$$\psi_v(z) = \begin{cases} 0 & \text{if } \sum_{e \in \delta(v)} \Delta(v, e)z_e = b_v \\
\infty & \text{otherwise} \end{cases}$$

$$\phi_e(z) = \begin{cases} c_e z & \text{if } 0 \leq z \leq u_e \\
\infty & \text{otherwise} \end{cases}$$

and apply Algorithm 2 with defined $\psi$ and $\phi$. When we update message functions $m_{e\rightarrow v}^t$ for all $e \in E_v$, the inequality $\sum_{e \in \delta(v)} x_e \leq \tilde{u}_v$, allow us to only look at the $\tilde{u}_v$ linear pieces from message functions $m_{e\rightarrow w}^{t-1}$ for all but constant number of $e \in E_w$. This allows us to further trim the running time of BP for $\text{MCF}^t$, and we will present this more formally in Section 6.1.

4 Convergence of BP for $\text{MCF}$

Before proving Theorem 3.1, we define the computation trees, and then establish the connection between computation trees and BP. Using this connection, we proceed to prove the main result of our paper.
4.1 Computation Tree and BP Let $T_e^N$, the $N$-computation tree corresponding to variable $x_e$, be defined as follows:

1. Let $V(T_e^N)$ be the vertex set of $T_e^N$ and $E(T_e^N)$ be the arc set of $T_e^N$. Let $\Gamma(\cdot)$ be the map which maps each vertex $v' \in V(T_e^N)$ to a vertex $v \in V$. And $\Gamma$ preserves the arcs (i.e. For each $e' = (v'_a, v'_b) \in E(T_e^N)$, $\Gamma(e') = (\Gamma(v'_a), \Gamma(v'_b)) \in E$).

2. Divide $V(T_e^N)$ into $N + 1$ levels, on the 0-th level, we have a “root” arc $e$, and $\Gamma(e) = e$. And a vertex $v'$ on the $t$-th level of $T_e^N$ if $v'$ is $t$ arcs apart from either nodes of $e$.

3. For any $v' \in V(T_e^N)$, let $E(v')$ be the set of all arcs incident to $v'$. If $v'$ is on the $t$-th level of $T_e^N$, and $t < N$, then $\Gamma(E(v')) = E(\Gamma(v'))$ (i.e. $\Gamma$ preserves the neighborhood of $v'$).

4. For every vertex $v'$ that is on the $N$-th level of $T_e^N$, $v'$ is incident to exactly one arc in $T_e^N$.

In other literatures, $T_e^N$ is known as the “unwrapped tree” of $G$ rooted at $e$. Figure 4 gives an example of a computation tree. We make a note that the definition of computation trees we have introduced is slightly different from the definition in other papers [4] [6] [18], although the analysis and insight for computation trees is very similar.

Let $V^\circ(T_e^N)$ be the set of all the vertices which are not on the $N$-th level of $T_e^N$. Consider the problem

\[
\min \sum_{f \in E(T_e^N)} c_{\Gamma(f)}x_f \quad (\mathcal{MCF}_e^N) \\
\sum_{f \in E_x} \Delta(v, f)x_f = b_{\Gamma(v)} , \quad \forall v \in V^\circ(T_e^N) \\
0 \leq x_f \leq u_{\Gamma(f)}, \quad f \in E(T_e^N)
\]

Loosely speaking, $\mathcal{MCF}_e^N$ is almost an $\mathcal{MCF}$ on $T_e^N$. There is a flow constraint for every arc $e \in E(T_e^N)$, and a demand/supply constraint for every node $v \in V^\circ(T_e^N)$. Now, we state the lemma which exhibits the connection between BP and the computation trees.

**Lemma 4.1.** $\mathcal{MCF}_e^N$ has an optimal solution $y^*$, satisfying $y^*_e = x^*_e$ ($r$ is the root of $T_e^N$, $\Gamma(r) = e$).

This result is not too surprising as BP is exact on trees. A thorough analysis can show that the message functions in BP are essentially the implementation of an algorithm using the “bottom up” (aka dynamic programming) approach, starting from the bottom level of $T_e^N$. A more formal, combinatorial proof of this lemma also exists, which is a rather technical argument using induction. Proofs for similar statements can be found in [4] and [18].

4.2 Proof of the Main Theorem

**Proof.** [Proof of Theorem 3.1] Suppose $\exists e_0 \in E$ and $N \geq \left(\left\lceil \frac{\log |X|}{\log |E|}\right\rceil + 1\right)$ such that $x^*_e \neq x^*_e$. By Lemma 4.1, we can find an optimal solution $y^*$ for $\mathcal{MCF}_e^N$, where $y^*_e = x^*_e$. Now, without loss of generality, we assume $y^*_e > x^*_e$. Let $r = (v_\alpha, v_\beta)$, because $y^*$ is feasible for $\mathcal{MCF}_e^N$ and $x^*$ is feasible for $\mathcal{MCF}$, we have

\[
b_{\Gamma(v_\alpha)} = \sum_{e \in E_{\Gamma(v_\alpha)}} \Delta(v_\alpha, e)y^*_e = y^*_r + \sum_{e \in E_{\Gamma(r)} \setminus \Gamma(r)} \Delta(v_\alpha, e)y^*_e \\
b_{\Gamma(v_\beta)} = \sum_{e \in E_{\Gamma(v_\beta)}} \Delta(v_\beta, e)x^*_e = x^*_r + \sum_{e \in E_{\Gamma(c)} \setminus \Gamma(c)} \Delta(v_\beta, e)x^*_e
\]

Then, we can find arc $e_1'$ incident to $v_\alpha$, $e_1' \neq r$, such that $y^*_{e_1'} > x^*_{e_1'}$ if $e_1'$ has the same orientation as $r$, and $y^*_{e_1'} < x^*_{e_1'}$ if $e_1'$ has the opposite orientation as $r$. Similarly, we can find some arc $e_{-1}'$ incident to $v_\beta$ satisfying the same condition. Let $v_\alpha_1, v_\alpha_{-1}$ be the vertices on $e_1', e_{-1}'$ which are one level below $v_\alpha, v_\beta$ on $T_e^N$, then, we can find some arc $e_2', e_{-2}'$ satisfying the same condition. If we continue this all the way down to the leaf nodes of $T_e^N$, we can find the set of arcs \{$e_{-N}', e_{-N+1}', \ldots, e_{-1}', e_1', \ldots, e_{N}'$\} such that

\[
y^*_{e_1'} > x^*_{e_1'} \iff e_1' \text{ has the same orientation as } r \\
y^*_{e_1'} < x^*_{e_1'} \iff e_1' \text{ has the opposite orientation as } r
\]

Let $X = \{e_{-N}', e_{-N+1}', \ldots, e_{-1}', e_0', \ldots, e_{N}'\}$. For any $e' = (v_p, v_q) \in X$, let $\text{Aug}(e') = (v_p, v_q)$ if $y^*_{e'} > x^*_{e'}$, and $\text{Aug}(e') = (v_q, v_p)$ if $y^*_{e'} < x^*_{e'}$. Then, each $\Gamma(\text{Aug}(e'))$ is an arc on the residual graph $G(x^*)$, and $W = (\text{Aug}(e_{-N}'), \text{Aug}(e_{-N+1}'), \ldots, \text{Aug}(e_0'), \ldots, \text{Aug}(e_N'))$ is a directed path on $T_e^N$. We call $W$ the augmenting path of $y^*$ with respect to $x^*$. Also, $\Gamma(W)$ is a directed path

\[285\]

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
walk on $G(x^*)$. Now we can decompose $\Gamma(W)$ into a simple directed path $P(W)$, and a set of simple directed cycles $\text{Cyc}(W)$. Because each simple directed cycle/path on $G(x^*)$ can have at most $n$ arcs, and $W$ has $2N + 1$ arcs, where $N \geq (\lfloor \frac{1}{2\epsilon_{\text{max}}} \rfloor + 1)n$, we have that $|\text{Cyc}(W)| > \frac{L}{\delta(x^*)}$. Then, we have

$$
c^*(W) = \sum_{C \in \text{Cyc}(W)} c^*(C) + c^*(P(W))
$$

$$\geq \sum_{C \in \text{Cyc}(W)} c^*(C) - L
$$

$$> \frac{L}{\delta(x^*)} \cdot \delta(x^*) - L
$$

$$= 0
$$

Let $\text{Forw} = \{e \in E : y^*_e > x^*_\text{Forw}(e)\}$, $\text{Back} = \{e \in E : y^*_e < x^*_\text{Back}(e)\}$. Since both $\text{Forw}$ and $\text{Back}$ are finite, we can find $\lambda > 0$ such that $y^*_e - \lambda \geq x^*_\text{Forw}(e)$, $\forall e \in \text{Forw}$ and $y^*_e + \lambda \geq x^*_\text{Back}(e)$, $\forall e \in \text{Back}$. Let $\tilde{y} \in \mathbb{R}^{|E|} \{\text{such that}

$$
\tilde{y}_e = \begin{cases}
y^*_e - \lambda & \text{if } e \in \text{Forw} \\
y^*_e + \lambda & \text{if } e \in \text{Back} \\
0 & \text{otherwise}
\end{cases}
$$

We can think $\tilde{y}$ as pushing $\lambda$ units of flow on $W$ for $y^*$. Since for each $e \in \text{Forw}$, $y^*_e - \lambda \geq x^*_\text{Forw}(e) \geq 0$, and for each $e \in \text{Back}$, $y^*_e + \lambda \leq x^*_\text{Back}(e) \leq u^e_\text{Back}(e)$, $\tilde{y}$ satisfy all the flow constraints. Also, because $\text{Forw} = \{e \in E, e \text{ has the same orientation as } r\}$, and $\text{Back} = \{e \in E, e \text{ has the opposite orientation as } r\}$, we have that for any $v \in V^e(T^N)$, $\sum_{e \in E_v} \Delta(v, e) \tilde{y}_e = \sum_{e \in E_v} \Delta(v, e) y^*_e = b^e_v$, which implies $\tilde{y}$ satisfies all the demand/supply constraints. Therefore, $\tilde{y}$ is feasible solution for $(\mathcal{MCF}^N)$. But

$$
\sum_{e \in E(T^N)} c^e(\tilde{y}_e) - \sum_{e \in E(T^N)} c^e(b^e_v) = \sum_{e \in \text{Forw}} c^e(\tilde{y}_e) \lambda - \sum_{e \in \text{Back}} c^e(\tilde{y}_e) \lambda
$$

$$= c^s(W) \lambda
$$

$$> 0
$$

This contradicts the optimality of $y^*$, and completes the proof of Theorem 3.1.

**Corollary 4.1.** Given an $\mathcal{MCF}$ with integral data $c$, $b$ and $u$. Suppose $\epsilon_{\text{max}} = \max_{e \in E} c^e_\text{max}$. Let Algorithm 2 run $N = n^2 \epsilon_{\text{max}} + n$ iterations. Let $z^*_e = \arg n^e_N(z^*)$, then we have $n^e_N(z^*_e - 1) - n^e_{\text{max}} > n^e_N(z^*_e) + 1 - n^e_{\text{max}} > n^e_N(z^*_e + 1)$ iff $\mathcal{MCF}$ has a unique optimal solution.

**Proof.** The proof of this corollary uses the same idea of finding an augmenting path on the computation tree as in Theorem 3.1.

$(\Rightarrow)$: Suppose $\mathcal{MCF}$ has a unique optimal solution. Let $y$ be the optimal solution $\mathcal{MCF}_N^y$, when the variable at $r$, $y_r$ is fixed to be $z^*_e - 1$. Then, we can find an augmenting path $W$ of $y$ with respect to $x^*$ of length $2n^2 \epsilon_{\text{max}}$. Then $W$ can be decomposed into at least $2n^e_{\text{max}}$ disjoint cycles and one path. Since each cycle has a cost of at most $-\delta(x^*)$, which is at least $-1$ as $\mathcal{MCF}$ has integral data. Hence, once we push 1 unit of flow of $y$ on $W$, we can decrease the cost of $y$ by at least $n^e_{\text{max}}$. Hence $n^e_N(z^*_e - 1) - n^e_{\text{max}} < n^e_N(z^*_e)$. Similarly, we can also show that $n^e_N(z^*_e) < n^e_N(z^*_e + 1) + n^e_{\text{max}}$.

$(\Leftarrow)$: Suppose $\mathcal{MCF}$ does not have a unique optimal solution. Let $x^*$ be an optimal solution of $\mathcal{MCF}$, and $y$ be the optimal solution $\mathcal{MCF}_N^y$. Again, we can find an augmenting path $W$ of $y$ with respect to $x^*$ of length $2n^2 \epsilon_{\text{max}}$, and we can decompose $W$ into at least $2n^e_{\text{max}}$ disjoint cycles and one path $P$. The cost of $P$ is at most $(n - 1) \epsilon_{\text{max}}$, while each cycle has a non-positive cost. Hence, when we push 1 unit of flow of $y$ on $W$, we increase the cost of $y$ by at most $(n - 1) \epsilon_{\text{max}} < n^e_{\text{max}}$. Therefore, depends on the orientation of $r$ on $W$, we either have $n^e_N(z^*_e - 1) - n^e_{\text{max}} \leq n^e_N(z^*_e)$ or $n^e_N(z^*_e + 1) - n^e_{\text{max}} \leq n^e_N(z^*_e)$.

Theorem 4.1 shows that BP can be used to detect the existence of an unique optimal solution for $\mathcal{MCF}$.

5 Extension of BP on Convex-Cost Network Flow Problems

In this section, we discuss the extension of Theorem 3.1 to convex-cost network flow problem (or convex flow problem). Convex flow problem is defined on graph $G = (V, E)$ as follows:

$$
\min \sum_{e \in E} c_e(x_e) \quad (CP)
$$

$$\sum_{e \in E_v} \Delta(v, e) x_e = b^e_v, \quad \forall v \in V
$$

$$0 \leq x_e \leq u_e, \quad \forall e \in E
$$

where each $c_e$ is a convex piecewise-linear function. Notice that if we define $\psi$ exactly same as we did for $\mathcal{MCF}$, and for each $e \in E$, define

$$
\phi_e(z) = \begin{cases}
c_e(z) & \text{if } 0 \leq z \leq u_e \\
\infty & \text{otherwise}
\end{cases}
$$

then, we can apply Algorithm 2 on the graph $G$ with functions $\psi$ and $\phi$, and hence obtain BP on convex flow problem.
Now, we give the definition of a residual graph on convex cost flow problem. Suppose \(x\) is a feasible solution for \((CP)\), let \(G(x)\) be the residual graph for \(G\) and \(x^*\) defined as follows: \(\forall e = (v_{\alpha}, v_{\beta}) \in E\), if \(x_e < u_e\), then \(e\) is also an arc in \(G(x)\) with cost \(c^e_e = \lim_{t \to 0} \frac{c(x_{\alpha} + \frac{1}{t}) - c(x_{\alpha})}{c(x_{\alpha}) - c(x_{\beta})}\) (the value of the slope of \(c_e\) at the right side of \(x_e\)); if \(x_e > 0\), then there is an arc \(e' = (v_{\beta}, v_{\alpha})\) in \(G(x)\) with cost \(c^e_{e'} = \lim_{t \to 0} \frac{c(x_e - \frac{t}{c}) - c(x_e)}{c(x_e) - c(x_{\beta})}\) (negative value of the slope of \(c_e\) at the left side of \(x_e\)). Finally, let

\[
\delta(x) = \min_{c \in C} \left\{ \sum_{e \in C} c^e_e \right\},
\]

where \(C\) is the set of directed cycles in \(G(x)\).

**Theorem 5.1.** Suppose \(x^*\) is the unique optimal solution for \((CP)\). Let \(L\) be the maximum cost of a simple directed path in \(G(x^*)\), and by uniqueness of optimal solution, \(\delta(x^*) > 0\). Then, for any \(N \geq (\frac{1}{2\delta(x^*)} + 1)m\), \(\bar{z}^N = x^*\). Namely, the BP algorithm converges to the optimal solution in at most \((\frac{1}{2\delta(x^*)} + 1)m\) iterations.

The proof of Theorem 5.1 is almost identical to the proof of Theorem 3.1. Theorem 5.1 is an illustration of the power of BP, that not only can solve \(\text{MCF}\), but possibly many other variants of \(\text{MCF}\) as well.

### 6 Running Time of BP on \(\text{MCF}\)

In the next two sections, we will always assume vectors \(c, u\) and \(b\) in problem \(\text{MCF}\) are integral. This assumption is not restrictive in practice, as the \(\text{MCF}\) solved in practice usually have rational data sets [1]. We define \(c_{\max} = \max_{e \in E} \{c_e\}\), and provide an upper bound for the running time of BP on \(\text{MCF}\) in terms \(m\) (the number of arcs in \(G\)), \(n\) (the number of nodes in \(G\)), and \(c_{\max}\).

**Lemma 6.1.** Suppose \(\text{MCF}\) has integral data, in Algorithm 2, the number of operations to update all the messages at iteration \(t\) is \(O(nm \log n \cdot t_{c_{\max}})\).

**Proof.** At iteration \(t\), fix a valid message function \(m^t_{\bar{x} \rightarrow w}\), and we update it as:

\[
m^t_{\bar{x} \rightarrow w}(z) = \phi_e(z) + \min_{\bar{\epsilon} \in \mathbb{R}^{|E|}, \bar{\epsilon}_z = z} \left\{ \psi_w(z) + \sum_{\bar{\epsilon} \in E \setminus e} m^{t-1}_{\bar{\epsilon} \rightarrow w}(\bar{\epsilon}_z) \right\}
\]

As \(\text{MCF}\) has integral data, by Corollary 3.3, any messages of the form \(m^{t-1}_{\bar{x} \rightarrow w}\) has integral slopes, and the absolute values of the slopes are bounded by \((t - 1)c_{\max}\). This implies for each \(m^{t-1}_{\bar{x} \rightarrow w}\), the number of linear pieces of \(m^{t-1}_{\bar{x} \rightarrow w}\) is bounded by \(2(t - 1)c_{\max}\).

As the size of \(E_w\) is at most \(n\), by Corollary 3.1, \(g(z) = \min_{\bar{\epsilon} \in \mathbb{R}^{|E|}, \bar{\epsilon}_z = z} \{\psi_w(z) + \sum_{\bar{\epsilon} \in E \setminus e} m^{t-1}_{\bar{\epsilon} \rightarrow w}(\bar{\epsilon}_z)\}\) can be calculated in \(O(\log n \cdot ntc_{\max})\) operations. As \(m^{t-1}_{\bar{x} \rightarrow w}(z) = g(z) + \phi_e(z)\), we have that \(m^t_{\bar{x} \rightarrow w}\) can also be calculated in \(O(\log n \cdot ntc_{\max})\) operations. Since at each iteration, we update \(2m\) message functions, the total number of computations at iteration \(t\) can be bounded by \(O(nm \log n \cdot t_{c_{\max}})\).

**Theorem 6.1.** Given \(\text{MCF}\) has a unique optimal solution \(x^*\) and integral data, the BP algorithm finds the unique optimal solution of \(\text{MCF}\) in \(O(n^3m \log n \cdot c_{\max}^3)\) operations.

**Proof.** Because \(c\) is integral, the value \(\delta(x^*)\) in Theorem 3.1 is also integral. Recall that \(L\) is the maximum cost of a simple directed path in \(G(x^*)\). Since simple directed path has at most \(n - 1\) arcs, \(L\) is bounded by \(n c_{\max}\). Thus, by Theorem 3.1, the BP algorithm (Algorithm 2) converges to the optimal solution of \(\text{MCF}\) after \(O(n^2c_{\max})\) iterations. Combine this with Lemma 6.1, we have that BP converges to the optimal solution in \(O(n^2c_{\max} \cdot nm \log n \cdot ntc_{\max}^3) = O(n^3m \log n \cdot c_{\max}^3)\) operations.

We would like to point out that for \(\text{MCF}\) with a unique optimal solution, Algorithm 2 can take an exponential number of iterations to converge. Consider the \(\text{MCF}\) on the directed graph \(G\) shown in Figure 5. Take a large positive integer \(D\), set \(c_{e_1} = c_{e_2} = D\), \(c_{e_3} = 2D - 1\), \(b_{e_1} = 1\), \(b_{e_2} = 0\) and \(b_{e_3} = 1\). It can be checked that \(\hat{x}^N\) alternates between 1 and -1 when \(2N + 1 < 2D\). This means that BP algorithm takes at least \(\Omega(D)\) iterations to converge. Since the input size of a large \(D\) is just \(\log(D)\), we have that Algorithm 2 for \(\text{MCF}\) does not converge to the unique optimal solution in polynomial time.

#### 6.1 Runtime of BP on \(\text{MCF}'\)

Here we analyze the run time of BP on \(\text{MCF}'\), which is defined in Section 3.2. We show that BP performs much better on this special class of \(\text{MCF}\). Specifically, we state the following theorem:

**Theorem 6.2.** If \(\text{MCF}'\) has a unique optimal solution, Algorithm 2 for \(\text{MCF}'\) finds the unique optimal solution in \(O(n^2)\) operations.
First, we state the convergence result of BP for $\mathcal{MCF}'$ which is reminiscent of Theorem 3.1.

**Theorem 6.3.** Suppose $\mathcal{MCF}'$ has a unique optimal solution $x^*$. Define $L$ to be the maximum cost of a simple directed path in $G(x^*)$. Then, Algorithm 2 for $\mathcal{MCF}'$ converges to the optimal solution in at most $(\frac{1}{2L} + 1)n$ iterations.

The proof of this theorem is exactly the same as the proof of Theorem 3.1, and it is hence omitted. Now, we apply this result to compute the running time of Algorithm 2 for $\mathcal{MCF}'$.

**Proof.** [Proof of Theorem 6.2] For the simplicity of the proof, we assume that every linear piece in a message function has unit length. This assumption is not restrictive, as each linear piece in general has integral vertices (from Corollary 3.4), so we can always break a bigger linear piece into many unit length linear pieces with the same slopes. At iteration $t \geq 1$, recall $m^{t-1}_{\tilde{v} \rightarrow w}(\tilde{z})$ is defined as:

$$
\phi_\epsilon(z) + \min_{\tilde{z} \in \mathbb{R}^{E_w}} \left\{ \psi_w(\tilde{z}) + \sum_{\tilde{e} \in E_w \setminus e} m^{t-1}_{\tilde{e} \rightarrow w}(\tilde{z}_e) \right\}
$$

We claim that each $m^{t-1}_{\tilde{v} \rightarrow w}$ can be constructed using the following information:

1. $\tilde{u}_w$ smallest linear pieces in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in \delta(w) \setminus e\}$, where $\delta(w)$ denotes all in-arcs of $w$;
2. $(\tilde{u}_w - b_w)$ smallest linear pieces in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in \gamma(w) \setminus e\}$ where $\gamma(w)$ denotes all out-arcs of $w$;
3. The value of $\sum_{\tilde{z} \in E_e} m^{t-1}_{\tilde{e} \rightarrow w}(0)$.

Observe that once all three sets of information is given, for any integer $0 \leq z \leq u_e$, one can find $m^{t-1}_{\tilde{v} \rightarrow w}(z)$ in a constant number of operations. As $u_e$ is also bounded by a constant, we have that $m^{t-1}_{\tilde{v} \rightarrow w}$ can be computed in constant time when information (1), (2) and (3) are given.

Now, since $|E_w|$ is bounded by $n$, obtaining each one of information (1), (2) or (3) for a specific message $m^{t-1}_{\tilde{v} \rightarrow w}$, takes $O(n)$ operations. When we update all the messages in the set $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in E_w\}$, note that every $e \in E_w$ except for a constant number of them, information (1) for updating $m^{t-1}_{\tilde{v} \rightarrow w}$ is equivalent to $\tilde{u}_w$ smallest pieces in $m^{t-1}_{\tilde{v} \rightarrow w}$, for some $\tilde{e} \in \delta(w)$; and information (2) for updating $m^{t-1}_{\tilde{v} \rightarrow w}$ is equivalent $\tilde{u}_w - b_w$ smallest linear pieces in $m^{t-1}_{\tilde{v} \rightarrow w}$ for some $\tilde{e} \in \gamma(w)$. In fact, the exceptions are precisely those $e$ such that $m^{t-1}_{\tilde{v} \rightarrow w}$ contain at least one of those $\tilde{u}_w$ smallest linear pieces in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in \delta(w)\}$ or one of those $(\tilde{u}_w - b_w)$ smallest linear pieces in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in \gamma(w)\}$. Moreover, update any message in the set $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in E_w\}$ has information (3) to be exactly $\sum_{\tilde{z} \in E_e} m^{t-1}_{\tilde{e} \rightarrow w}(0)$. Hence, to update all messages in the set $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in E_w\}$, we can obtain information (1), (2), (3) for all the messages in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in E_w\}$, and hence can compute all messages in $\{m^{t-1}_{\tilde{v} \rightarrow w}| \tilde{e} \in E_w\}$ in $O(n)$ operations.

Therefore, updating all the message functions at every iteration takes $O(n^2)$ operations. By Theorem 6.2, since $L$ is bounded by $nk$, $\delta(x^*) \geq 1$, then $(\frac{1}{2L} + 1)n$ is bounded by $O(n^2)$. This concludes that Algorithm 2 finds a unique optimal solution in $O(n^2)$ operations.

Note both the shortest-path problem and bipartite matching problem belongs to the class of $\mathcal{MCF}'$, where $\tilde{u}, b, u$ are all bounded by 2.

**7 FPRAS Implementation**

In this section, we provide a fully polynomial-time randomized approximation scheme (FPRAS) for $\mathcal{MCF}$ using BP as a subroutine. First, we describe the main idea of our approximation scheme.

As Theorem 6.1 indicated, in order to come up with an efficient approximation scheme using BP (Algorithm 2), we need to get around the following difficulties:

1. The convergence of BP requires $\mathcal{MCF}$ to have a unique optimal solution.
2. The running time of BP is polynomial in $m, n$ and $c_{\max}$.

For an instance of $\mathcal{MCF}$, our strategy for finding an efficient $(1 + \epsilon)$ approximation scheme is to find a modified cost vector $\tilde{c}$ (let $\overline{\mathcal{MCF}}$ be the problem with the modified cost vector) which satisfies the following properties:

1. The $\overline{\mathcal{MCF}}$ has a unique optimal solution.
2. $\tilde{c}_{\max}$ is polynomial in $m, n$ and $\frac{1}{\epsilon}$.
3. The optimal solution of $\overline{\mathcal{MCF}}$ is a “near optimal” solution of $\mathcal{MCF}$. The term “near optimal” is rather fuzzy and we will address this question later in the section.

First, in order to find a modified $\overline{\mathcal{MCF}}$ with a unique optimal solution, we first state a result which is a variant of the Isolation Lemma introduced in [15].

**7.1 Variation of Isolation Lemma**

**Theorem 7.1.** Let $\overline{\mathcal{MCF}}$ be an instance of min-cost flow problem with underlying graph $G = (V, E)$, demand vector $b$, constraint vector $u$. Let its cost vector $c$ be generated as follows: for each $e \in E$, $\tilde{c}_e$ is chosen
independently and uniformly over \( N_e \), where \( N_e \) is a discrete set of 4m positive numbers (m = \( |E| \)). Then, the probability that \( \overline{MCF} \) has a unique optimal solution is at least \( \frac{1}{4} \).

**Proof.** Fix an arc \( e_1 \in E \), and fix \( e \) for all \( e \in E \setminus e_1 \). Suppose that there exists \( \alpha \geq 0 \), such that when \( e_1 = \alpha \), \( \overline{MCF} \) have optimal solutions \( x^*, x^{**} \), where \( x_i^* = 0 \) and \( x_i^{**} > 0 \). Then, if \( e_1 > \alpha \), for any feasible solution \( x \) of \( \overline{MCF} \), where \( x_{e_1} > 0 \), we have

\[
\sum_{e \in E} \tilde{c}_e x_e^* = \sum_{e \in E, e \neq e_1} \tilde{c}_e x_e^* \\
< \sum_{e \in E, e \neq e_1} \tilde{c}_e x_e + x_{e_1} \alpha \\
< \sum_{e \in E} \tilde{c}_e x_e 
\]

And if \( e_1 < \alpha \), for any feasible solution \( x \) of \( \overline{MCF} \) where \( x_{e_1} = 0 \), we have

\[
\sum_{e \in E} \tilde{c}_e x_e^{**} = \sum_{e \in E, e \neq e_1} \tilde{c}_e x_e^{**} + \alpha x_{e_1}^{**} \\
< \sum_{e \in E, e \neq e_1} \tilde{c}_e x_e + \alpha x_{e_1} \\
= \sum_{e \in E} \tilde{c}_e x_e 
\]

This implies there exists at most one value for \( \alpha \), such that if \( e_1 = \alpha \) then \( \overline{MCF} \) have optimal solutions \( x^*, x^{**} \), where \( x_i^* = 0 \) and \( x_i^{**} > 0 \). Similarly, we can also deduce that there exists at most one value for \( \beta \), such that if \( e_1 = \beta \), \( \overline{MCF} \) have optimal solutions \( x^*, x^{**} \), where \( x_i^* < u_{e_1} \) and \( x_i^{**} = u_{e_1} \).

Let \( OS \) denote the set of all optimal solutions of \( \overline{MCF} \), and let \( D(\epsilon) \) be the condition of either: 0 = \( x_e \), \( \forall x \in OS \), or 0 < \( x_e < u_e \), \( \forall x \in OS \), or \( x_e = u_e \), \( \forall x \in OS \). Since \( e_1 \) has 4m possible values, where each value is chosen with equal probability, we conclude that the probability of \( D(\epsilon) \) is satisfied is at least \( \frac{4m - 2}{4m} = \frac{2m - 1}{2m} \). By the union bound of probability, we have that the probability of \( D(\epsilon) \) is satisfied for all \( e \in E \) is at least \( 1 - \sum_{e \in E} \frac{1}{2m} = \frac{1}{2} \). Now, we state the following lemma:

**Lemma 7.1.** If \( \forall \epsilon \in E \), condition \( D(\epsilon) \) is satisfied, then \( \overline{MCF} \) have a unique optimal solution.

Then by Lemma 7.1, we conclude that the probability that \( \overline{MCF} \) has a unique optimal solution is at least \( \frac{1}{2} \).

**Proof.** [Proof of Lemma 7.1] Suppose \( x^* \) and \( x^{**} \) are two distinct optimal solutions of \( \overline{MCF} \). Let \( d = x^{**} - x^* \), then \( x^* + \lambda d \) is an optimal solution of \( \overline{MCF} \) if 0 \leq (\( x^* + \lambda d \))_e \leq u_e, \( \forall e \in E \). As \( \epsilon_1 > 0 \) for any \( e \in E \), and \( \epsilon d = \epsilon d x^{**} - \epsilon d x^* = 0 \), there exists some \( \epsilon' \) such that \( d \epsilon' < 0 \). Let \( \lambda = \sup \{ \lambda : x^* + \lambda d \epsilon' = 0 \} \) is bounded; and since \( x^* + d = x^{**} \), \( \lambda > 0 \). By optimality of \( \lambda \), there must exists some \( \epsilon'' \) such that either \( (x^* + \lambda d)_{\epsilon''} = 0 \) or \( u_{e_{\epsilon''}} \). Since \( \epsilon > 0 \), \( x_{e_{\epsilon''}} \neq (x^* + \lambda d)_{\epsilon''} \), this contradicts the assumption that \( D(\epsilon'') \) is satisfied. Thus, \( \overline{MCF} \) must have a unique optimal solution.

We note that Theorem 7.1 can be easily modified for LP in standard form.

**Corollary 7.1.** Let \( \overline{LP} \) be an LP problem with constraint Ax = \( b \), where A is a \( m \times n \) matrix, \( b \in \mathbb{R}^m \). The cost vector \( \tilde{c} \) of \( \overline{LP} \) is generated as follows: for each \( e \in E \), \( c_e \) is chosen independently and uniformly over \( N_e \), where \( N_e \) is a discrete set of 2n elements. Then, the probability that \( \overline{LP} \) has a unique optimal solution is at least \( \frac{1}{2} \).

### 7.2 Finding the Correct Modified Cost Vector \( \tilde{c} \)

Next, we provide a randomly generate \( \tilde{c} \) with the desired properties stated in the beginning of this section. Let \( X : E \rightarrow \{1, 2, ..., 4m\} \) be a random function where for each \( e \in E \), \( X(e) \) is chosen independently and uniformly over the range. Let \( t = \frac{\max_{e \in E} X(e)}{m} \), and generate \( \tilde{c} \) as: for each \( e \in E \), let \( \tilde{c}_e = 4m \cdot \frac{X(e)}{4m} + X(e) \). Then, for any \( \tilde{c} \) generated at random, \( \tilde{c}_{\max} \) is polynomial in \( m, n \) and \( \frac{1}{2} \), and by Theorem 7.1, the probability of \( \overline{MCF} \) having a unique optimal solution is greater than \( \frac{1}{2} \).

Now, we introduce algorithm APRXMT(\( \overline{MCF}, \epsilon \)) which works as follows: select a random \( \tilde{c} \); try to solve \( \overline{MCF} \) using BP. If BP discovers that \( \overline{MCF} \) has no unique optimal solution, then we start the procedure by selecting another \( \tilde{c} \) at random, otherwise, return the unique optimal solution found by BP. Formally, we present APRXMT(\( \overline{MCF}, \epsilon \)) as Algorithm 3.

**Corollary 7.2.** The expected number of operations for APRXMT(\( \overline{MCF}, \epsilon \)) to terminate is \( O(\frac{m^n \log n}{\epsilon^2}) \).

**Proof.** With Theorem 7.1, when we call APRXMT(\( \overline{MCF}, \epsilon \)), the expected number of \( \overline{MCF} \) BP tried to solve is bounded by 2. For each selection of \( \tilde{c} \), we run Algorithm 2 for \( 2 \tilde{c}_{\max}^2 n^2 \) iterations. As \( \tilde{c}_{\max} = O(\frac{1}{\epsilon^2}) \), by Lemma 6.1, the expected number of operations for APRXMT(\( \overline{MCF}, \epsilon \)) to terminate is \( O(\frac{3^2 \tilde{c}_{\max}^2 n^2 \log n}{\epsilon^2}) = O(\frac{m^n \log n}{\epsilon^2}) \).

Now let \( \tilde{c} \) be a randomly chosen vector such that \( \overline{MCF} \) has a unique optimal solution \( x^* \). The next thing we want to show is that \( x^* \) is a “near optimal” solution of \( \overline{MCF} \). To accomplish this, let \( \epsilon' = \arg \max \{|e_i| \} \) and define
Algorithm 3 APRXMT($MCF, \epsilon$)

1. Let $t = \frac{\max c_e}{4mn}$, for any $e \in E$, assign $\tilde{c}_e = 4m \cdot \lfloor \frac{c_e}{t} \rfloor + p_e$, where $p_e$ is an integer chosen independently, uniformly random from $\{1, 2, \ldots, 4m\}$.
2. Let $\overline{MCF}$ be the problem with modified cost $\tilde{c}$.
3. Run Algorithm 2 on $\overline{MCF}$ for $N = 2\epsilon_{\text{max}}n^2$ iterations.
4. Use Corollary 4.1 to determine if $\overline{MCF}$ has a unique solution.
5. if $\overline{MCF}$ does not have a unique solution then
6. Restart the procedure APRXMT($MCF, \epsilon$).
7. else
8. Terminate and return $x^2 = \hat{x}^N$, where $\hat{x}^N$ if the “belief” vector found in Algorithm 2.
9. end if

\[ \begin{align*}
\min \sum_{e \in E} c_e x_e & \quad (\mathcal{MCF}) \\
\sum_{e \in E_v} \Delta(v, e)x_e = b_v, & \forall v \in V \quad \text{ (demand constraints)} \\
x_e' = x_e^2, & \quad \forall e \in E \quad \text{ (flow constraints)}
\end{align*} \]

Suppose $x^3$ is an optimal solution for $\mathcal{MCF}$, and $x^1$ is an optimal solution of $MCF$, then we have that $\epsilon'k \leq 0$. And
\[
\tilde{c}_e = 4m \lfloor \frac{c_e}{t} \rfloor + p_e, 1 \leq p_e \leq 4m, \\
\implies \tilde{c}_e, 4mc_e t \in [4m \lfloor \frac{c_e}{t} \rfloor, 4m(\lfloor \frac{c_e}{t} \rfloor + 1)], \\
\implies |\tilde{c}_e - \frac{4mc_e}{t}| \leq 4m, \\
\implies \sum_{e} |\frac{4mc_e}{t} - c_e||k_e| \leq \sum_{e} |4m||k_e| \leq 4mn,
\]
but
\[\frac{4mc'k}{t} \leq \frac{4mc'k}{t} - \epsilon'k \leq \sum_{e} \frac{4mc_e}{t} - \tilde{c}_e||k_e|,
\]
thus, we have
\[
\frac{4mc'k}{t} \leq 4mn, \\
\implies \epsilon'k \leq nt.
\]
Clearly, $x^1 + \sum_{k \in K'} k$ satisfy the demand/supply constraints of $\mathcal{MCF}$. Since $\min\{x_{e_1}, x_{e_2}\} \leq x_1 + \sum_{k \in K'} k \leq \max\{x_{e_1}, x_{e_2}\}$ for all $e \in E$, and each $k \in K'$, we have that $x^1 + \sum_{k \in K} k$ is a feasible solution for $\mathcal{MCF}$. Since $x^3$ is the optimal solution of $\mathcal{MCF}$, $c^T x^3 \leq c^T x^1 + \sum_{k \in K} c^T k \leq c^T x^1 + |K'|nt$. Since $|K'| = |x_1^2 - x_1|$, we have $c^T x^3 - c^T x^1 \leq |x_1^2 - x_1|nt$.

**Corollary 7.3.** $c^T x^3 \leq (1 + \frac{\epsilon}{2m})c^T x^1$, for any $\epsilon \leq 2$.

**Proof.** By Lemma 7.2,
\[\frac{c^T x^3 - c^T x^1}{c^T x^1} \leq \frac{|x_1^2 - x_1^1|nt}{c^T x^1} \leq \frac{|x_1^2 - x_1^1|nt}{|x_1^2 - x_1^1||c_e|} = \frac{\epsilon}{4m} \quad \text{as} \quad t = \frac{c_e}{4mn}.
\]
Thus,
\[\frac{c^T x^3 - c^T x^1}{c^T x^3} \leq \frac{\epsilon}{4m},
\]
\[\implies c^T x^3 \leq \frac{1}{1 - \frac{\epsilon}{4m}} c^T x^1.
\]
\[\implies c^T x^3 \leq (1 + \frac{\epsilon}{2m})c^T x^1
\]
where the last inequality holds because $(1 - \frac{\epsilon}{2m})(1 + \frac{\epsilon}{2m}) = 1 + \frac{\epsilon}{2m} - \frac{\epsilon^2}{4m} \geq 1$, as $\frac{\epsilon}{2m} \leq 1$.

**7.3 A (1 + $\epsilon$) Approximation Scheme** Loosely speaking, Corollary 7.3 shows that $x^2$ at arc $e'$ is “near optimal”, since fixing the flow at arc $e'$ to $x_1^2$ helps us in finding a feasible solution of $\mathcal{MCF}$ which is close to optimal. This leads us to approximate algorithm AS($MCF, \epsilon$) (see Algorithm 4); at each iteration, it uses APRXMT (Algorithm 3), and iteratively fixes the flow at the arc with the largest cost.
Algorithm 4 AS($\mathcal{MCF}$, $\epsilon$)

1: Let $G = (V, E)$ be the underlying directed graph of $(\mathcal{MCF})$, $m = |E|$, and $n = |V|$.
2: while $(\mathcal{MCF})$ contains at least 1 arcs do
3: Run APRXMT $(\mathcal{MCF}, \epsilon)$, let $x^2$ be the solution returned.
4: Find $(i', j') = \arg \max_{e \in E} \{c_e\}$, modify $\mathcal{MCF}$ by fix the flow on arc $e'$ by $x^2_{i'}$, and change the demands/supply on nodes $i'$, $j'$ accordingly.
5: end while

Theorem 7.2. The expected number of operations for $\text{AS}(\mathcal{MCF}, \epsilon)$ to terminate is $O\left(\frac{m^2 n^3 \log n}{\epsilon^2}\right)$. Suppose $x^* \leq (1 + \epsilon) c^T x^1$, for any $\epsilon \leq 1$.

Proof. By Corollary 7.2, the expected number of operations for APRXMT$(\mathcal{MCF}, \epsilon)$ to terminates is $O\left(\frac{m^2 n^3 \log n}{\epsilon^2}\right)$. Since $\text{AS}(\mathcal{MCF}, \epsilon)$ calls the method APRXMT$(\mathcal{MCF}, \epsilon)$ $m$ times, the expected number of operations for $\text{AS}(\mathcal{MCF}, \epsilon)$ to terminate is $O\left(\frac{m^3 n^3 \log n}{\epsilon^2}\right)$. And by Corollary 7.3,

$$c^T x^* \leq (1 + \frac{\epsilon}{2m}) m c^T x^1 \leq \epsilon c^T x^1 \leq (1 + \epsilon) c^T x^1$$

where the last two inequalities follows using standard analysis arguments.

And thus, $\text{AS}(\mathcal{MCF}, \epsilon)$ is an $(1 + \epsilon)$ approximation scheme for $\mathcal{MCF}$ with polynomial running time.

8 Discussions

In this paper, we formulated belief propagation (BP) for $\mathcal{MCF}$, and proved that BP solves $\mathcal{MCF}$ exactly when the optimal solution is unique. This result generalizes the result from [6], and provides new insights for understanding BP as an optimization solver. Although the running time of BP for $\mathcal{MCF}$ is slower than other existing algorithms for $\mathcal{MCF}$, BP is rather a generic and flexible heuristic, and one should be able to modify it to solve other variants of network flow problems. One such example, the convex-cost flow problem, was analyzed in Section 5. Moreover, the distributed nature of BP can be taken advantage of using parallel computing.

This paper also presents the first fully-polynomial running time scheme for optimization problems using BP. Since the correctness of BP often rely on the optimization problem having a unique optimal solution, Corollary 7.1 can be used for modifying BP in general to work in the absence of a unique optimal solution. In the approximation scheme, the heuristic of fixing values on variables while running BP is commonly known as ‘decimation’ (see [14]). To the best of our knowledge, this is the first disciplined, provable instance of decimation procedure. The ‘decimation’ in our scheme is extremely conservative, and a natural question is if there exists a more aggressive ‘decimation’ heuristic which can improve the running time.

Acknowledgements

While working on this paper, D. Gamarnik was partially supported by NSF Project CMMI-0726733; D. Shah was supported in parts by NSF EMT Project CCF 0829893 and NSF CAREER Project CNS 0546590; and Y. Wei was partially supported NSERC PGS M. The authors would also like to thank the anonymous referees for the helpful comments.

References


