Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

A Scalable Message-Passing Algorithm for Supply Chain Formation

Toni Penya-Alba
Jesus Cerquides
Juan A. Rodriguez-Aguilar
IIIA - CSIC
Campus de la UAB, E-08193 Bellaterra, Spain
{tonipenya, cerquide, jar}@iiia.csic.es

Meritxell Vinyals
Department of Computer Science
University of Verona
Strada le Grazie, 15, Verona, Italy
meritxell.vinalssalgado@univr.it

Abstract

Supply Chain Formation (SCF) is the process of determining the participants in a supply chain, who will exchange what with whom, and the terms of the exchanges. Decentralized SCF appears as a highly intricate task because agents only possess local information and have limited knowledge about the capabilities of other agents. The decentralized SCF problem has been recently cast as an optimization problem that can be efficiently approximated using max-sum loopy belief propagation. Along this direction, in this paper we propose a novel encoding of the problem into a *binary* factor graph (containing only binary variables) as well as an alternative algorithm. We empirically show that our approach allows to significantly increase scalability, hence allowing to form supply chains in market scenarios with a large number of participants and high competition.

1 Introduction

Supply Chain Formation (SCF) is the process of determining the participants in a supply chain (SC), who will exchange what with whom, and the terms of the exchanges (Walsh and Wellman 2003). Today's companies are required to dynamically form and dissolve trading relationships at a speed and scale that can be unmanageable by humans, giving rise to the need for automated SCF.

Automating SCF poses an intricate coordination problem to firms that must simultaneously negotiate production relationships at multiple levels of the SC. This problem has been already tackled by the AI literature, mostly through auction-based approaches. Several contributions (Walsh *et al.* 2000; Collins *et al.* 2002; Cerquides *et al.* 2007) have addressed the problem by means of combinatorial auctions (CAs) that compute the optimal SC allocation in a centralized manner. Unfortunately, since even finding any feasible SC allocation is NP-Hard (Walsh and Wellman 2000; Fionda and Greco 2009), sufficiently large SCF problems will be intractable, hence hindering the scalability of the global optimization performed by auction-based approaches. Furthermore, as argued in (Walsh and Wellman 2003), even when the computation is tractable, no single entity may have global allocative authority to compute allocations over the entire SC.

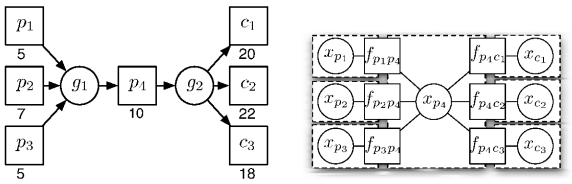
The work in (Walsh and Wellman 2003) proposes to solve the SCF problem in a fully decentralized manner. Each good in the SC is auctioned separately and all auctions run simultaneously without direct coordination. Therefore, each auction allocates a single resource considering the offers to buy or sell submitted by agents. More recently, Winsper et al. (Winsper and Chli 2010) have cast the decentralized SCF problem as an optimization problem that can be approximated using (max-sum) loopy belief propagation (Farinelli *et al.* 2008). We shall refer to Winsper's approach as LBP. Winsper et al. empirically show that LBP is able to produce more efficient SCs than (Walsh and Wellman 2003). Nonetheless, the problem representation employed by LBP leads to exponential memory and communication requirements that largely hinder its scalability. Furthermore, the larger the number of agents, the further the SC assessed by LBP is from the optimal one.

Against this background, in this paper we propose a novel approach to the decentralized SCF problem, the so-called Reduced Binary Loopy Belief Propagation (RB-LBP), which significantly outperforms LBP in terms of scalability. RB-LBP allows agents to form SCs in a bottom-up manner, requiring only local communication and limited knowledge of other participants. The main contributions of RB-LBP are: (i) a novel encoding of the SCF problem into a binary factor graph (containing only binary variables); and (ii) a derivation of simplified messages that dramatically lowers the communication requirements of message passing.

We show that in the worst case memory and communication requirements of RB-LBP scale linearly, whereas LBP scales exponentially. Furthermore, we empirically compare RB-LBP versus LBP. We observe that as competition in the market increases:[1] (i) RB-LBP can save several orders of magnitude in terms of memory and communication with respect to LBP; (ii) the value of the SCs assessed by RB-LBP can be up to 2 times higher than those assessed by LBP.

The paper is organized as follows. Section 2 introduces LBP from (Winsper and Chli 2010) and analyzes its memory and communication requirements. Section 3 describes RB-LBP, our main contribution. Section 4 details the results after comparing LBP with RB-LBP, and section 5 draws conclusions and sets paths for future research.

[1] In terms of number of providers offering each good.



(a) TDN (b) SC factor graph

Figure 1: Example of a TDN and its LBP encoding.

x_{p_1} [f_{p_1}]	x_{p_4} [f_{p_4}]	x_{c_1} [f_{c_1}]
σ_0: sell to p_4 [-5]	σ_2:buy from p_1, sell to c_1 [-10]	σ_{12}:buy from p_4 [20]
σ_1: don't sell [0]	σ_3: buy from p_1, sell to c_2 [-10]	σ_{13}: don't buy [0]
	σ_4: buy from p_1, sell to c_3 [-10]	
	σ_5: buy from p_2, sell to c_1 [-10]	
	σ_6: buy from p_2, sell to c_2 [-10]	
	σ_7: buy from p_2, sell to c_3 [-10]	
	σ_8: buy from p_3, sell to c_1 [-10]	
	σ_9: buy from p_3, sell to c_2 [-10]	
	σ_{10}: buy from p_3, sell to c_3 [-10]	
	σ_{11}: don't buy, don't sell [0]	

Table 1: Example of states (and values) of agent variables.

2 A Base Approach to Decentralized SCF

To automate SCF, Walsh and Wellman introduce the notion of Task Dependency Network (TDN) as a way of capturing complementarities among production processes. A TDN takes the form of a bipartite directed acyclic graph. Figure 1a depicts an example of a TDN. Producers (p_1, p_2, p_3, p_4) and consumers (c_1, c_2, c_3) are represented by rectangles, goods (g_1, g_2) are represented by circles, and links between rectangles represent potential flows of goods. Producers p_1, p_2, p_3 do not require any input goods to produce each one unit of output good g_1, whereas producer p_4 requires g_1 to produce g_2. Good g_2 can be consumed by either c_1, c_2, or c_3.[2] Producers p_1, p_2, p_3 and consumers c_1, c_2, c_3 are *potential partners* of p_4 as they are all eligible to establish trading relationships. The number below each producer stands for the selling price of her service, whereas the number below each consumer stands for the purchasing price.

The TDN in figure 1a allows several feasible SC configurations. For instance, configuration $SC_1 : p_1 \rightarrow p_4 \rightarrow c_2$ is feasible, whereas $SC_2 : p_4 \rightarrow c_2$ is not (nobody provides g_1 to p_4). The value of a configuration is assessed by subtracting its producers' values from its consumers' values. The value of SC_1 is $22 - 10 - 5 = 7$. In general, the value of a configuration SC is $\sum_{c_i \in SC} v(c_i) - \sum_{p_j \in SC} v(p_j)$, where $v(c_i)$ and $v(p_j)$ stand for the i-th consumer and j-th producer values respectively. The SCF problem is that of finding the feasible configuration with maximum value.

2.1 Using Loopy Belief Propagation

The work in (Winsper and Chli 2010) shows that the SCF problem can be cast as an optimization problem that can be efficiently approximated using max-sum. The formalism employed to represent SC problems is inspired on TDNs (Walsh and Wellman 2003). Thus, (Winsper and Chli 2010) offers the means of converting a TDN into a graphical model, and concretely into a factor graph on which max-sum can operate. Figure 1 depicts the conversion of a TDN into a factor graph that we detail next.

In the factor graph there is a dashed box per agent in the TDN and the goods do not appear. Inside an agent's box there is a variable, represented by a circle. For instance, the box for p_1 contains variable x_{p_1}. The values (states) of each variable encode the individual decisions that the agent needs to make regarding her exchange relationships plus an inactive state. For instance, table 1 lists the states of the variables corresponding to agents p_1, p_4, and c_1. Notice that the states of p_4 (from σ_2 to σ_{11}) encode all possible exchange relationships for the agent.

In the factor graph variables corresponding to potential partners are connected through a **compatibility factor** (represented as a square). Each compatibility factor encodes the compatibility between the decisions of the two agents involved. Two agents decisions are incompatible whenever one of them is willing to trade with the other, but the other does not. Consider agent variable x_{p_4}, its state σ_2 is compatible with x_{p_1}'s state σ_0, but it is incompatible with x_{p_1}'s σ_1 (p_1 doesn't provide g_1 to p_4!). If two states are compatible, the value of the pairwise function is zero, otherwise is negative infinity. Thus, considering p_1 and p_4, $f_{p_1p_4}(\sigma_0, \sigma_2) = 0$ and $f_{p_1p_4}(\sigma_1, \sigma_2) = -\infty$. Note that to determine their compatibility, both p_1 and p_4 need to keep a copy of compatibility factor $f_{p_1p_4}$.

The factor graph resulting from converting a TDN also contains **activation factors** (omitted in figure1b), each one representing the purchasing or selling price of an agent. Following table 1, p_1 assigns value -5 to its state σ_0, and c_1 assigns value 20 to its state σ_{12}.

To summarize, the conversion process transforms a TDN into an SC factor graph with agent variables $V = \{x_1, \ldots, x_N\}$, activation factors $\{f_1, \ldots, f_N\}$, and a set F of compatibility factors. Then, the SCF problem amounts to finding a state assignment for the agent variables in V that maximizes the following reward function:

$$\mathcal{R}(x_1, \ldots, x_N) = \sum_{x_i \in V} f_i(x_i) + \sum_{f_{kl} \in F} f_{kl}(x_k, x_l) \qquad (1)$$

Notice that $\mathcal{R}(x_1, \ldots, x_N)$ assesses the value of an SC configuration. The optimization of $\mathcal{R}$ can be approximated by running max-sum over an SC factor graph. LBP (Winsper and Chli 2010) has SC participants iteratively exchange messages encoding their trading decisions along with their values. Message-passing occurs in a decentralized manner and agents only exchange messages with their potential partners. This process continues until all agent variables settle on some state. At that point, the states of agent variables represent a solution to the SCF problem.

In max-sum, messages in a factor graph flow from variable to factor and from factor to variable. Max-sum assesses

[2] In (Walsh and Wellman 2003), producers can only produce a single unit of a single type of output good.

the message from variable x to factor f ($\mu'_{x\to f}$) as follows:

$$\mu'_{x\to f}(\bar{x}) = \sum_{g\in\mathcal{N}(x)\setminus\{f\}} \mu_{g\to x}(\bar{x}) \quad (2)$$

where $\mathcal{N}(x)$ stands for the neighboring factors of variable x, $\bar{x}$ stands for a state of x, and $\mu_{g\to x}$ stands for the last message received by variable x from factor g. Max-sum assesses the message from f to x ($\mu'_{f\to x}$) as follows:

$$\mu'_{f\to x}(\bar{x}) = \max_{\bar{Y}}(f(\bar{x},\bar{Y}) + \sum_{y\in Y}\mu_{y\to f}(\bar{y})) \quad (3)$$

where Y is the set of variables linked to factor f excluding x, $\bar{Y}$ is the joint state for all the variables in Y including y.

Every time a variable receives messages from all its neighboring factors, it can locally update the value of each of its states given the messages received so far:

$$\tilde{\mathcal{R}}_x(\bar{x}) = \sum_{f\in\mathcal{N}(x)} \mu_{f\to x}(\bar{x}) \quad (4)$$

Equation 4 allows each agent to periodically obtain an approximation to the function to optimize in equation 1. Thus, the highest-value state corresponds to the SC configuration with maximum value. Convergence of LBP occurs when all agents find that the state that maximizes their $\tilde{\mathcal{R}}_x$ is the same as in the previous iteration. Upon convergence, LBP includes a post-processing phase that removes incompatibilities from the computed SC configuration.

2.2 Analysis

The example in figure 1b makes us wonder about LBP's memory and communication requirements. Notice that agent variable x_{p_4} requires 3^2+1 states and each compatibility factor requires $2\cdot(3^2+1)$ entries (the product of the number of states of the two agents). If agent p_4 had another input good provided by three other agents, x_{p_4} would require 3^3+1 states and factors with $2\cdot(3^3+1)$ entries. In general, the memory requirements of an SC factor graph exponentially grow with the number of goods and neighboring agents. Notice that equations 2 and 3 indicate that the size of messages between agent variable and factor is as large as the number of states in the agent variable. In figure 1b, LBP would employ messages of size 3^2+1 from agent variable x_{p_4} to its compatibility factors.

From this discussion follows that in markets with high degrees of competition, the SC factor graph resulting from converting a TDN is highly demanding in terms of memory and communication requirements. Next, we assess some upper bounds on the amount of memory required by an SC factor graph along with the overall bandwidth required by LBP at each iteration. We assume that there are n agents, each agent is connected to at most G goods, and each good is connected to at most A agents. Hence, an agent has at most $G\cdot A$ potential partners. Therefore, the requirements are:

Memory. Each compatibility factor requires at most A^{2G} entries to store compatibility values. Since each agent shares compatibility factors with $G\cdot A$ neighbors, the memory each agent requires is $\mathcal{O}(G\cdot A^{2G+1})$.

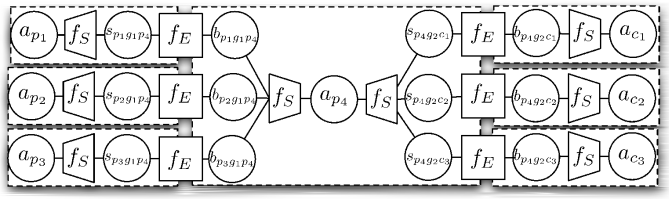


Figure 2: RB-LBP binary factor graph of the TDN in Fig. 1a.

Communication. The messages between an agent variable and its compatibility factor are of size $\mathcal{O}(A^G)$. Since each agent shares compatibility factors with at most $G\cdot A$ neighbors, she consumes $\mathcal{O}(G\cdot A^{G+1})$ bandwidth. Finally, since we consider n agents in the SC, LBP requires $\mathcal{O}(n\cdot G\cdot A^{G+1})$ bandwidth overall per iteration.

Therefore, the exponential resource requirements of LBP, particularly in markets with high degrees of competition, significantly hinders its scalability. Furthermore, regarding privacy, the message each agent receives from a trading partner also contains information about her competitors. For example, in figure 1b producer p_4 would send a message to p_1 that contains all her states and thus p_1 would be aware of the existence of other producers of good g_1.

3 Binarized Belief Propagation for SCF

We have argued that in markets with high competition, the mapping to a factor graph provided in (Winsper and Chli 2010) requires an overlarge amount of memory. In this section we develop RB-LBP (first introduced in (Penya-Alba *et al.* 2012)), an alternative algorithm that scales up to high competition markets. We start by introducing a new mapping of a TDN into an **SC binary factor graph** (containing only binary variables). Thereafter, we show that variables, factors and messages in an SC binary factor graph take a simpler form than their counterparts in an SC factor graph. Furthermore, we describe RB-LBP's post-processing phase to remove incompatibilities from the computed SC configuration. Finally, we provide a complete description of RB-LBP and we analyze its worst-case requirements.

3.1 From TDN to SC Binary Factor Graph

Figure 2 shows the SC binary factor graph that results from encoding the TDN in figure 1a. In our encoding each agent is responsible for several decision variables. In figure 2, each dashed box surrounds the variables and the factors that an agent is responsible for. This new mapping introduces two types of binary decision variables:

Activation variables a_α encoding whether agent α is active ($a_\alpha=1$) or inactive ($a_\alpha=0$), namely part of the SC or not.

Option variables $b_{\alpha g\beta}$ **and** $s_{\alpha g\beta}$. $b_{\alpha g\beta}$ encodes agent's β decision of buying good g from agent α, whereas $s_{\alpha g\beta}$ encodes agent's α decision to sell good g to agent β.

For example, consider the variables p_4 is responsible for. The decision of p_4 to participate in the SC is encoded as variable a_{p_4}. Furthermore, p_4 can acquire good g_1 either from p_1, p_2 or p_3. Therefore, p_4 requires variables $b_{p_1g_1p_4}$, $b_{p_2g_1p_4}$

and $b_{p_3g_1p_4}$ to encode each of the three possible decisions. If p_4 is inactive ($a_{p_4} = 0$), she should not buy g_1 and so $b_{p_1g_1p_4}$, $b_{p_2g_1p_4}$ and $b_{p_3g_1p_4}$ should all be 0. Furthermore, whenever p_4 is active, she should buy g_1 from only one of her providers, that is, only one out of $b_{p_1g_1p_4}$, $b_{p_2g_1p_4}$ and $b_{p_3g_1p_4}$ should be 1. The information about whether a set of values is acceptable is stored in factor f_S linking a_{p_4} with option variables $b_{p_1g_1p_4}$, $b_{p_2g_1p_4}$, $b_{p_3g_1p_4}$. Since this factor guarantees that only one of the providers is selected, we call it **selection factor**. In general, to guarantee that only one of the providers of a given good is selected, we make use of a selection factor. A selection factor links the activation variable from the agent with the different choices for that good. Note that in a selection factor, the role of the activation variable is different from that of option variables.

We do also need to guarantee that different agents take coherent decisions. Thus, we add a factor that constrains $s_{\alpha g\beta}$ and $b_{\alpha g\beta}$ to be either both 1 or both 0. We call this kind of factor **equality factor**. Equality and selection factors encode hard constraints. Whenever a variable assignment satisfies a hard constraint, its value is 0, otherwise it is negative infinity.

Finally, we introduce agents' buying and selling prices into the SC binary factor graph by means of **activation factors** (not included in figure 2). Thus, each agent α has a factor f_α that stores: (i) zero whenever a_α is 0; and (ii) agent α's buying or selling price otherwise.

From the example, we are ready to specify the steps needed to transform a TDN into an SC binary factor graph.

Conversion Algorithm: Each agent α performs:

1. **Variable creation.**
   (a) She creates a variable a_α encoding whether agent α is active or not.
   (b) For each good g and agent β such that agent α offers good g to agent β, she creates a variable $s_{\alpha g\beta}$ encoding agent α's decision to sell good g to agent β .
   (c) For each good g and agent β such that agent α is offered good g by agent β, she creates a variable $b_{\beta g\alpha}$ encoding agent α's decision to buy good g from agent β.
2. **Activation factors creation.** She creates activation factor f_α to store: (i) zero whenever a_α is 0; and (ii) agent α's buying or selling price otherwise.
3. **Selection factors creation.**
   (a) For each good g that α can buy, she creates selection factor $f_{\alpha g}$ with: (i) a_α as activation variable; and (ii) an option variable $b_{\beta g\alpha}$ per provider β of good g.
   (b) For each each good g that α can sell, she creates selection factor $f_{\alpha g}$ with: (i) a_α as activation variable; and (ii) an option variable $s_{\alpha g\beta}$ for each buyer β of good g.
4. **Equality factors creation.** For each good g and each agent β such that agent α is offering good g to agent β, she creates an equality factor linking $s_{\alpha g\beta}$ and $b_{\alpha g\beta}$.

3.2 Simplifying Message-Passing

The algorithm above produces a SC binary factor graph on which max-sum can run. However, next we show that the memory and communication requirements for max-sum over SC binary factor graphs can be severely reduced. First, we argue that there is no need to store constraint factors because we can provide explicit equations for their messages. Secondly, we argue that since we employ max-sum, we can represent any function over a binary variable using a single real number. Based on these observations, we provide the reduced equations for the messages that RB-LBP will employ (refer to (Penya-Alba 2012) for complete details of the derivation).

Simplifying factor to variable messages. Max-sum iterates exchanging messages from factors to variables (equation 2) and from variables to factors (equation 3). Since variables are binary, computing the expression in equation 2 is straightforward. On the other hand, the right hand side of equation 3 assesses the maximum over a potentially large set of variables. Note that both equality and selection factors represent hard constraints, whose value is either zero or minus infinity. Hence, we can assess the maximum by only considering those assignments satisfying the hard constraint (namely those with zero value). This severely simplifies the assessment of equation 3.

Reducing messages. Observe that a single value $\delta = b(1) - b(0)$ is enough to represent a function b over a binary variable. Thus, $\delta > 0$ means $x = 1$ is preferred to $x = 0$, $\delta < 0$ means $x = 0$ is preferred to $x = 1$, and $\delta = 0$ represents no preference. The same idea can be applied to messages. Whenever RB-LBP needs to send a message μ to a binary variable, instead of sending $\langle \mu(0), \mu(1) \rangle$, it can send a single value $\nu = \mu(1) - \mu(0)$. To take full advantage of this idea, we assess the single-valued messages that are exchanged on an SC binary factor graph.

After compiling constraint factors into equations and reducing messages, the messages employed by RB-LBP can be computed as follows:

1. The message **from variable** x **to factor** f contains a single value $\nu'_{x\to f}$, the addition of the last messages received from every factor linked to x but f.

$$\nu'_{x\to f} = \sum_{f'\in\mathcal{N}(x)\setminus\{f\}} \nu_{f'\to x} \tag{5}$$

2. The message $\nu'_{f_S\to a}$ sent **from selection factor** f_S to its **activation variable** a is the largest message received by f_S from any of its option variables $o_i, 1 \le i \le n$.

$$\nu'_{f_S\to a} = \max_{1\le i\le n} (\nu_{o_i\to f_S} + \varepsilon_{o_i}) \tag{6}$$

3. The message $\nu'_{f_S\to o_i}$ sent from **selection factor** f_S **to option variable** o_i is assessed as:

$$\nu'_{f_S\to o_i} = \varepsilon_{o_i} + \min(\nu_{a\to f_S}, \min_{1\le j\ne i\le n} -(\nu_{o_j\to f_S} + \varepsilon_{o_j})) \tag{7}$$

4. The message $\nu'_{f_\alpha\to a_\alpha}$ sent from **activation factor** f_α **to activation variable** a_α is simply the agent's buying or selling price.

5. The message **from equality factor** f_E joining variables s and b **to variable** b is a single value $\nu'_{f_E\to b}$ assessed as the last message received by f_E from s. Since equality factors are symmetric, the message from f_E to variable s is the last message received from b.

$$\nu'_{f_E \to b} = \nu_{s \to f_E} \qquad \nu'_{f_E \to s} = \nu_{b \to f_E} \tag{8}$$

Note that to assess the messages from factors to variables (using equations 5-7) agents do not need to store the tables representing constraint factors. Since the size of a selection factor grows exponentially with the number of options, the equations above yield large savings in memory.

Finally, notice that equations 6 and 7 include epsilon values. These are included to help RB-LBP break ties. Consider the example in figure 1a. There, p_4 has two possible providers, p_1 and p_3, selling good g_1 at the very same price. When running RB-LBP over the SC binary factor graph in figure 2, option variables $b_{p_1 g_1 p_4}$ and $b_{p_3 g_1 p_4}$ can be set to 1 to reflect that p_4 equally values buying from either p_1 or p_3. However,that would break the selection (hard) constraint linking a_{p_4} with its option variables. To avoid this problem, each agent assigns an economically negligible value to each of her trading relationships encoding the agent's preferences in case of equally-valued options. Thus, for each option variable o_i, we draw a value ε_{o_i} from a uniform distribution $U[-0.00005, 0.00005]$.

3.3 Determining the SC Configuration

The RB-LBP algorithm distributedly runs max-sum using the above-described equations until convergence or until reaching a maximum number of steps. After that, each agent stores a value for each variable representing her preferences. A first solution is assessed distributedly, each agent setting each variable to her preferred value. Since this first solution may not satisfy some of the constraints, it needs to be revised (and possibly mended). The mending process has two main steps: during the first one, each agent ensures that all of her internal constraints are satisfied; during the second one, agents ensure that providers and consumers agree on their decision to collaborate.

Step 1: Internal consistency. Each agent checks each of her selection constraints. When a selection constraint breaks, there are three possible cases:

1. If the activation variable is 0, the agent clears all the option variables.
2. If the activation variable is 1 and there are no option variables set to 1, the agent toggles the activation variable
3. If the activation variable is 1 and there are two or more option variables set to 1, the agent randomly selects one of them and clears the remaining ones.

Step 2: Collaboration consistency. Agents need to communicate their neighbors to agree on whether collaborating or not. Firstly, each agent sends to each of her neighbors her decision to collaborate with her or not. Secondly, each active agent (her activation variable is 1) will determine (based on the information received in the previous step) whether all of her selected partners want to collaborate with her. If that is not the case, she will set all of her variables to 0 and will send this information to her neighbors. This second step is repeated until no agent changes her variables any further.

Measure	LBP	RB-LBP
Memory needed per agent to store the preferences over her state	$\mathcal{O}(A^G)$	$\mathcal{O}(G \cdot A)$
Size of largest factor	$\mathcal{O}(A^{2G})$	$\mathcal{O}(1)$
Maximum memory needed per agent (to store both preferences and factors)	$\boldsymbol{\mathcal{O}(G \cdot A^{2G+1})}$	$\boldsymbol{\mathcal{O}(G \cdot A)}$
Maximum message size	$\mathcal{O}(A^G)$	$\mathcal{O}(1)$
Maximum bandwith consumed per agent	$\boldsymbol{\mathcal{O}(G \cdot A^{G+1})}$	$\boldsymbol{\mathcal{O}(G \cdot A)}$
Overall consumed bandwith	$\mathcal{O}(n \cdot G \cdot A^{G+1})$	$\mathcal{O}(n \cdot G \cdot A)$
Maximum computation time per agent and iteration	$\boldsymbol{\mathcal{O}(G \cdot A^{2G+1})}$	$\boldsymbol{\mathcal{O}(G \cdot A^2)}$

Table 2: Required resources: LBP vs. RB-LBP.

3.4 Reduced Binary Loopy Belief Propagation

Putting together all the elements above, we obtain RB-LBP as a distributed algorithm that proceeds as follows:

1. Each agent stores the variables defined at step 1 of the procedure in section 3.1.
2. Each agent randomly assigns a weight ε_{o_i} to each of her variables, except the activation variable.
3. Until convergence or reaching the maximum number of iterations each agent:
   (a) Internally uses messages in equations 5, 6 and 7 between the variables and factors she is responsible for.
   (b) Uses the messages in equations 8 to communicate with her potential partners.
4. The solution is determined as described in section 3.3.

3.5 Analysis

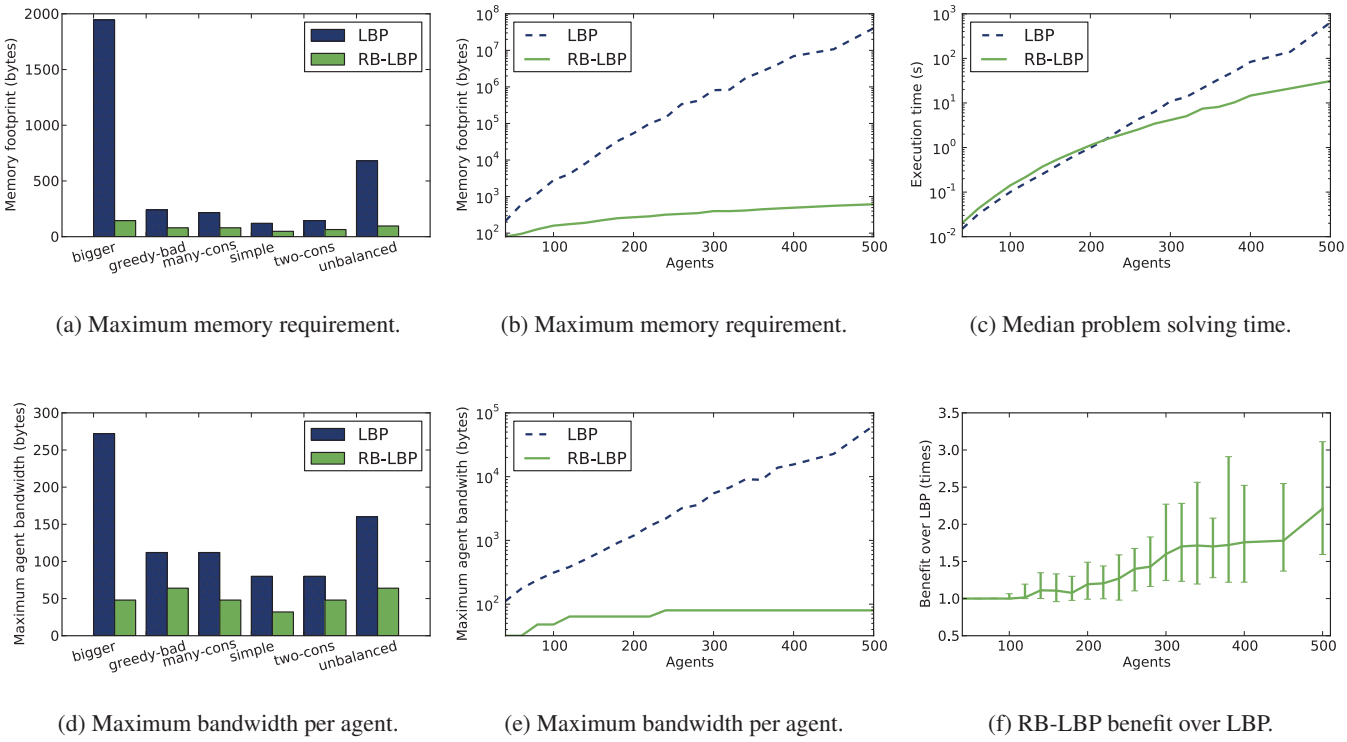
Along the analysis conducted in section 2.2 for LBP, in this section we provide worst-case bounds on the amount of memory, the size of messages exchanged at each iteration, and the computation time needed by RB-LBP agents.

Memory requirements. Each agent needs to store a real number per variable in order to maintain preferences over a variable's states. Since each agent collaborates with at most $G \cdot A$ other agents, the amount of memory to store her preferences is in $\mathcal{O}(G \cdot A)$. Regarding factors, each RB-LBP agent only needs to store her activation factor. Hence, the memory that each RB-LBP agent needs is in $\mathcal{O}(G \cdot A)$.

Communication requirements. Each pair of agents involved in a potential collaboration in RB-LBP are linked by a single equality constraint. Hence, they exchange a single number. Since each agent collaborates with at most $G \cdot A$ other agents her communication requirements are in $\mathcal{O}(G \cdot A)$. Thus, the total bandwidth consumed by all RB-LBP agents is in $\mathcal{O}(n \cdot G \cdot A)$.

Computation time requirements. At each iteration, each agent needs to assess $\mathcal{O}(G \cdot A)$ messages and by looking at the equations of the messages we see that each message takes at most $\mathcal{O}(A)$ operations. Thus, the total computation time required by RB-LBP agents is in $\mathcal{O}(G \cdot A^2)$.

Privacy analysis. Unlike LBP, the only information that an agent in RB-LBP requires regarding her potential partners is limited to their trading relationship. Thus, an agent only



(a) Maximum memory requirement. (b) Maximum memory requirement. (c) Median problem solving time.

(d) Maximum bandwidth per agent. (e) Maximum bandwidth per agent. (f) RB-LBP benefit over LBP.

Figure 3: RB-LBP vs. LBP in Walsh's (left) and large-scale (right) networks. Plots b, c and e use a log-scale for the y axis.

shares with a potential partner her own decision about buying or selling a good. For instance, in figure 1a agent p_4 would only share with p_1 her decision about purchasing good g_1, but not any information about p_1's competitors. Hence, the gain of privacy of RB-LBP with respect to LBP.

Table 2 compares the resources needed by the LBP and RB-LBP algorithms, showing that RB-LBP scales better in terms of memory, communication and computation requirements.

4 Empirical Results

In this section we empirically compare the memory, communication, computation time and the value of the SC assessed by LBP with respect to those obtained by RB-LBP.

We perform our comparison over two different sets of problems. Firstly, following the experiments in (Winsper and Chli 2010), we analyze the performance of LBP and RB-LBP over the TDNs described in (Walsh and Wellman 2003). The experimental settings are the same as those in (Winsper and Chli 2010).

Secondly, we compare LBP and RB-LBP in scenarios with a larger number of agents and higher degrees of competition. Since the test-suite described in (Vinyals *et al.* 2008) is designed specifically to mimic real-world SCF problems, we employ it to generate the TDNs used in this comparison. We generate TDNs with 50 goods and a number of agents ranging from 40 to 500. Since the number of goods is fixed across scenarios, the degree of competition in the TDNs grows with the number of agents.

For each scenario, we generate 100 TDNs with different selling and buying prices. We run LBP and RB-LBP over each TDN and we record the maximum memory required per agent, the maximum bandwidth used per agent at each iteration, the running time, and the value of the SC configurations assessed. Since the distributions obtained for these measures are long-tailed and skewed, we use the median instead of the mean as a measure of central tendency following the recommendations in (Wilcox and Keselman 2003).

LBP and RB-LBP are implemented using libDAI (Mooij 2010). Our tests are run on an Intel(R) Xeon(TM) CPU running at 3.20GHz with 2GB of RAM on linux-2.6_x86_64.

The results over the TDNs introduced in (Walsh and Wellman 2003) are depicted in figures 3a and 3d. Figure 3a shows that LBP requires from 2 up to 13 times more memory than RB-LBP depending on the TDN structure. Figure 3d shows that the maximum bandwidth consumed by an agent during an LBP iteration is up to 5 times larger than RB-LBP's. Note that the names in the x axis in figures 3a and 3d correspond to the network structures described in (Walsh and Wellman 2003). The average values of the SC configurations obtained by LBP match the results in (Winsper and Chli 2010). RB-LBP's average SC values are identical to LBP's. Since the largest TDN contains at most 33 agents, we can consider the TDNs in (Walsh and Wellman 2003) as small. Thereupon, both LBP and RB-LBP converge to an SC configuration in

the order of the millisecond, making the difference between both methods negligible in any practical sense.

Figures 3b to 3f show the results of the comparison of LBP and RB-LBP in TDNs with a larger number of agents and larger degrees of competition. Observe, in figure 3b, that the memory requirements for LBP are up to 5 orders of magnitude (10^5 times) greater than for RB-LBP. This is because as the number of agents increases memory requirements grow exponentially for LBP, and only linearly for RB-LBP. Bandwidth usage for LBP is up to 787 times greater than for RB-LBP as shown in figure 3e. Regarding computational time, RB-LBP is up to 20 times faster than LBP, as shown in figure 3c. Finally, we observe that the median SC value obtained by RB-LBP is up to 2 times greater than those obtained by LBP as shown in figure 3f.

5 Conclusions and Future Work

We have described RB-LBP, a novel approach for decentralized SCF. We have shown both theoretically and experimentally that RB-LBP scales up to solve market scenarios with a large number of participants and high competition. RB-LBP can significantly reduce the usage of memory and communication several orders of magnitude with respect to LBP. Furthermore, RB-LBP produces up to two times higher value SCs and has smaller time complexity. Therefore, RB-LBP allows to tackle large-scale decentralized SCF.

Up to date state-of-the-art decentralized SCF (Walsh and Wellman 2003; Winsper and Chli 2010) only applies to TDNs whose agents can produce at most a single good. Although we report empirical results over TDNs to compare with existing approaches, RB-LBP can readily be applied to supply chains whose producers can deliver more than one good. Evaluating RB-LBP in such scenarios and over a variety of actual-world network structures is left as future work.

6 Acknowledgments

This work has been funded by projects EVE (TIN2009-14702-C02-01), AT (CSD2007-0022), CSIC 201050I008, and the Generalitat of Catalunya (2009-SGR-1434).

References

Jesus Cerquides, Ulle Endriss, Andrea Giovannucci, and Juan A Rodriguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *IJCAI*, pages 1221–1226, 2007.

John Collins, Wolfgang Ketter, and Maria Gini. A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *Int. J. Electron. Commerce*, 7:35–57, 2002.

Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, pages 639–646, 2008.

Valeria Fionda and Gianluigi Greco. Charting the tractability frontier of mixed multi-unit combinatorial auctions. In *IJCAI*, pages 134–139, 2009.

Joris M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 11:2169–2173, 2010.

Toni Penya-Alba, Meritxell Vinyals, Jesus Cerquides, and Juan A. Rodriguez-Aguilar. Scalable decentralized supply chain formation through binarized belief propagation. In *Proceedings of AAMAS 2012*.

Toni Penya-Alba. Scalable decentralized supply chain formation through binarized belief propagation. Master's thesis, Universitat Politècnica de Catalunya. Departament de Llenguatges i Sistemes Informàtics, 2012.

Meritxell Vinyals, Andrea Giovannucci, Jesus Cerquides, Pedro Meseguer, and Juan A Rodriguez-Aguilar. A test suite for the evaluation of mixed multi-unit combinatorial auctions. *Journal of Algorithms*, 63(1-3):130–150, 2008.

William E. Walsh and Michael P. Wellman. Marketsat: An extremely decentralized (but really slow) algorithm for propositional satisfiability. In *AAAI/IAAI*, pages 303–309, 2000.

William E. Walsh and Michael P. Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Journal of Artificial Intelligence Research (JAIR)*, 19:513–567, 2003.

William E. Walsh, Michael P. Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *ACM Conference on Electronic Commerce*, pages 260–269, 2000.

Rand R Wilcox and H J Keselman. Modern robust data analysis methods: measures of central tendency. *Psychological methods*, 8(3):254–74, 2003.

Michael Winsper and Maria Chli. Decentralised supply chain formation: A belief propagation-based approach. In *ECAI*, pages 1125–1126, 2010.