Exploiting Max-Sum for the Decentralized Assembly of High-Valued Supply Chains

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
ECS, Faculty of Physical and Applied Sciences
University of Southampton
Southampton, United Kingdom. SO17 1BJ
mv2y11@ecs.soton.ac.uk

ABSTRACT

Supply Chain Formation involves determining the participants and the exchange of goods within a production network. Today's companies operate autonomously, making local decisions, and coordinating with other companies to buy and sell goods along their Supply Chains. Such temporal interactions need to be formed rapidly and in a decentralized manner. For sufficiently large problems, current state-of-the-art approaches for Decentralized Supply Chain Formation are only capable of either (i) producing Supply Chains of high value at the expense of high resource requirements; or (ii) require low resources at the expense of producing Supply Chains of low value. In this paper we describe an algorithm that is able to produce Supply Chains of high value while keeping a low resource usage profile. Moreover, our method is able to produce near optimal Supply Chains while using up to four orders of magnitude less resources that the state-of-the-art.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Performance, Experimentation

Keywords

Teamwork, coordination, distributed problem solving

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 [15]. 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. Initial contributions [16, 5, 4] addressed the problem by means of combinatorial auctions that compute the optimal SC allocation in a centralized manner.

The decentralized assessment of SCs has been studied in a number of works [15, 10, 21, 3]. Companies taking part in the SCF operate autonomously making local decisions and coordinating with each other, making SCF an inherently distributed process. Therefore, no single entity may have global allocative authority over the entire SC. Moreover, since the SCF problem is NP-Complete [17], sufficiently large SCF problems will be intractable, hence hindering the scalability of the global optimization performed by centralized, auction-based approaches.

In [15], Walsh et al. proposed 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 good considering the offers to buy or sell submitted by agents. Nevertheless, the approach proposed by Walsh et al. suffers from high communication requirements, as discussed in [14]. Later on, Burke et. al. [3], cast the SCF problem into a Distributed Constraint Optimization (DCOP) problem. Unfortunately, their approach suffers from scalability issues.

More recently, Winsper et al. [21] cast the decentralized SCF problem as an optimization problem that can be approximated using (max-sum) loopy belief propagation [6]. Nonetheless, as shown in [10], the problem representation employed by Winsper et al. leads to exponential memory and communication requirements that largely hinder its scalability. Thus, Penya-Alba et al. provide in [10] a scalable approach to the decentralized SCF problem through a new encoding of the SCF problem into a binary factor graph. However, as we show in this paper, the algorithm in [10] is unable to find SCs whose value is close to the optimal (i.e. the one with maximal value) as the number of agents at trade increases.

To summarize, state-of-the-art approaches fall into two categories: those that provide higher-valued SCs at the expense of higher resource (memory, bandwidth, computation) requirements; and those that reduce the resources required at the expense of obtaining lower-valued SCs. Against this background, in this paper we present CHAINME, a novel decentralized SCF algorithm. Our main contributions are: (i) a novel encoding of the SCF problem into an unconstrained binary linear program; (ii) an efficient max-sum implementation optimized to solve the unconstrained binary linear

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

program with significantly less resources; and (iii) an empirical evaluation of CHAINME, showing that CHAINME assesses SCs with higher value while requiring from one up to four orders of magnitude less resources than the state-of-the-art methods.

The paper is organized as follows. Section 2 describes the SCF problem. Section 3 reviews the max-sum algorithm. Section 4 reviews the state-of-the-art approaches to decentralized SCF. Section 5 details CHAINME's encoding of the SCF problem into an unconstrained binary linear program. Section 6 describes the CHAINME algorithm. Section 7 benchmarks CHAINME against previous algorithms for decentralized SCF, and Section 8 draws conclusions and sets paths to future research.

2. THE SUPPLY CHAIN FORMATION PROBLEM

In this section we describe the SCF problem. Before describing the formal details, we illustrate an SC with an example from a local lime juice industry. In our example, there are three lime producers (Alice, Bob, and Carol), each of them can produce a kilo of limes at a particular cost: Alice and Carol ask for \$5 each, whereas Bob asks for \$7. Then we have Dave, the lime squeezer, who given a kilo of limes can produce a liter of juice for \$10. Thus, Dave acts as a buyer of limes and as a seller of juice. Finally, there are three juice buyers (Eve, Frank, and Gaby), each aiming at buying a liter of juice at a given price (Eve offers \$20, Frank \$22, and Gaby \$18).

Figure 1 depicts the Task Dependency Network (TDN)[15] representing the scenario described above. In a TDN, participants (lime producers, squeezers and juice consumers) are represented by rectangles, goods (Lime and Juice) are represented by circles, and links between participants and goods represent potential flows of goods. The number below each participant p_i represents her activation cost (C_{p_i}). The activation cost of a participant can either be a positive value (the participant is willing to pay in order to be part of the SC) or a negative value (the participant requests to be paid to be part of the SC). Notice that each participant produces only one unit of each of her output goods, thus potentially limiting the amount of goods in the SC. In the example, the juice is a limited resource since Dave can only produce one liter of it. Furthermore, SCs may grow in the number of levels among which the goods are distributed, the number of participants and the number of goods, and their interdependencies.

An SC can have several feasible configurations. In a feasible configuration, each active participant can buy her required goods and sell her produced goods. That is, for each good, there must be the same number of active buyers and sellers (i.e. the good must be at equilibrium). The example in Figure 1 allows several feasible SC configurations. For instance, configuration SC_1 : Alice $\rightarrow$ Dave $\rightarrow$ Frank is feasible, whereas SC_2 : Dave $\rightarrow$ Frank is not (nobody provides lime to Dave).

In general, an SCF problem will have $P = \{p_1, \ldots, p_n\}$ participants and $G = \{g_1, \ldots, g_m\}$ goods. An SC configuration is denoted by SC and its value is assessed by adding the activation costs of the active participants in SC. More

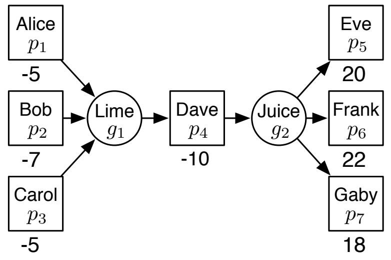


Figure 1: Example of a SCF scenario.

formally,

$$V(SC) = \sum_{p_i \in SC} C_{p_i}. \tag{1}$$

Moreover, for each good g we denote S_g as the set of participants willing to sell good g, and B_g as the set of participants willing to buy good g.

PROBLEM 1. *The SCF problem is that of finding the feasible configuration with maximum value (optimal configuration henceforth). In a scenario with participants P and goods G the optimal configuration (SC^*) can be assessed as:*

$$SC^* = \arg\max_{SC \subseteq P} V(SC)$$
$$\textit{subject to} \quad |S_g \cap SC| = |B_g \cap SC| \quad , \forall g \in G$$

We say that a good is at equilibrium when it has the same number of buyers and sellers. Hence, the SCF problem introduces a constraint for each good(g) that guarantees that it is at equilibrium, henceforth referred as equilibrium constraint. An SC configuration that satisfies every equilibrium constraint is guaranteed to be feasible.

In our example, the optimal configuration corresponds to $SC_1 = \{\text{Alice}, \text{Dave}, \text{Frank}\}$, and its value is $V(SC_1) = -5 + (-10) + 22 = 7$.

3. MAX-SUM ALGORITHM

Max-sum is an approximate optimization algorithm for unconstrained optimization that has been applied in different coordination problems. In the following, let $X = \langle x_1, \ldots, x_n \rangle$ be a sequence of variables, with each variable x_i taking states in a finite set $\mathcal{D}_i$ known as its *domain.* The joint domain $\mathcal{D}_X$ is the cartesian product of the domain of each variable. We use $\mathbf{x}_i$ to refer to a possible state of x_i, that is $\mathbf{x}_i \in \mathcal{D}_i$ and $\mathbf{X}$ to refer to a possible state for each variable in X, that is $\mathbf{X} \in \mathcal{D}_X$. We say that $\mathbf{X}$ is an assignment. Given a sequence of variables $Y \subseteq X$, factor f is a function $f : \mathcal{D}_Y \rightarrow [-\infty, \infty)$. We refer to the variables in the scope of f as X_f.

The max-sum algorithm provides an approximate solution to the problem of maximizing a function that decomposes additively as a sum of functions with smaller scope.

$$\text{maximize} \quad \sum_{f \in F} f(\mathbf{X}_f) \tag{2}$$

$$\text{subject to} \quad \mathbf{x}_i \in \mathcal{D}_i. \quad \forall i \in \{1, \ldots, n\} \tag{3}$$

where $\mathbf{X}_f$ contains the states assigned by $\mathbf{X}$ to the variables in the scope of factor f.

Max-sum provides an approximate solution for this problem in two steps. First, messages are sent from variable to factor and from factor to variable. This step is repeated until the messages no longer change or until a specified number of iterations is reached. After that, max-sum determines the best state for each variable independently.

Thus, at the first stage, max-sum assesses the message from variable x to factor f (μ_x^f) as follows:

$$\mu_x^f(\mathbf{x}) = \sum_{g \in \mathcal{N}(x)\setminus\{f\}} \mu_g^x(\mathbf{x}). \tag{4}$$

where $\mathcal{N}(x)$ stands for the factors that have variable x in its scope, $\mathbf{x}$ stands for a state of x, and μ_g^x stands for the last message received by variable x from factor g. Furthermore, the message from f to x (μ_f^x) is assessed as follows:

$$\mu_f^x(\mathbf{x}) = \max_{\mathbf{Y}} \left(f(\mathbf{x}, \mathbf{Y}) + \sum_{y \in Y} \mu_y^f(\mathbf{y}) \right). \tag{5}$$

where Y is the set of variables in the scope of factor f excluding x, and $\mathbf{Y}$ is their joint state. Notice that the assessment of the max-sum message for a factor over k d-ary variables takes time $\mathcal{O}(d^k)$.

At this point we can define the factor graph associated to a max-sum encoding. A factor graph is a bipartite graph with variable and factor nodes. The factor graph associated to a max-sum encoding, contains a variable node for each of the variables in the encoding and a factor node for each of the factors in the encoding. Moreover, there is a link joining a variable node an a factor node whenever the variable is in the scope of the factor.

At the second stage, max-sum assesses the preferred states for each variable x_i^* as

$$\mathbf{x}_i^* = \operatorname*{argmax}_{\mathbf{x}_i} \sum_{f \in \mathcal{N}(x_i)} \mu_f^{x_i}(\mathbf{x}_i). \tag{6}$$

Finally, max-sum is guaranteed to converge to the optimal configuration on acyclic factorizations. Whenever the underlying factorization contains cycles, max-sum may converge to a suboptimal solutions and even fail to converge. Yet, if max-sum converges, it is known to provide neighborhood maximum configurations [19, 12]. For instance, in factor graphs with a single cycle, the neighborhood maximum is the global maximum.

4. RELATED WORK

In this section we review the state-of-the-art on decentralized SCF after separating the contributions in the literature in two groups: market-based approaches and message-passing approaches.

4.1 Market-based approaches

In [1], Babaioff and Nisan provide a distributed mechanism providing ex-post individual rationality and incentive compatibility, budget balance and high global economic efficiency in linear SCs. Later on, Babaioff and Walsh [2] extended this result to SCs that satisfy the unique manufacturing technologies property, that is, markets where two participants that have the same good, should have exactly the same required goods in the same amounts. In this paper we are interested in the more general SCF scenario introduced in [15].

To efficiently solve a SCF problem, Walsh and Wellman introduced the Simultaneous Ascending (M+1)st Price with Simple Bidding protocol (SAMP-SB-D) [15]. SAMP-SB-D protocol is composed of an auction mechanism. A SAMP-SB-D mechanism comprises a set of auctions (run by mediators), one per good. Each auction runs independently of the other auctions in the SC. However, all auctions run simultaneously. Additionally, SAMP-SB-D provides simple bidding policies for participants. Since SAMP-SB-D may converge to solutions in which some participants obtain a negative utility to participate in the SC, the SAMP-SB-D protocol includes an additional phase that allows agents to decommit.

4.2 Message-passing approaches

Winsper and Chli [21] propose an alternative graphical representation for the encoding of a SCF problem as a factor graph. Under this representation, the SCF problem is cast as an optimization problem and subsequently approximated by the (max-sum) loopy belief propagation algorithm (LBP)[6]. Recently, Penya-Alba et al. proposed an alternative factor graph encoding for the SCF problem [10], the so-called Reduced Binary Loopy Belief Propagation (RB-LBP), which dramatically lowers max-sum requirements, from exponential to quadratic, while leading to higher valued SCs than LBP.

Since the goods are not explicitly represented in the encodings in [21] and [10], message-passing takes place directly between participants. This is different from SAMP-SB-D, which does employ mediators. Upon convergence, each message passing algorithm includes a decommitment phase to guarantee feasibility in the resulting SC configuration.

5. THE CHAINME ENCODING

In this section we start describing our approach for decentralized SCF, the so-called CHAINME (CHaining Agents IN Mediated Environments) algorithm. First, in what follows, we detail a novel way to encode the SCF problem defined in Section 2 into an unconstrained optimization problem. Thereafter, in Section 6 we will provide a description of the CHAINME algorithm, which is based on an efficient max-sum implementation specifically optimized to solve the unconstrained optimization problem introduced in this section.

To encode the SCF problem into a binary linear program, we represent each SC configuration by means of a set of binary variables, one per participant. Thus, for each participant p_i, her participant variable $\mathbf{p_i}$ takes value 1 if p_i is active in the SC configuration, and 0 otherwise. The SCF problem amounts to finding the values of $\mathbf{p_1}, \dots, \mathbf{p_n}$ that

$$\begin{aligned} &\text{maximize} && V(\mathbf{p_1}, \dots, \mathbf{p_n}) \\ &\text{subject to} && \sum_{p_i \in S_g} \mathbf{p_i} = \sum_{p_j \in B_g} \mathbf{p_j} \quad \forall g \in G \end{aligned}$$

where $V(\mathbf{p_1}, \dots, \mathbf{p_n})$ stands for the value of the SC encoded by the values taken by the variables $\mathbf{p_1}, \dots, \mathbf{p_n}$.

Now our purpose is to encode this problem in a way that can be solved by max-sum. With this aim, we introduce two types of factors on which max-sum is to operate: *activation* factors to assess the cost of each participant, and *equilibrium* factors to determine whether the equilibrium constraints are satisfied.

First, notice that the value of an SC, $V(\mathbf{p_1}, \dots, \mathbf{p_n})$, must include the activation cost of a participant p_i only if she is active. Thus, we define an *activation* factor for each participant p_i as follows:

$$f_{p_i}(\mathbf{p_i}) = \begin{cases} C_{p_i}, & \text{if } \mathbf{p_i} = 1 \\ 0, & \text{if } \mathbf{p_i} = 0. \end{cases} \quad (7)$$

In the example in Figure 1, where Alice is participant p_1, $f_{p_1}(\mathbf{p_1})$ will take value -5 if p_1 is active ($\mathbf{p_1} = 1$) and 0 otherwise. Therefore, the objective function of the above-defined binary linear program, corresponding to the SC value, can now be expressed as:

$$V(\mathbf{p_1}, \ldots, \mathbf{p_n}) = \sum_{p_i \in P} f_{p_i}(\mathbf{p_i}). \quad (8)$$

Second, since max-sum is unable to deal with constrained problems, we need to incorporate the equilibrium constraints into the expression to maximize. Thus, we will add additional factors, one per good, representing all equilibrium constraints. Each equilibrium factor will return 0 if the constraint is satisfied and $-\infty$ if it is violated. For each good g, the factor encoding the equilibrium constraint for the good only depends on the participant variables of the sellers (S_g) and buyers (B_g) of that good. Thus, we use $\mathbf{S_g} = \{\mathbf{p_i} | p_i \in S_g\}$ to denote the joint state of the variables of the sellers of g and $\mathbf{B_g} = \{\mathbf{p_i} | p_i \in B_g\}$ to denote the joint state of the variables of the buyers of g. The *equilibrium factor* for a good g is defined as:

$$f_g(\mathbf{S_g}, \mathbf{B_g}) = \begin{cases} 0, & \text{if } \sum_{p_i \in S_g} \mathbf{p_i} = \sum_{p_j \in B_g} \mathbf{p_j} \\ -\infty, & \text{otherwise.} \end{cases} \quad (9)$$

In the example in Figure 1, where Lime is good g_1, the equilibrium factor of good g_1 will take as arguments $\mathbf{S_{g_1}} = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\}$ and $\mathbf{B_{g_1}} = \{\mathbf{p_4}\}$. The value of g_1's equilibrium factor will be zero whenever all participants are inactive, or if p_4 is active and there is one and only participant active out of $\{p_1, p_2, p_3\}$.

Finally, we are ready to express the SCF problem as an *unconstrained* optimization problem:

$$\text{maximize} \quad \sum_{p_i \in P} f_{p_i}(\mathbf{p_i}) + \sum_{g \in G} f_g(\mathbf{S_g}, \mathbf{B_g}) \quad (10)$$

The first term of the objective function stands for the SC value as computed by Equation 8, whereas the second term stands for the equilibrium constraints. In the example in Figure 1, the assignment that maximizes Equation 10 is $\langle \mathbf{p_1}, \ldots, \mathbf{p_7} \rangle = \langle 1, 0, 0, 1, 0, 1, 0 \rangle$, whose SC value is $1 \cdot (-5) + 0 \cdot (-7) + 0 \cdot (-5) + 1 \cdot (-10) + 0 \cdot 20 + 1 \cdot 22 + 0 \cdot 18 = 7$.

Observe that, in Equation 10, we have managed to provide a factorization of the SCF problem defined in Section 2. It is worth noticing that CHAINME's factorization does not contain cycles as long as the original TDN does not contain cycles in its undirected form. Recall that max-sum's performance is known to be affected in factorizations [19] with higher number of cycles. On the other hand, other factorizations introduce additional cycles. That is because CHAINME mapping creates just one variable per participant and the connections between variables generated by the equilibrium factor of each good follow the links present in the TDN. For instance, in the TDN depicted in Figure 2, CHAINME produces a factorization without cycles, whereas RB-LBP will introduce a cycle.

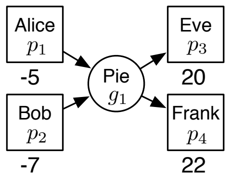


Figure 2: Example of a SCF scenario.

Moreover, CHAINME will be able to approximately solve the SCF problem by means of an efficient max-sum implementation as we detail in Section 6.

6. THE CHAINME ALGORITHM

CHAINME employs two types of agents, participants and mediators. On the one hand, there is a **participant agent** for each of the participants in the SC. Each participant agent communicates exclusively with the mediators of the goods it is interested in buying or selling. On the other hand, there is a **mediator agent** for each good on the SC. Each mediator agent is responsible of enforcing the equilibrium constraint of a particular good. Moreover, each mediator communicates only with the participant agents interested in buying or selling the good it is responsible for.

In this section we detail the operation of the CHAINME algorithm. Agents in CHAINME follow a protocol that has two phases: the *max-sum* phase and the *decommitment* phase. During the max-sum phase, agents apply a new efficient implementation of the max-sum algorithm to obtain an approximate solution to the problem in Equation 10. Since it could happen that eventually the solution assessed is unfeasible, during the decommitment phase. Unfeasible solutions are transformed into feasible ones.

Section 6.1 describes the max-sum phase, while Section 6.2 describes the decommitment phase. Finally, Section 6.3 analyzes CHAINME's resource requirements and compares them with the state-of-the-art.

6.1 Max-sum phase

Recall from Section 3 that the max-sum algorithm has two stages. During the first one, messages are iteratively exchanged between variables and factors until convergence (or a previously specified number of iterations) is reached. During the second one, a solution is assessed. In max-sum terms, the problem is distributed as follows. On the one hand, each mediator agent handles the equilibrium factor of one of the goods. On the other hand, each participant agent handles a participant variable along with its activation factor. In the following we provide a new and efficient way of assessing max-sum messages for the CHAINME encoding of the SCF problem.

6.1.1 Efficient message passing

Recall from Section 3 that, conventionally, the assessment of the messages from a factor to a variable in max-sum can take exponential time. Specifically, the assessment of the max-sum message for a factor over k binary variables takes time $\mathcal{O}(2^k)$. In the remainder of this section we provide CHAINME's efficient version of max-sum messages.

Since in CHAINME all participant variables are binary, agents can just exchange single-valued messages representing the difference between two values [7]. Intuitively, this value encodes the preference for the active state against the inactive one. We use ν_a^b to denote the message from agent a to agent b.

Before formally defining the messages from mediator to participant, we need to introduce some notation. Let $S_g^{\succ} = \langle s_1^{\succ}, \ldots, s_r^{\succ} \rangle$ (where $r = |S_g|$) be a sequence of the sellers of good g ordered decreasingly by the value of the last message sent to the mediator of good g. Similarly, let $B_g^{\succ} = \langle b_1^{\succ}, \ldots, b_t^{\succ} \rangle$ (where $t = |B_g|$) be a sequence of the buyers of good g ordered decreasingly by the value of the last message sent the mediator of good g.

Finally, we define

$$\eta = \max\{j | \nu^g_{s_j^{\succ}} + \nu^g_{b_j^{\succ}} \geq 0\}, \tag{11}$$

$$\tau^+ = \min(\nu^g_{b_\eta^{\succ}}, -\nu^g_{s_{\eta+1}^{\succ}}), \tag{12}$$

$$\tau^- = \max(-\nu^g_{s_\eta^{\succ}}, \nu^g_{b_{\eta+1}^{\succ}}), \tag{13}$$

$$S_g^a = \{s_i^{\succ} \in S_g^{\succ} | i \leq \eta\}, \tag{14}$$

$$B_g^a = \{b_i^{\succ} \in B_g^{\succ} | i \leq \eta\}. \tag{15}$$

Now, messages in CHAINME can be proved to be the following ones[1]:

- The message from **participant** p_i **to the mediator of good** g contains a single value $\nu^g_{p_i}$, computed as the addition of all messages received by p_i, excluding the message received from g, plus the participant's activation cost.

$$\nu^g_{p_i} = C_p + \sum_{g' \in \mathcal{N}(p_i) \setminus g} \nu^{p_i}_{g'}, \tag{16}$$

  where $\mathcal{N}(p_i)$ is the set of goods p_i is required to sell or buy if active, and $\nu^{p_i}_{g'}$ is the last message received by agent p_i from the mediator of good g'.

- The message from the **mediator of good** g **to its seller** s_i contains a single value $\nu_g^{s_i}$ that is assessed as:

$$\nu_g^{s_i} = \begin{cases} \tau^+, & \text{if } s_i \in S_g^a \\ \tau^-, & \text{otherwise .} \end{cases} \tag{17}$$

- The message from the **mediator of good** g **to its buyer** b_i contains a single value $\nu_g^{b_i}$ that is assessed as:

$$\nu_g^{b_i} = \begin{cases} -\tau^-, & \text{if } b_i \in B_g^a \\ -\tau^+, & \text{otherwise.} \end{cases} \tag{18}$$

Note that the assessment of messages from mediators to participants is particularly efficient.

The assessment of the standard max-sum message for an equilibrium factor with r sellers and t buyers can take $\mathcal{O}(2^{r+t})$ time. On the other hand, with CHAINME's efficient message calculation, it takes only $\mathcal{O}((r+t) \cdot \log(r+t))$ time (the time taken to order the messages). This simplification stems from taking benefit of the fact that our equilibrium factor represents a constraint and the alternatives that do not satisfy the constraint can directly be discarded.

Importantly, we are not approximating the messages, but proposing a particularly efficient way to assess them. Since the assessment of the messages is exact, it does not affect the quality of the solution achieved by the max-sum algorithm. Therefore, CHAINME keeps all theoretical guarantees proven for the max-sum algorithm. That is, CHAINME max-sum phase is guaranteed to converge to the optimal configuration on acyclic factorizations. Morover, in cyclic factorizations, CHAINME's max-sum phase will produce neighborhood maximum configurations whenever it converges. Moreover, as shown in Section 7, CHAINME also presents the good empirical performance that max-sum has shown in a wide variety of applications [18, 6].

6.1.2 *Basic solution assessment*

Since messages in CHAINME are single-valued, the basic solution assessment slightly differs from regular max-sum. However, the results are the same. In CHAINME, after message-passing, each participant p_i determines its activation value as

$$V_{p_i} = C_{p_i} + \sum_{g \in \mathcal{N}(p_i)} \nu_g^{p_i}. \tag{19}$$

Then, it decides to be active (to set its variable to 1), whenever it has a positive activation value. In this case, we say that p_i is active in the *basic solution.* It could happen that the solution assessed at this stage is unfeasible. Therefore, in the next section, we describe a decommitment phase that guarantees feasible solutions.

6.2 Decommitment phase

The decommitment phase follows an iterative protocol where messages are sent from participants to mediators, processed by mediators and sent back from mediators to participants. Essentially, at each iteration each mediator determines whether its good is at equilibrium. Whenever it is not, it forces some of the good's participants to be inactive. This is repeated until every good is at equilibrium.

In what follows we describe this decommitment phase in more detail. Each participant starts the decommitment phase by sending whether it is active or not (as assessed in the basic solution) to its neighboring mediators. Once a participant decides that it is inactive, it will no further change its status.

Algorithm 1 Active participant determination process for the mediator of good g mediator.

```
1: Let $\langle s'_1, \ldots, s'_n \rangle$ be the active sellers and $\langle b'_1, \ldots, b'_m \rangle$ be
   the active buyers, both sorted decreasingly by last message
2: $j \leftarrow 1$
3: $active \leftarrow \emptyset$
4: while $\nu^g_{s'_j} + \nu^g_{b'_j} \geq 0$ and $j \leq \max(|S_g|, |B_g|)$ do
5:     $active \leftarrow active \cup \{s'_j, b'_j\}$
6:     $j \leftarrow j + 1$
7: end while
```

After receiving the status from *all* its neighboring agents, good g's mediator determines the active participants as de-

[1]A complete derivation of the messages can be found in [9]. Although the results are new, the techniques are similar to the ones applied in [7, 11].

Measure	CHAINME	SAMP-SB-D	RB-LBP
Memory (overall)			
participant	$\mathcal{O}(G)$	$\mathcal{O}(G)$	$\mathcal{O}(G \cdot P)$
mediator	$\mathcal{O}(P)$	$\mathcal{O}(P)$	
Bandwidth (per iteration)			
participant	$\mathcal{O}(G)$	$\mathcal{O}(G)$	$\mathcal{O}(G \cdot P)$
mediator	$\mathcal{O}(P)$	$\mathcal{O}(P)$	
Operations (per iteration)			
participant	$\mathcal{O}(G)$	$\mathcal{O}(G)$	$\mathcal{O}(G \cdot P^2)$
mediator	$\mathcal{O}(P \cdot \log P)$	$\mathcal{O}(\log P)$	

Table 1: Worst case resource requirements.

scribed in Algorithm 1. The purpose of the algorithm is to match buyers and sellers who are still active and have the larger message values. After that, each mediator communicates to its participants whether they should be active or not.

When a participant receives the status request from each of its neighboring mediators, it decides to be active only if *all* of its neighboring mediators requested it to be active. Then, it sends its status to all the mediators it is connected with. This process continues iteratively until no participant changes its status. Participants who are active at this stage will be the active agents in the *decommited solution*, and will compose the SC configuration proposed by CHAINME.

6.3 Resource requirements

Next, we analyze the resources necessary to assess an SC configuration using CHAINME. Provided that there are n participants in a market with m goods, we will need a total of $n + m$ agents. Furthermore, assume that the maximum number of participants a good is connected to is P, and the maximum number of goods that a participant is connected to is G.

Memory. Each participant needs to store the last message received from each of the mediators it is connected to plus its activation cost, namely $\mathcal{O}(G)$ memory. Each mediator needs to store the messages sent by each of the participants it is connected to, namely $\mathcal{O}(P)$ memory.

Communication. For each iteration, each participant needs to send messages to and receive messages from each of its mediators, namely $\mathcal{O}(G)$ messages per iteration. Each message contains a single real number. Analogously, each mediator needs to send messages to and receive messages from each of its participants, namely $\mathcal{O}(P)$ messages per iteration.

Computation. At each iteration, each participant requires $\mathcal{O}(G)$ operations. On the other hand, the costliest operation for a mediator is ordering the messages, which takes $\mathcal{O}(P \cdot \log P)$ operations per iteration.

Table 1 compares the resources needed by the CHAINME, RB-LBP, and SAMP-SB-D algorithms. Computational requirements for SAMP-SB-D mediators are in the order of $\mathcal{O}(\log P)$ [22]. Finding other values for SAMP-SB-D is direct from its description in [15]. Values for RB-LBP are taken from [10].

7. EXPERIMENTAL EVALUATION

In this section we benchmark CHAINME against the state-of-the-art on decentralized SCF algorithms: SAMP-SB-D and RB-LBP. We perform our comparison in terms of solution quality (SC value) and resource requirements (bandwidth, computation and memory usage).

CHAINME has been tested in all the SC structures described in [15]. These problems represent toy examples where the three methods compared obtain optimal or near optimal solutions at a minimal cost. Thus, our evaluation focuses on large-scale problems where differences among these methods arise.

We use the test-suite described in [13], along the lines of [10], which is specifically designed to generate SCF problems. We generate SCs with 50 goods spread over four SC levels. We analyze different scenarios by varying the number of participants from 40 to 500. For each scenario, we generate 100 different instances. The code for the algorithms, the generated problems, and the results obtained can be freely downloaded from [8].

We solve each SCF problem with CHAINME, RB-LBP, and SAMP-SB-D. We measure bandwidth as the number of messages sent and received by each agent times the size of the message. To measure computation we simply count the number of operations[2] each agent performs.

Following [10], we impose a hard limit of 250 iterations after which the execution of RB-LBP and CHAINME is stopped and a solution is assessed. SAMP-SB-D is run until convergence since convergence is guaranteed [15]. The number of iterations and the convergence rate of each algorithm is discussed in detail in Section 7.2.

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 [20]. Where possible, we also show the 20th and 80th percentile as a measure of dispersion. Results reported in this section are statistically significant with a p-value ≤ 0.03.

Section 7.1 analyzes the quality of the solution obtained (in terms of the value of the SC) by each algorithm. Then, Section 7.2 analyzes the resource requirements of each algorithm.

7.1 Solution quality

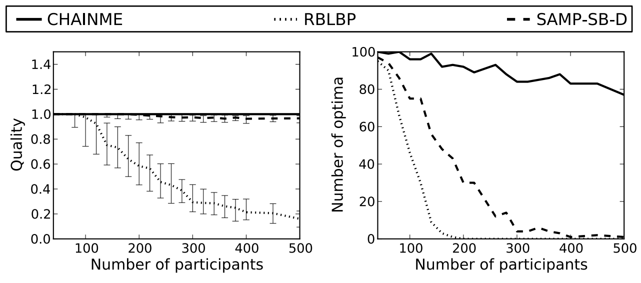
We normalize solution quality to the 0-1 scale by dividing by the value of the optimal SC.[3] Hence, a quality of one means that the SC found is optimal. Figure 3a shows the median and dispersion of the SC value for CHAINME, RB-LBP, and SAMP-SB-D as the number of participants increases[4]. We observe that CHAINME outperforms RB-LBP and performs slightly better than SAMP-SB-D. Moreover, the quality of the solution obtained by RB-LBP and SAMP-SB-D decreases as the number of participants increases whilst the quality for CHAINME remains almost constant (this effect is more noticeable for RB-LBP). Notice that, due to the small variance in the quality of the solutions assessed by CHAINME, the confidence bars for CHAINME are too small to be distinguished in print.

Figure 3b plots the number of problems for which each method was able to find the optimal SC. The number of

[2]The number of operations is measured as the number of additions/assignments/comparisons an agent performs.

[3]Optimal SCs are computed using a centralized mixed integer programming solver.

[4]Increasing the maximum number of iterations up to 2000 yields similar results in terms of quality.



(a) Median quality. (b) Number of optimal solutions.

Figure 3: Solution quality comparison analysis.

problems optimally solved by SAMP-SB-D and RB-LBP decreases very rapidly as the number of participants increases. By contrast, CHAINME finds the optimal SC in most of the problems, even in scenarios with a large number of participants. In the 500 participants scenario, CHAINME finds the optimal SC in more than 70% of the problems, whereas the other methods almost never find it.

In summary, the SC values assessed by CHAINME are larger than those obtained by its contenders. Moreover, CHAINME finds optimal solutions much more frequently.

7.2 Resource Requirements

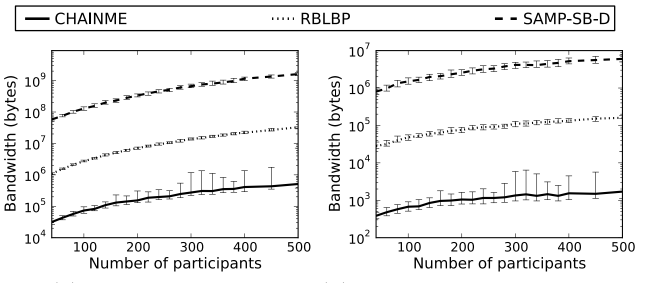
The memory requirements for each agent are proportional to the number of neighbors for the three algorithms compared. Thus, any current computational environment will be able to run any of the algorithms.

Recall that CHAINME and SAMP-SB-D are mediated algorithms whilst RB-LBP is not. For that reason we benchmark bandwidth usage along four different dimensions: total bandwidth used by all agents, maximum bandwidth used by any participant, total bandwidth used by mediators, and maximum bandwidth used by any mediator.

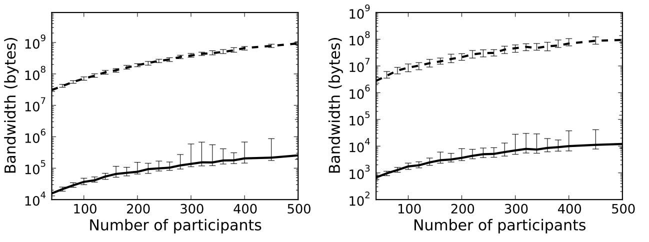
Figure 4 shows how the different algorithms performed in terms of bandwidth usage. Note that RB-LBP is left out of Figures 4c and 4d due to its lack of mediator agents. Figures 4a and 4b show that CHAINME uses at most 1/60 of the bandwidth used by RB-LBP and at least 3 orders of magnitude less bandwidth than SAMP-SB-D. This difference is when only considering mediators (Figures 4c and 4d).

Figure 5 shows how the different algorithms performed in terms of computation. In Figures 5a and 5b we see that the number of operations performed by CHAINME is at least 2 orders of magnitude less than those performed by the runner-up. This difference is confirmed when we only consider mediators in Figures 5c and 5d.

Recall from Section 6.3 that SAMP-SB-D's computational complexity was in the order of $\mathcal{O}(\log P)$ whereas CHAINME's was in $\mathcal{O}(P \cdot \log P)$. Then, it may seem counter-intuitive that CHAINME outperforms SAMP-SB-D in terms of computational requirements. However, as shown in Figure 6a, SAMP-SB-D requires up to four orders of magnitude more iterations than CHAINME to converge to a solution thus resulting in higher computational requirements. Finally, Figure 6b depicts the convergence rate of each of the algorithms. As expected, SAMP-SB-D converges in all instances (since convergence is guaranteed [15]). Notice that, CHAINME converges to a solution in over 70% of the instances even for the problems with 500 agents. Finally, RB-LBP is unable to converge to a solution in none of the problems tested.

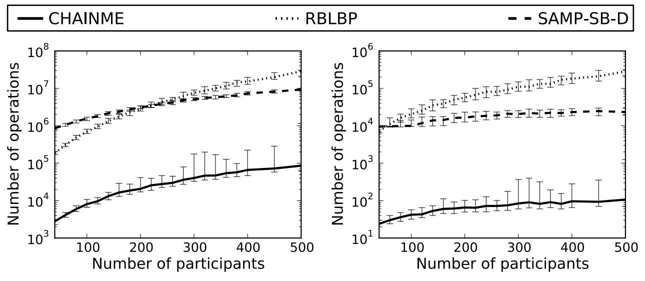


(a) Total bandwidth. (b) Maximum bandwidth per participant.

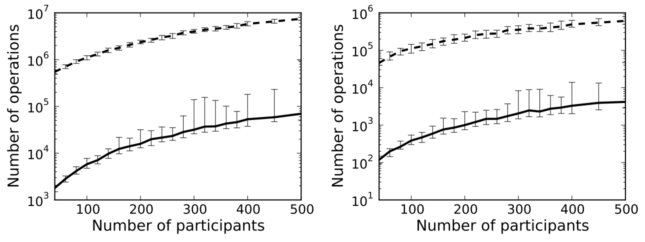


(c) Mediators' total bandwidth. (d) Maximum bandwidth per mediator.

Figure 4: CHAINME, RB-LBP, and SAMP-SB-D bandwidth requirements. Plots use a log-scale for the y axis.



(a) Total operations. (b) Maximum operations per participant.



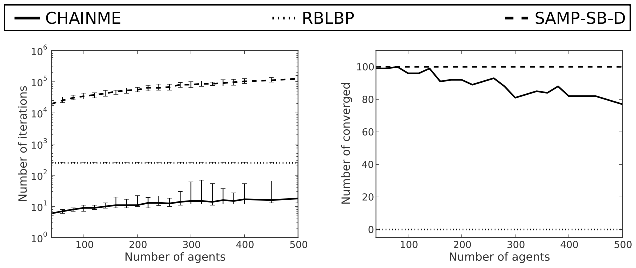
(c) Mediators' total operations. (d) Maximum operations per mediator.

Figure 5: CHAINME, RB-LBP, and SAMP-SB-D operations. Plots use a log-scale for the y axis.

In summary, we have shown that CHAINME yields better-valued solutions than the state-of-the-art algorithms for decentralized SCF while requiring from one up to four orders of magnitude less resources.

8. CONCLUSIONS AND FUTURE WORK

We have described CHAINME, a novel decentralized SCF algorithm. In our experiments, CHAINME outperforms state-of-the-art algorithms, RB-LBP and SAMP-SB-D, in terms of the value of the SCs obtained. For instance, CHAINME finds the optimal SC in more than 70% of the instances, whereas



(a) Number of iterations. (b) Number converged.

Figure 6: CHAINME, RB-LBP, and SAMP-SB-D iterations and convergence. Plots use a log-scale for the y axis.

the other methods almost never find them, for large-scale SCF problems. Furthermore, CHAINME consumes less than one tenth of the communication resources and one percent of the computational resources used by those algorithms.

Note that, since no payment function has been defined, CHAINME (like RB-LBP) should only be considered as a distributed approximate winner determination algorithm. Therefore, the design of a mechanism would require the definition of a payment function. The design of such payment function and the analysis of the properties of the corresponding mechanisms should be pursued as future work.

The experiments show that the SC values obtained by CHAINME are optimal much more often than the state-of-the-art algorithms. Thus, we consider that it could be an approximate, low resources alternative to optimal centralized approaches (such as standard integer linear programming solvers) in very large SCF scenarios. We plan to explore that path in the future.

9. ACKNOWLEDGMENTS

This work was funded by projects AT(CSD2007-0022), COR(TIN2012-38876-C02-01), MECER (201250E053) and the Generalitat of Catalunya (2009-SGR-1434).

10. REFERENCES

[1] M. Babaioff and N. Nisan. Concurrent auctions across the supply chain. *Journal of Artificial Intelligence Research*, 21:595–629, 2004.

[2] M. Babaioff and W. E. Walsh. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems*, 39(1):123–149, 2005.

[3] D. A. Burke, K. N. Brown, M. Dogru, and B. Lowe. Supply chain coordination through distributed constraint optimization. In *9th International Workshop on Distributed Constraint Reasoning*, 2007.

[4] J. Cerquides, U. Endriss, A. Giovannucci, and J. A. Rodriguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *IJCAI*, pages 1221–1226, 2007.

[5] J. Collins, W. Ketter, and M. Gini. A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *Int. J. Electron. Commerce*, 7:35–57, 2002.

[6] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, pages 639–646, 2008.

[7] I. E. Givoni and B. J. Frey. A binary variable model for affinity propagation. *Neural computation*, 21(6):1589–1600, 2009.

[8] T. Penya-Alba, M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar. Code and experiments. `http://db.tt/V7zQtrTS`.

[9] T. Penya-Alba, M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar. Technical report: Chainme message simplification. `http://db.tt/MrTvVxpl`.

[10] T. Penya-Alba, M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar. A scalable Message-Passing Algorithm for Supply Chain Formation. In *26th Conference on Artificial Intelligence (AAAI)*, 2012.

[11] D. Tarlow, I. E. Givoni, and R. S. Zemel. Hop-map: Efficient message passing with high order potentials. In *International Conference on Artificial Intelligence and Statistics*, pages 812–819, 2010.

[12] M. Vinyals, A. Farinelli, J. A. Rodríguez-aguilar, et al. Worst-case bounds on the quality of max-product fixed-points. In *Advances in Neural Information Processing Systems*, pages 2325–2333, 2010.

[13] M. Vinyals, A. Giovannucci, J. Cerquides, P. Meseguer, and J. A. Rodriguez-Aguilar. A test suite for the evaluation of mixed multi-unit combinatorial auctions. *Journal of Algorithms*, 63:130–150, 2008.

[14] W. E. Walsh. *Market protocols for decentralized supply chain formation*. PhD thesis, U. of Michigan, 2001.

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

[16] W. E. Walsh, M. P. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *ACM Conference on Electronic Commerce*, pages 260–269, 2000.

[17] W. E. Walsh, M. Yokoo, K. Hirayama, and M. P. Wellman. On market-inspired approaches to propositional satisfiability. *Artificial Intelligence*, 144(1):125–156, 2003.

[18] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1–41, 2000.

[19] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736–744, 2001.

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

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

[22] P. R. Wurman, W. E. Walsh, and M. P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24(1):17–27, 1998.