Approximate algorithms for decentralized Supply Chain Formation

Toni Penya-Alba

Approximate algorithms for decentralized Supply Chain Formation

by

Toni Penya-Alba

Advisors:
Dr. Jesus Cerquides Bueno
Dr. Juan A. Rodriguez-Aguilar

Doctorat en Matemàtiques i Informàtica.
Ph.D. in Mathematics and Computer Science.

Facultat de Matemàtiques
Departament de Matemàtica Aplicada i Anàlisi
September 2014

The road and the tale have both been long,
would you not say so?

The trip has been long and the cost has been high...
but no great thing was ever attained easily.

A long tale, like a tall Tower,
must be built a stone at a time.

–Stephen King,
The Dark Tower

Contents

List of Figures

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. Decentralized decision making is well suited to this scenario since it better preserves the privacy of the participants, offers better scalability on large-scale scenarios, and is more resilient to failure. Moreover, decentralized supply chain formation can be tackled either by means of peer-to-peer communication between supply chain participants or by introducing local markets that mediate the trading of goods. Unfortunately, current approaches to decentralized supply chain formation, both in the peer-to-peer and the mediated scenario, are unable to provide computationally and economically efficient solutions to the supply chain formation problem.

The main goal of this dissertation is to provide computationally and economically efficient methods for decentralized supply chain formation both in the peer-to-peer and the mediated scenario. This is achieved by means of two optimized max-sum based methods for supply chain formation.

On the one hand, we contribute to peer-to-peer supply chain formation via the so-called Reduced Binarized Loopy Belief Propagation (RB-LBP) algorithm. The RB-LBP algorithm is run by a multi-agent system in which each of the participants in the supply chain is represented by a computational agent. Moreover, RB-LBP's message computation mechanisms allow the efficient computation of max-sum messages. This results in an algorithm that is able to find solutions to the supply chain formation problem of higher value than the state of the art while reducing the memory, bandwidth and computational resources required by several orders of magnitude.

On the other hand, we contribute to mediated supply chain formation via the so-called CHaining Agents IN Mediated Environments (CHAINME) algorithm. The CHAINME algorithm is run by a multi-agent system in which each of the participants and each of the goods in the supply chain is represented by a computational agent. In CHAINME participant agents communicate exclusively with the agents representing the goods who act as mediators. Likewise RB-LBP, CHAINME is also endowed with a message computation mechanism for the efficient computation of max-sum messages. This results in an algorithm that is able to find economically efficient solutions while requiring a fraction of the computational resources needed by the state-of-the-art methods for both peer-to-peer and mediated supply chain formation.

Finally, the design and implementation of both of our contributions to decen-

tralized supply chain formation follow the same methodology. That is, we first map the problem at hand into a local term graph over which max-sum can operate. Then, we assign each max-sum local term to a computational agent. Last, we derive computationally efficient expressions to assess the max-sum messages exchanged between these agents. Although our methodology proved to be valid for the design of SCF algorithms, its generality makes it appear as a promising candidate for other multi-agent coordination problems.

Acknowledgements

During the past few years I had the chance to work side by side with some of the most interesting and brilliant minds I have come along in my life. To say that this journey has been rewarding at a professional and personal level might be an understatement. Among the numerous people that have shared this journey some deserve special mention. I hope not forget anyone.

First, I want to express my gratitude to my advisors, Jesus and Juan, for giving me the chance to enrol in this journey. From them I have learnt not only how to be a better professional but also how to be a better person. I also want to express my gratitude to Meritxell, my advisor-in-the-shadows who has been a source of inspiration and optimism from day one.

Also, I would like to express my most sincere gratitude to Prof. Nick Jennings and the AIC group for their guidance during the short stay I realized in Southampton during the last year of my PhD. To Enrico and Seb for their insights in areas that were out of my reach. To Amir, Beining, Chetan, Chris, James, Long, Mateo, and Moody for the good coffee breaks shared. And to Zoli, for the many hours we expended thinking on how to coordinate those UAVs of his.

To everyone at the Institut d'Investigaciò en Intel.ligència Artificial for creating such an environment for research. I want to thank the staff for being always open to discussion, the administrative staff for being always open to finding solutions to any problem, and the reception desk clerks for their welcoming smiles. Special mention goes to my fellow PhD students, Andrew, Javi, Jesus, Jose Luis, Mari, Norman, Pablo, Pere, Tomas, Xavi, and many others for the good moments we shared both in and out of work. And to Marc, my brother-in-thesis, for providing a fresh point of view whenever I was struggling with a problem.

A mis padres y hermanas por creer en mi más que nadie. Esta tesis no hubiera sido posible sin su apoyo incondicional, sus ánimos y soporte constantes.

A mis amigos de Terrassa, los de toda la vida, por estar siempre ahí, aceptar mis neuras y perdonar mis errores. Mención especial merecen Cristian por ser la mente analítica con la que tantas veces hemos disectionado el universo, Dani por no juzgarme jamás y ser el mejor compañero de aventuras y Noemi por entenderme mejor que yo mismo y servirme de guía cuando he andado perdido.

Per últim vull agraïr als meus dos amors, la muntanya i la música, sense les quals hagués estat impossible mantenir el seny aquests quatre anys. Vull agraïr en particular als escaladors del SAF (Albert, Erica, Julia, Leo, Maria, Pau, Sergio, ...) per ser una colla tan oberta i als Tronats del Tabal per donar-me l'opotunitat de formar part d'aquest increïble projecte musical.

This work has been partially funded by projects EVE (TIN2009-14702-C02-01), AT (CSD2007-0022), CSIC 201050I008, Generalitat of Catalunya (2009-SGR-1434), COR(TIN2012-38876-C02-01), and MECER (201250E053).

Chapter 1

Introduction

In the current industrial global setting, enterprises are facing an era of changing production paradigms. Instead of taking care of the whole production process, from raw materials to final consumer-ready product, companies rely more and more on outsourcing both the components and the processes needed to obtain the desired goods [Collins et al., 2002]. That, paired with the fact that transportation costs are decreasing while delivery times grow shorter, is leading to the creation of the so called extended enterprise: the dynamic network of interconnected organizations, from suppliers' suppliers to customers' customers, which work collaboratively to bring value to the marketplace.

There is plenty of evidence of businesses shifting their production models towards this new paradigm [Norman et al., 2004]. Large traditional manufacturing companies are increasingly outsourcing their production. Software companies forward part of their work to subcontracted companies in India [Borenstein and Saloner, 2001]. Start-ups form temporal coalitions that allow them to compete with larger companies. In all these processes it is of critical importance to be able to choose the right partners among those available in the market.

This new production paradigm, in which companies no longer take care of the whole production process but rather combine their resources, can be studied as Supply Chain Formation (SCF). In the rest of this chapter we first, in Section 1.1 introduce the SCF problem and argue that there is a need for automating SCF. Then, in Section 1.2 we introduce the different approaches to SCF found in the literature highlighting their advantages and weaknesses. Furthermore, in Section 1.3, we outline the contributions in this thesis to overcome some of these shortcomings. Finally, in Sections 1.4 and 1.5, we provide a list publications produced by this thesis and a guide for the reader to the thesis.

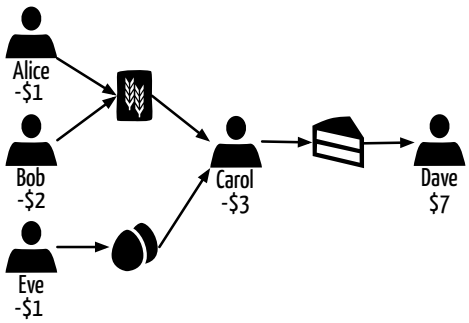


Figure 1.1: Simple SCF scenario described in Example 1.

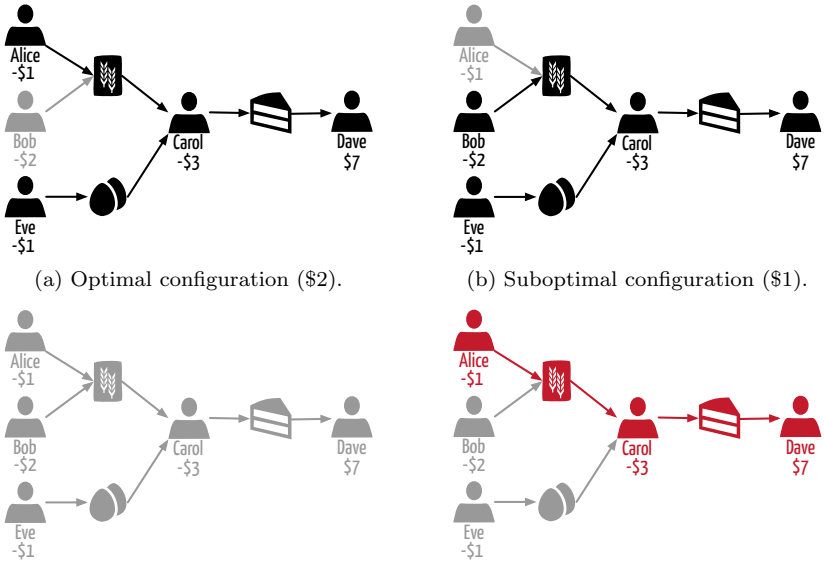
1.1 The Supply Chain Formation problem

In [Walsh and Wellman, 2003], the authors define a Supply Chain (SC) as "a network of production and exchange relationships that spans multiple levels of production or task decomposition". The most basic supply chain is that in which a supplier provides a consumer with a good. According to [Walsh and Wellman, 2003], "Supply Chain Formation (SCF) is the process of determining the participants in the supply chain, who will exchange what with whom, and the terms of the exchanges."

Next, we provide a simple example that illustrates the Supply Chain Formation problem. This example will serve to introduce key concepts to SCF.

Example 1. Consider a service requester, Dave, who wants to buy some breakfast. The participants in the SC for providing Dave with his breakfast are depicted in Figure 1.1. In this case, Dave is willing to pay $7 for a piece of cake. The piece of cake can be produced by Carol given that she can obtain some flour and eggs and she is paid $3 for baking the cake. Carol has to decide wether to buy the flour either from Alice at $1 or from Bob at $2. Finally, Carol has to buy eggs from Eve at $1 to produce the cake.

We say that Carol is a seller of cakes and a buyer of flour and eggs. Similarly, flour and eggs are the inputs of Carol's transformation while a piece of cake is the output of her production process. In general, the input goods of a participant are those she requires to perform her transformation. Similarly, her output goods are those obtained after performing her transformation. In Example 1, flour is the output good for both Alice and Bob and the input good for Carol. In the same way, the cake is Carol's output good and Dave's input good. Whenever a participant requires a good we say that she is a buyer for that good and, whenever she produces a good we say she is acting as a seller for the produced good. In Example 1, Alice and Bob are flour sellers, Eve is an eggs seller, Carol is a flour and eggs buyer and a cake seller, and Dave is a cake buyer. A certain set of goods are complementary for a participant when she needs to acquire all of them in order to produce her output [Walsh and Wellman, 2003]. In the



(a) Optimal configuration ($2).

(b) Suboptimal configuration ($1).

(c) Empty configuration ($0).

(d) Unfeasible configuration.

Figure 1.2: Solutions to the problem in Example 1. SC values inside parentheses.

example, Carol needs both flour and eggs in order to produce a cake. Neither flour nor eggs are of any use to Carol unless she can obtain the other. Therefore, flour and eggs are complementary goods for Carol.

The participants in a SC can either be part of the SC process (active) or not (inactive). A SC configuration contains a subset of active participants. Furthermore, a SC configuration is feasible if all active participants are able to buy their input goods as well as to sell their output goods [Walsh and Wellman, 2003]. For instance, Figure 1.2a depicts one feasible SC configuration. Active participants are shown in black while inactive participants are shown in grey. This configuration, in which Alice, Carol, Dave and Eve are active while Bob is inactive, is a feasible one since all active participants buy their input goods and sell their output goods. On the other hand, the SC configuration in which Alice, Carol and, Dave are active and Bob and Eve are inactive (Figure 1.2d) is not feasible since nobody is selling eggs to Carol. Besides the feasible configuration in Figure 1.2a, Example 1 allows for two further feasible configurations:

- Bob, Carol, Dave, and Eve are active; while Alice is inactive. That is, Carol buys flour from Bob and, eggs from Eve, bakes the cake and sells it to Dave. This configuration is depicted in Figure 1.2b.
- None of the participants is active. This configuration is depicted in Figure 1.2c

Following [Walsh and Wellman, 2003], the value of a SC configuration is de-

fined as the total surplus of the SC configuration. Each participant in a SC incurs in an activation cost whenever she takes part on the SC. These costs correspond to the amount a participant needs to be paid or the amount a participant is willing to pay for taking part in the SC. Participants that need to be paid (e.g. Alice, Bob, Carol and, Eve) have negative activation costs whereas participants that are willing to pay (e.g. Dave) have positive activation costs. Therefore, the value of a SC configuration, amounts to the addition of the activation costs of all the active participants. The value of the SC configuration depicted in Figure 1.2a is $2 (7-3-1-1). Moreover, the values of the SC configurations depicted in Figure 1.2b and Figure 1.2c are $1 and $0 respectively. Notice that the SC configuration in which all the participants are inactive is always feasible and its value is zero.

Solving the SCF problem amounts to finding the feasible SC configuration with maximum value [Walsh and Wellman, 2003]. Recall that the SC in Example 1 only allows the three feasible configurations depicted in Figures 1.2a, 1.2b, and 1.2c, whose values are $2, $1, and $0 respectively. Thus the optimal solution to the SCF problem described in Example 1 is the one in Figure 1.2a where Alice, Carol, Dave, and Eve are active.

So far, we have introduced the SCF problem with the help of a simple example. In what follows, we argue that manually solving the SC problem is unpractical for real-world problems and that there is a need for automated SCF.

1.1.1 The need for automated Supply Chain Formation

Today's market is in constant change. Producers are faced with ever-changing customer needs and resources costs and availability. Consequently it is no longer possible to maintain SCs over extended periods of time [Walsh and Wellman, 2003]. Thus, the ability to quickly form effective, mutually beneficial trading partnerships becomes increasingly important. That is, today's companies are in need for support to swiftly create business collaborations that allow them to readily respond to changing market needs. Moreover, the logistical complexity of manufacturing and other business activities has been increasing nearly exponentially [Collins et al., 2002]. The old approach for SCF, where SCs were assessed manually after extended negotiations is no longer viable. Furthermore, human irrationality coupled with the complexity of the problem often lead to inefficient SCs [Winsper, 2012]. Therefore, there is a need for agile and flexible methods that support temporal collaboration between enterprises [Norman et al., 2004].

One such method is that of computational agents. Computational agents are able to explore a larger number of offers. Moreover, they are able to evaluate scenarios that are too complex for humans. Therefore, computational agents are well suited for the task at hand [Collins et al., 2002]. Indeed, such techniques have proved to be useful in real-world scenarios saving over $5 billion to businesses by means of combinatorial auctions [Sandholm, 2008].

In this section we have introduced the SCF problem. Moreover, we have

argued that manually solving the SCF problem becomes unpractical as the size and complexity of SCs grow. Thus, the need for automating the SCF process arises. In the next section we provide a brief overview of the approaches in the literature to automated SCF. Moreover, we point out areas with room for improvement as well as key questions that motivate this thesis.

1.2 Motivation

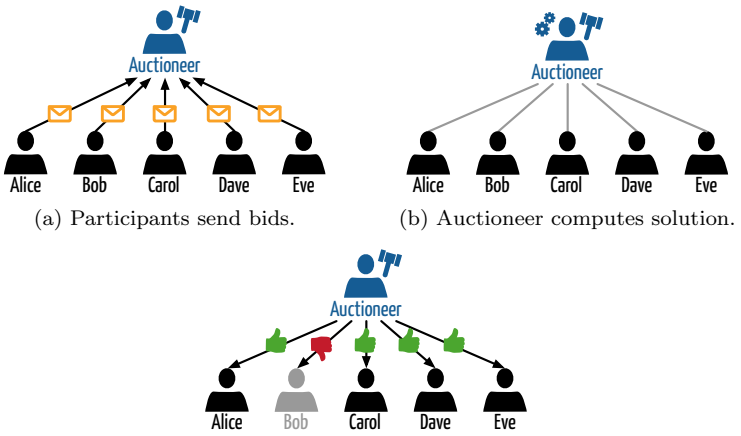
The SCF problem has been widely studied in the multi-agent systems literature [Davis and Smith, 1983, Walsh et al., 2000, Collins et al., 2002, Walsh and Wellman, 2003, Cerquides et al., 2007, Giovannucci et al., 2008, Winsper and Chli, 2010, Mikhaylov et al., 2011, Winsper and Chli, 2013]. In these approaches participants are represented by computational agents (participant agents henceforth) that act in their behalf during the SCF process [Norman et al., 2004]. After the deliberation is over, agents assess the new SC in a fraction of the time required by the manual approach [Winsper, 2012].

The SCF problem has been addressed in the literature mainly by centralized methods [Walsh et al., 2000, Collins et al., 2002, Cerquides et al., 2007, Giovannucci et al., 2008, Mikhaylov et al., 2011]. In centralized approaches, participant agents initially submit bids (that encode their preferences) to a central authority. This central authority assesses the SC configuration with maximum value and notifies the participant agents of the outcome. However, a much less explored field is that of decentralized approaches for SCF which disregard a central authority when deciding which participants must be active [Walsh and Wellman, 2000, Walsh and Wellman, 2003, Winsper and Chli, 2010, Winsper and Chli, 2013].

Next we provide more details about both centralized and decentralized approaches to SCF. Moreover, we go one step further in the decentralized SCF and differentiate between approaches that use direct communication between participant agents and those that employ mediators for goods.

1.2.1 Centralized approaches

When facing the decision of how to solve the SCF problem one might be initially compelled to employ a centralized solution [Norman et al., 2004]. That is, to rely on a central authority to assess the resulting SC. Several contributions employ an auctioneer to solve the SCF problem in a centralized manner by means of Combinatorial Auctions (CAs)[Walsh et al., 2000, Collins et al., 2002, Cerquides et al., 2007, Giovannucci et al., 2008, Mikhaylov et al., 2011]. CAs [Cramton et al., 2006] are a negotiation mechanism well suited to deal with complementarities among the goods at trade. Since production technologies often have to deal with strong complementarities, SCF automation appears as a very promising application area for CAs. Mixed Multi-Unit Combinatorial Auctions (MMUCAs) were introduced in [Cerquides et al., 2007] as a generalization of standard CAs. MMUCAs extend the expressivity of standard CAs. Moreover, there



(a) Participants send bids.

(b) Auctioneer computes solution.

(c) Auctioneer notifies participants.

Figure 1.3: Information flow in a centralized architecture.

is a computationally efficient solver for MMUCAs [Giovannucci et al., 2008] that provides the optimal SC.

In such scenario as described above, an auctioneer would determine the active participants in the SC after collecting information about all the participants involved in the production of a good or a set of goods. When solving the SCF problem described in Example 1 in a centralized manner, each of the participants (Alice, Bob, Carol, Dave and Eve) will rely on a participant agent to communicate with a central auctioneer. Figure 1.3 depicts the information flow in such scenario. First, participants send their bids to the auctioneer (Figure 1.3a); then, the auctioneer computes the SC configuration (Figure 1.3b); finally, the auctioneer communicates to each participant whether she should be active in the SC or not (Figure 1.3c). The directed edges between participant agents and auctioneer represent the information flows. Notice that, when solving the problem in a centralized manner there is no communication between participant agents.

However appealing a centralized optimal solution seems it might not be the best strategy [Hayek, 1945]. In general, there are several arguments against solving the SCF problem in a centralized manner and in favor of a decentralized approach. First, due to the decentralized nature of the problem, no central entity might have the allocative authority to perform such operation [Walsh and Wellman, 2003]. Second, even finding a feasible configuration is an NP-HARD problem [Walsh and Wellman, 2000, Fionda, 2009]. Therefore, the SCF problem in large markets results in complex optimization problems [Babaioff and Nisan, 2004] which might render impossible to solve the problem in an exact manner due to computational constraints. Third, centralizing the communication and the computation in a central entity introduces a single point

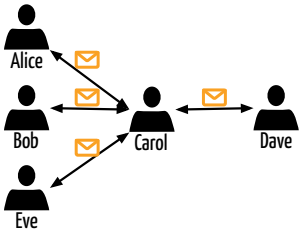


Figure 1.4: Information flow in a P2P architecture.

of failure [Winsper, 2012]. Therefore, certain SCF problems are better addressed by some form of decentralization. In the next section we introduce two forms of decentralization.

1.2.2 Decentralized approaches

When facing the problem of assessing SCs in a decentralized manner, the first question to answer is how to distribute knowledge [Hayek, 1945]. A common approach is for each participant in the SC to be represented by a participant agent that acts on her behalf during the SCF process [Norman et al., 2004]. However, the question of how agents communicate and assess the SC with maximum value still remains open. On the one hand, participant agents in decentralized SCF can engage in a peer-to-peer negotiation with their suppliers and consumers. On the other hand, these participant agents, can resort to local markets in which the goods they want to sell or buy are being traded. In what follows we outline both approaches for solving the decentralized SCF problem.

Peer-to-peer approaches

A first approach to decentralized SCF is peer-to-peer (P2P) communication. In P2P approaches, each participant agent directly communicates with the participant agents of the buyers of the goods she is producing and the sellers of the goods she is consuming [Winsper and Chli, 2013]. Therefore, the SCF process takes place directly between the participant agents.

When solving the SCF problem described in Example 1 in a decentralized manner with peer-to-peer communication, each participant agent will communicate with the participant agents of their potential partners. For instance, Alice's participant agent will only communicate with Carol's since Carol is the only participant interested in buying flour (Alice's only output). Figure 1.4 depicts such scenario. The edges between participant agents represent the possible information flows.

An interesting approach to solve the SCF problem in a decentralized manner with P2P communication was proposed in [Winsper and Chli, 2010]. This approach casts the SCF problem as an optimization problem that can be approximated using max-sum. Max-sum is a message passing algorithm that has

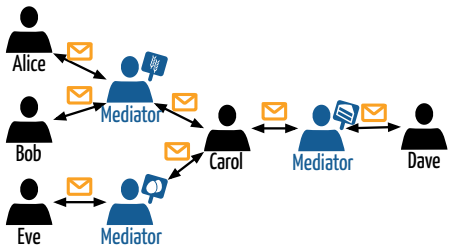


Figure 1.5: Information flow in a mediated architecture.

been widely used in the coordination of agents with very promising results [Farinelli et al., 2008, Stranders, 2009].

Although it is the first fully decentralized approach, the work in [Winsper and Chli, 2010] suffers from several drawbacks. For instance, each of the participant agents has to communicate with all its potential partners, thus increasing its communication requirements. Moreover, a participant agent needs to decide whether or not to collaborate with each potential partner. Therefore, the growth of potential partners comes with increased communication and computation for the participant agent.

Mediated approaches

An alternative to the direct communication between participant agents is the introduction of a local market for each of the goods in the SC. Each of these markets is represented by a mediator agent (mediator henceforth) that communicates with the sellers and buyers of the good at trade in the market. Therefore, markets allow participants to reach new potential partners [Kalin, 2000] with no extra cost in terms communication or computation for the participants. Notice that markets can be profitable in a variety of ways such as advertising, charging transaction fees, or membership fees. Furthermore, a broader selection of buyers and sellers means that high profit can be achieved even if the fees are low [Kalin, 2000].

When solving the SCF problem described in Example 1 in a mediated manner, each participant agent will solely communicate with the mediators of the goods she is interested in buying or selling. Figure 1.5 depicts such scenario. For instance, Carol's participant agent will communicate with the flour, egg and cake mediators, whereas Alice's participant agent will only communicate with the flour mediator. The edges between participant agents and mediators represent the possible information flows.

In [Walsh and Wellman, 2003], the authors propose SAMP-SB-D, a method for solving the SCF problem in a decentralized manner with the use of mediators. In this approach there is a mediator agent for each of the goods at trade and participant agents communicate exclusively with the mediators of the goods they are interested in buying or selling. On the one hand, each

participant agent submits ascending bids to the mediators. On the other hand, each mediator simultaneously runs an auction for the good it is in charge of. Although this method achieves high economical efficiency, as discussed in [Winsper and Chli, 2013], it incurs in a high penalty in terms of communication. SAMP-SB-D is costly in terms of communication due to the fact that its mediators need to re-evaluate and communicate the winners of the running auctions each time they receive a new bid.

1.2.3 Analysis

So far we have introduced the different approaches in the literature to SCF in terms of the agents involved in the process and how they organize and communicate. However, in the literature, there is no comparison in terms of economical nor computational efficiency between these approaches for SCF. Moreover it is unclear whether decentralized methods can perform efficiently in terms of computation and the value of the SCs assessed. Therefore, the following research questions remain unanswered:

Question 1. Can decentralized SCF methods be economically efficient?

Question 2. Can decentralized SCF methods be computationally efficient?

In the next section we detail our contributions in this thesis that aim to answer the aforementioned research questions.

1.3 Contributions

In this thesis we contribute to the state-of-the-art with two methods for the decentralized assembly of SCs. Both methods (RB-LBP and CHAINME) are based on the max-sum algorithm. Moreover, RB-LBP follows the P2P architecture for decentralized SCF whereas CHAINME follows the mediated architecture. Our methods advance the state-of-the by providing a computational and economical efficient way to solve the SCF problem in a decentralized manner. That is, we propose two decentralized methods for SCF that:

1. **Produce economically efficient solutions.** We show that CHAINME is able to find solutions to the SCF problem that are close to the optimal one (98% of the optimal value). Moreover, CHAINME is able to find the optimal solution in 78% of the instances. On the other hand, the value of the SCs assessed by RB-LBP can be up to 2 times higher than those assessed by the state-of-the art method for P2P SCF as the size of the problem grows.

2. **Approximate solutions to the SCF problem in a computationally efficient manner.** On the one hand, RB-LBP uses over 700 times less bandwidth, runs up to 20 times faster, and uses up to five orders of magnitude less memory than the state-of-the-art methods for P2P SCF in

large-scale scenarios. On the other hand, CHAINME is able to assess solutions up to 100 times faster while using up to three orders of magnitude less bandwidth than the state-of-the-art methods for mediated SCF with no impact on the memory requirements.

Moreover, the design and implementation of both of our contributions to decentralized supply chain formation follow the same methodology. Therefore, we further contributed to the state of the art with:

3. **A methodology for the design of decentralized decision making algorithms** that proved to be valid for the SCF problem. Furthermore its generality makes it appear as a promising candidate for other multi-agent coordination problems.

To summarize, we proposed a methodology for the design of decentralized decision making algorithms that resulted in two max-sum based algorithms that prove to be computationally efficient as well as to provide solutions of higher value than the state-of-the-art methods for decentralized SCF.

1.4 Dissertation outline

The remaining of this dissertation is organized as follows. In Chapter 2, we provide background knowledge on Periodic Double Auctions and the max-sum algorithm. This background is needed for understanding the concepts in following chapters.

In Chapter 3, we provide the context for our work. We introduce the state of the art methods for Supply Chain Formation. Moreover, we classify this methods into three broad categories: centralized SCF, decentralized P2P SCF, and decentralized SCF with mediators for goods.

In Chapter 4, we provide a formalization of the SCF problem in which participants have only knowledge of their potential buyers and sellers. Moreover, we provide a method for solving the decentralized SCF problem with direct communication between participant agents. Furthermore, we experimentally evaluate our approach against the state of the art in P2P SCF.

In Chapter 5, we provide a formalization of the SCF problem in which participants have knowledge of markets trading their input and output goods. Moreover, we provide a method for solving the SCF problem in decentralized manner in which participant agents communicate with mediators for goods. Furthermore, we experimentally evaluate our approach against the state of the art in mediated SCF with mediators for goods. Finally, we provide a thorough comparison between all the state-of-the-art methods for SCF regardless of their nature.

Finally, in Chapter 6, we draw some conclusions and describe lines for future research.

1.5 Publications

The material contained in this thesis has produced the following publications:

1. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012d). RB-LBP: Scaling Up Decentralized supply chain formation. In *5th International Workshop on Optimization in Multi-Agent Systems @AAMAS (OPTMAS 2012)*, Valencia.

2. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012f). Scalable decentralized supply chain formation through binarized belief propagation. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1275–1276, Valencia.

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

4. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2013c). Social Value Propagation for Supply Chain Formation. In *6th International Workshop on Optimization in Multi-Agent Systems @AAMAS (OPTMAS 2013)*.

5. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2013b). CHAINME: Fast Decentralized Finding of Better Supply Chains. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*.

6. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2014b). Exploiting Max-Sum for the Decentralized Assembly of High-Valued Supply Chains. In *13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*.

Moreover, the experimental evaluation conducted in this thesis has produced the following code and datasets:

7. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012e). Rb-lbp source code. `http://iiia.csic.es/~tonipenya/rblbp_code.zip`.

8. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012c). Rb-lbp experiments. `http://iiia.csic.es/~tonipenya/rblbp_experiments.zip`.

9. Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2014a). Chainme code and experiments. `http://iiia.csic.es/~tonipenya/chainme_code_experiments.zip`.

Finally, as a result of various collaborations and projects, during my PhD I also produced the following publications and library:

10. Penya-Alba, T., Pujol-Gonzalez, M., Esteva, M., Rosell, B., Cerquides, J., Rodriguez-Aguilar, J. A., Sierra, C., Carrascosa, C., Julian, V., Rebollo, M., Rodrigo, M., and Vasirani, M. (2011). ABC4MAS: Assembling Business Collaborations for MAS. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2011)*, number 3, pages 431–432. IEEE.

11. Penya-Alba, T., Mikhaylov, B., Pujol, M., Rosell, B., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012a). Tecnologia de subastas para la formacion automatizada de cadenas de suministro. *Novatica*, (218).

12. Penya-Alba, T., Mikhaylov, B., Pujol-Gonzalez, M., Esteva, M., Rosell, B., Cerquides, J., Rodriguez-Aguilar, J. A., Sierra, C., Carrascosa, C., Julian, V., Rebollo, M., Rodrigo, M., and Vasirani, M. (2013a). An environment to build and track agent-based business collaborations. In et al., S. O., editor, *Agreement Technologies (Law, Governance and Technology Series)*, pages 609–622. Springer.

13. Penya-Alba, T. (2013). From Supply Chain Formation to Multi-agent Coordination. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*.

14. Pujol-Gonzalez, M. and Penya-Alba, T. (2014). Binary max-sum. `https://binarymaxsum.github.io/`.

Chapter 2

Mathematical Background

In Chapter 1 we introduced the Supply Chain Formation (SCF) problem. Moreover, we argued that there is a need for automated SCF. Furthermore, we pointed out that the automated assessment of Supply Chains (SCs) in a decentralized manner is an emergent field that needs to be studied in depth. In this chapter, we review some concepts that are the building blocks for the state-of-the-art methods for decentralized SCF as well as our own contributions. On the one hand, we review periodic double auctions as a mechanism to assess the participants in a SC in which there is only one good at trade (i.e. the exchanges of a single good with multiple buy and sell offers). Periodic double auctions are used as the main component in the SCF method proposed in [Walsh and Wellman, 2003]. On the other hand, we review the max-sum algorithm as a general technique for decentralized coordination of multi-agent systems. In [Winsper and Chli, 2010], the authors exploit the max-sum algorithm for the formation of SCs.

2.1 Periodic double auctions

In this section we review periodic double auctions, sometimes termed a call market [Mccabe et al., 1990], as a mechanism to assess the participants in a SC. Periodic double auctions are well suited to assess SC participants in a scenario in which there is a single good to be exchanged between buyers and sellers. One such scenario is described in the example below.

Example 2. Consider a vintage computer market such as the one depicted in Figure 2.1. In this market, there are eight participants from which four are looking to sell an Apple Macintosh and four are looking to buy an Apple Macintosh. On the one hand, Alice, Bob, Carol, and Dave (the sellers) offer to sell an Apple Macintosh for \$2, \$3, \$4, and \$5 each. On the other hand, Eve, Frank, Gene, and Hank are willing to pay \$6, \$5, \$2, and \$1 respectively for an Apple computer.

In the vintage computer market described in Example 2, solving the SCF

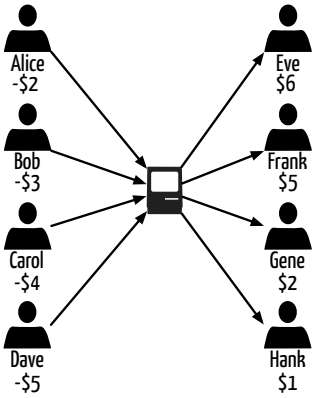


Figure 2.1: Vintage computer market described in Example 2.

equals to finding the SC configuration with maximum value. That is, finding the SC configuration in which the total surplus is maximized. The SC configuration that maximizes the total surplus is that in which Alice and Bob sell the computer to Eve and Frank for a total surplus of $6.

The SCF problem described in Example 2 can be tackled by means of a periodic double auction. Two-sided or double auctions permit multiple buyers and sellers to bid to exchange a designated commodity. In the periodic version of the double auction an auctioneer collects buy and sell bids over a specified period of time, then clears the market at the expiration of the bidding period. In order to clear the market, the auctioneer needs to determine the winners of the auction and the price at which the goods will be exchanged. The former is decided by means of a clearing rule whereas the latter is fixed by means of a pricing rule. We cover the process of deciding the winners of the auction in Section 2.1.1 and the process of fixing the price in Section 2.1.2. Finally, in Section 2.1.3 we provide an algorithm to assess both the active participants and the price along with a complexity analysis of the algorithm in terms of memory, communication and computation resources required to run a periodic double auction.

2.1.1 Clearing rule

Going back to the SCF problem described in Example 2, after collecting buy and sell bids from the participants, the auctioneer in a periodic double auction has to decide which participants will be active in the resulting SC configuration (i.e. which participants will exchange the computer). Intuitively, this can be done by matching the best (cheaper) sellers with the best buyers until the exchanges no longer produce surplus. That is, the auctioneer will first match Alice ($-\$2$) with Eve ($6) and then Bob ($-\$3$) with Frank ($5). The auctioneer will stop matching buy and sell bids at that point since the exchange between Carol ($-\$4$)

and Gene ($2) does not generate surplus. Therefore, the active participants in the resulting SC configuration will be Alice, Bob, Eve, and Frank, corresponding to the optimal solution to this SCF problem.

In general, we note the largest possible surplus as π^*. We refer to a π^*-configuration as a set of buyers and sellers that achieve surplus π^*. Alice, Bob, Eve, and Frank are *active* in the π^*-configuration and Carol, Dave, Gene, and Hank are not. The number of buyers at trade in the π^*-configuration (2 in this case) is noted as η.

In the general case, consider an auctioneer (m_g) that aims to trade good g. Let $\mathcal{S}_g$ be the set of sellers that are willing to sell g and $\mathcal{B}_g$ the set of buyers that are willing to buy g. Table 2.1 describes a general periodic double auction, with $s^1, ..., s^\eta, ..., s^{|\mathcal{S}_g|}$ being the sellers ordered descendingly by offer; $b^1, ..., b^\eta, ..., b^{|\mathcal{B}_g|}$ being the buyers ordered descendingly by offer; $\nu^g_{s^i}$ is the offer from seller s^i to the auctioneer for good g; and $\nu^g_{b^j}$ is the offer from buyer b^j to the auctioneer for good g. The buyers and sellers over the dashed line are in the π^*-configuration whilst the ones below are not.

Sellers	**Buyers**	**Fact**
s^1	b^1	$\nu^g_{b^1} + \nu^g_{s^1} > 0$
$\vdots$	$\vdots$	
s^η	b^η	$\nu^g_{b^\eta} + \nu^g_{s^\eta} \geq 0$
$s^{\eta+1}$	$b^{\eta+1}$	$\nu^g_{b^{\eta+1}} + \nu^g_{s^{\eta+1}} < 0$
$\vdots$	$\vdots$	

Table 2.1: General periodic double auction scenario.

So far we have covered how to decide the active participants in a periodic double auction. However, the auctioneer must face one more decision: to assign the price payed by buyers and received by sellers. In the following section we cover two pricing rules.

2.1.2 Pricing rules

A price rule in a periodic double auction establishes the clearing price (τ) paid by buyers and received by sellers. We focus on the (M+1)st and M-th price rules [Wurman et al., 1998] . There are some constraints that the price has to fulfil to ensure individual rationality [Fudenberg and Tirole, 1991] for the active participants (i.e. no active participant receives negative surplus) and fairness for the inactive participants (i.e. no inactive participant would receive a positive surplus if active).

To ensure individual rationality, no seller at trade should be paid less than her bid ($\tau \geq -\nu^g_{s^\eta}$) and no buyer at trade should pay more than her bid ($\tau \leq \nu^g_{b^\eta}$).

To ensure fairness the price cannot be larger than the bid of any seller that is left out of trade ($\tau \leq -\nu^g_{s^{\eta+1}}$), and it cannot be smaller than the bid of any buyer that is left out of trade ($\tau \geq \nu^g_{b^{\eta+1}}$). Thus, the clearing price τ can take any value such that $\tau^- \leq \tau \leq \tau^+$ where

$$\tau^- = \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}) \tag{2.1}$$

and

$$\tau^+ = \min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta}). \tag{2.2}$$

The rule that sets the clearing price at τ^- is known as the (M+1)st price rule, whilst the rule that sets the price at τ^+ is known as the M-th price rule [Wurman et al., 1998]. The price interval (τ^-, τ^+) is known as the bid-ask interval [Wurman et al., 1998]. Regarding Example 2, the (M+1)st price rule would set the price at $\tau^- = \max(-(-3), 2) = \$3$ (corresponding to Bob's offer), whereas the M-th price would do it at $\tau^+ = \min(-(-4), 5) = \$4$, established by Carol's offer.

2.1.3 Periodic double auction clearing algorithm

So far, we have described the rules for determining the active participants in a periodic double auction and fixing the price at which the goods could be exchanged. In what follows we describe a general algorithm to determine both the active participants in the π^*-configuration and the bid-ask interval for the price. Recall from Section 1.2.3 that we are interested in studying whether decentralized SCF can be computationally efficient. Therefore, at the end of this section we study the complexity of the algorithm for periodic double auctions since it is the basic building block of some of the methods in the state-of-the-art for SCF [Walsh and Wellman, 2003].

As we have seen in Section 2.1.1, given a SC, assessing the active participants in its π^*-configuration equals to matching the best buyers and sellers until the exchanges do not generate surplus. Furthermore, as we have seen in Section 2.1.2, determining the bid-ask interval equals to applying Equations 2.1 and 2.2 after finding the π^*-configuration. Intuitively, assessing the set of active participants in the π^*-configuration and the bid-ask interval amounts to:

1. Sorting sellers decreasingly by offer.
2. Sorting buyers decreasingly by offer.
3. Assessing the number of active participants (η) by matching buyers and sellers in order until the matches stop generating surplus, that is, until the buy offer is not able to cover the sell offer or there are no more offers.
4. Selecting the top η sellers and buyers to be active.
5. Assessing the bid-ask interval according to Equations 2.1 and 2.2.

More formally, Algorithm 1 describes the procedure followed by the auctioneer to assess the bid-ask interval (τ^-, τ^+) and the set of active buyers (B_g^a) and sellers (S_g^a) in the π^*-configuration given a set of buy offers $(\nu_B^g = \langle \nu_{b^1}^g, \ldots, \nu_{b^{|\mathcal{B}_g|}}^g \rangle)$, and a set of sell offers $(\nu_S^g = \langle \nu_{s^1}^g, \ldots, \nu_{s^{|\mathcal{S}_g|}}^g \rangle)$ for a good g.

Algorithm 1 Clearing and pricing rules for a periodic double auction.

Require: A set of buy offers (ν_B^g) and a set of sell offers (ν_S^g)
Sort sellers decreasingly by offer getting:
  $\mathcal{S} = \langle s^1, \ldots, s^{|\mathcal{S}_g|} \rangle$
Sort buyers decreasingly by offer getting:
  $\mathcal{B} = \langle b^1, \ldots, b^{|\mathcal{B}_g|} \rangle$
$\eta \leftarrow 0$ // *Assess the number of trading participants*
while $\nu_{s^{\eta+1}}^g + \nu_{b^{\eta+1}}^g \geq 0$ **do**
  $\eta \leftarrow \eta + 1$
end while
$S_g^a = \{s^1, \ldots, s^\eta\}$ // *Assess the set of active sellers at trade*
$B_g^a = \{b^1, \ldots, b^\eta\}$ // *Assess the set of active buyers at trade*
$\tau^- \leftarrow \max(-\nu_{s^\eta}^g, \nu_{b^{\eta+1}}^g)$ // *Assess the bid-ask interval*
$\tau^+ \leftarrow \min(-\nu_{s^{\eta+1}}^g, \nu_{b^\eta}^g)$
return $(S_g^a, B_g^a, \tau^-, \tau^+)$

Next, we analyze the resources necessary to run a periodic double auction. Given $P = \mathcal{S} \cup \mathcal{B}$, the set of participants interested on selling or buying good g. First, the auctioneer needs to store the buy and sell offers received from each of the participants, namely $\mathcal{O}(|P|)$ memory. Second, in a periodic double auction, participants send their offers to the auctioneer and then are notified of the bid-ask interval and wether they are active in the π^*-configuration or not. Therefore, the communication requirements are also in $\mathcal{O}(|P|)$. Finally, the costliest operation for the auctioneer is sorting the offers received from the participants which takes $\mathcal{O}(|P| \cdot \log |P|)$ operations. Hence, running a periodic double auction requires low computational resources.

2.2 Max-sum: Maximizing a function that decomposes additively

As mentioned in Section 1.2, the SCF problem has been widely studied in the multi-agent systems literature [Davis and Smith, 1983, Walsh et al., 2000, Collins et al., 2002, Walsh and Wellman, 2003, Cerquides et al., 2007, Giovannucci et al., 2008, Winsper and Chli, 2010, Mikhaylov et al., 2011, Winsper and Chli, 2013]. Max-sum [Bishop et al., 2006], a message passing algorithm that can find approximate solutions to optimization problems, has been used to solve the SCF problem in a decentralized manner [Winsper and Chli, 2012, Winsper and Chli, 2013]. Moreover, max-sum has

shown good empirical performance in a wide range of multi-agent systems coordination scenarios [Farinelli et al., 2008, Stranders, 2009, Kim et al., 2010, Rogers et al., 2011, Pujol-Gonzalez et al., 2013b]. As mentioned in Section 1.3, our contributions to decentralized SCF are based on the max-sum algorithm. Therefore, in this section we review it, showing that it can be understood as an exchange of messages over a graph.

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 $\mathbf{x}_i \in \mathcal{D}_i$. Moreover, we use $\mathbf{X}$ to refer to a possible state for each variable in X, that is $\mathbf{X} \in \mathcal{D}_X$. Given a sequence of variables $X_f \subseteq X$, a local term f is a function $f : \mathcal{D}_{X_f} \to \mathbb{R}$. We say that X_f is the scope of f, and $\mathbf{X}_f$ is a possible state for each variable in X_f. Finally, a term whose scope is a single variable is said to be a *simple* term, and a term whose scope is two or more variables is said to be a *composite* term.

A function $g : \mathcal{D}_X \to \mathbb{R}$ is said to *decompose additively* if it can be broken as a sum of local terms. That is, whenever there is a set of local terms F (referred to as the additive decomposition F) such that $g(\mathbf{X}) = \sum_{f \in F} f(\mathbf{X}_f)$. Many problems, such as decoding [Forney Jr, 1973] or finding minimal graph cuts [Tarlow et al., 2011], can be reduced to solving the problem of maximizing a function that decomposes additively.

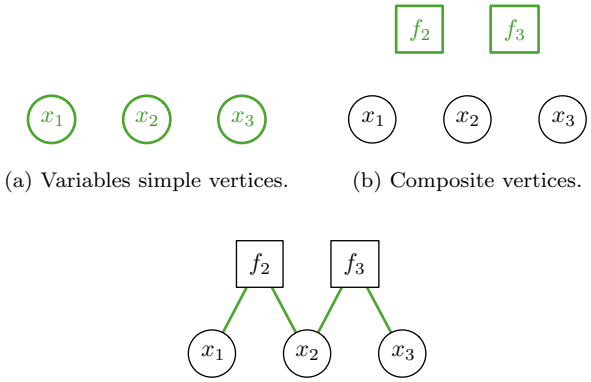
Formally, the problem of maximizing a function that decomposes additively can be expressed as follows:

$$\begin{array}{lll} \text{maximize} & g(\mathbf{X}) = \sum_{f \in F} f(\mathbf{X}_f) & \\ \text{subject to} & \mathbf{x}_i \in \mathcal{D}_i. & \forall i \in \{1, \ldots, n\} \end{array}$$

Example 3. Consider a function g that takes variables x_1, x_2, and x_3 as parameters. Moreover, consider the domains of x_1, x_2, and x_3 to be $\mathcal{D}_1 = \{0, 1\}$, $\mathcal{D}_2 = \{0, 1, 2\}$, and $\mathcal{D}_3 = \{1, 2\}$ respectively. Finally, consider that the value of function g is assessed by $g(x_1, x_2, x_3) = x_1 \cdot x_1 + x_1 \cdot x_2 - x_2 \cdot x_3$.

It is easy to see that function g described in Example 3 can be decomposed additively in three local terms. Taking terms $f_1(\mathbf{x_1}) = x_1 \cdot x_1$, $f_2(\mathbf{x_1}, \mathbf{x_2}) = x_1 \cdot x_2$, and $f_3(\mathbf{x_2}, \mathbf{x_3}) = -(x_2 \cdot x_3)$ with scopes $X_{f_1} = \{x_1\}$, $X_{f_2} = \{x_1, x_2\}$, and $X_{f_3} = \{x_2, x_3\}$ we obtain g's additive decomposition $F = \{f_1, f_2, f_3\}$. Notice that the problem of maximizing function g has more than one optimal solution. One of these optimal solutions is $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = 0$, $\mathbf{x}_3 = 2$, with a value of 1, that is $g(1, 0, 2) = 1$.

Max-sum provides an approximate solution for the problem of maximizing a function that decomposes additively in three steps. First, it maps the problem into a structure called local term graph. Then, it iteratively exchanges messages between the vertices of that graph. Finally, it determines the states of the variables. In the following sections, we review each of the three steps.



(a) Variables simple vertices. (b) Composite vertices.

(c) Edges between vertices.

Figure 2.2: Local term graph for the additive decomposition of g in Example 3.

2.2.1 From an additively decomposable function to a local term graph

The first step that max-sum performs to solve the problem of maximizing a function g that decomposes additively is mapping an additive decomposition F of g into a graph known as the local term graph. The local term graph G_{LT} is simply a specialization of the local domain graph defined in [Aji and McEliece, 2000] to the max-sum semiring. Each vertex of the G_{LT} is, as its name suggests, a local term in an additive decomposition F of our objective function g. An edge between two terms in the G_{LT} means that these two terms share one variable and that they are willing to exchange information about the variable they share. For each vertex $v \in G_{LT}$, we note f_v its associated term. We refer to vertices associated with simple terms as *simple vertices* and to vertices associated to composite terms as *composite vertices*.

Figure 2.2 illustrates the process of mapping the additive decomposition of function g in Example 3 into a G_{LT}. In this illustration we use circles to depict simple vertices and squares to depict composite vertices. First, a simple vertex is created for each of the variables (Figure 2.2a). Since f_1 is a simple term that only depends on x_1 it is included in the simple vertex for variable x_1. Second, a composite vertex is created for each of the composite terms, that is, f_2 and f_3 (Figure 2.2b). Finally, each composite vertex is connected to the simple vertices of each of the variables in its scope (Figure 2.2c). That is, the composite vertex associated to f_2 is connected to the simple vertices for variables x_1 and x_2. Similarly, the composite vertex associated to f_3 is connected to the simple vertices for variables x_2 and x_3.

In general, the local term graph used by max-sum is built from F as follows. First, for each variable x_i, a simple vertex is created, associating with it a term (f_{x_i}) that is the addition of every simple term in F whose scope is x_i. Second,

for each composite term f_j, a composite vertex is created. With a slight abuse of notation, we label x_i the simple vertex associated to variable x_i and f_j the composite vertex associated to composite term f_j. Finally, each composite vertex is connected to the simple vertex of each of the variables in its scope. Notice that, since composite vertices are only connected to simple ones and vice versa, any pair of connected vertices in this graph share a single variable, the one corresponding to the simple vertex. Furthermore, notice that the G_{LT} is a bipartite graph with two disjoint sets corresponding to the set of simple vertices and the set of composite vertices.

2.2.2 Exchanging messages on a local term graph

After an additive decomposition of the function to maximize has been mapped into a local term graph, max-sum proceeds by iteratively exchanging messages over that local term graph. The procedure followed by each vertex v of the local term graph, is a particular case of iterative GDL (see section VII in [Aji and McEliece, 2000]). Each vertex of the local term graph is in charge of receiving messages from its neighbors, composing new messages and sending them to its neighbors. Recall from Section 2.2.1 that in a G_{LT} any two vertices connected by an edge share a single variable. Therefore, there will be messages exchanged from simple vertices x_i to composite vertices f_j and vice versa. The message exchanged between a pair of vertices is a vector of real numbers, one for each possible state of the variable shared by both vertices. The exchange of messages continues until a convergence criterion is met.

The message sent from a composite vertex f to its neighbor x (μ_f^x) is assessed from the messages previously received by f as follows

$$\mu_f^x(\mathbf{x}) = \max_{\mathbf{X_f}\setminus\{\mathbf{x}\}} \left\{ f(\mathbf{X}_f) + \sum_{x' \in N(f)\setminus\{x\}} \mu_{x'}^f(\mathbf{x}') \right\} \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{2.3}$$

where $\mathbf{X}_{f\setminus\{x\}}$ can take every possible state of every variable in the scope of local term f except x; f is the local term associated to the vertex, and $N(f) \setminus \{x\}$ is the set of neighbors of f excluding x.

On the other hand, the message sent from a simple vertex x to its neighbor f (μ_x^f) is assessed from the messages previously received by x as follows

$$\mu_x^f(\mathbf{x}) = f_x(\mathbf{x}) + \sum_{f' \in N(x)\setminus\{f\}} \mu_{f'}^x(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{2.4}$$

where f_x is the simple term associated to x and $N(x)\setminus\{f\}$ is the set of neighbors of x excluding f. Next, we describe the algorithm followed by each of the vertices in the G_{LT}.

Initially, each vertex v will initialize the message from each of its neighbors w to zeros. After that, for each neighbor w, it will assess message μ_v^w according to Equations 2.3 or 2.4, send the message to the corresponding neighbor, and

Algorithm 2 Algorithm run by any vertex v of the local term graph

1: For each vertex w neighbor of v, initialize message μ_w^v to zeros.
2: **while not** convergence **and not** maximum number of iterations **do**
3: Assess message μ_v^w for each vertex w neighbor of v.
4: Send message μ_v^w to each vertex w neighbor of v.
5: Receive message μ_w^v from each vertex w neighbor of v.
6: **end while**

receive the message μ_w^v from vertex w. The procedure, given by Algorithm 2, is repeated until convergence or a maximum number of iterations is reached.

Max-sum is said to have converged after none of the messages change from one iteration to another [Koller and Friedman, 2009]. A slightly less stringent criterion for convergence is to stop max-sum after the preferred states for the variables do not change from one iteration to another [Farinelli et al., 2008]. This second criterion is useful for instances in which the preferred state of the variables converges but the messages marginally change at each iteration.

When the local term graph is a tree, max-sum is guaranteed to converge to the optimal configuration [Weiss, 2000]. Otherwise, if the local term graph contains cycles, the max-sum message exchange may converge to an approximation and even fail to converge. Yet, if max-sum converges, it is known to provide neighborhood maximum configurations [Weiss and Freeman, 2001, Vinyals et al., 2010]. For instance, in local term graphs with a single cycle, the neighborhood maximum is the global maximum and so max-sum is optimal in this case.

2.2.3 Determining the states of the variables

After the message exchange ends, it is necessary to determine the states of the variables. That is, for each variable x of the function to optimize we need to choose a state from its domain $\mathcal{D}_x$. There are different strategies for fixing the states of the variables. Here, we will review the most commonly used one: *independent variable marginals*. With this approach, decisions on the state of each variable are simultaneously taken by each simple term. These decisions can be taken independently for each variable since each variable is represented by only one simple term. The assignment of the state for a variable x in its single term that maximizes its local value is then assessed as:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \left\{ f_x(\mathbf{x}) + \sum_{f \in N(x)} \mu_f^x(\mathbf{x}) \right\}. \tag{2.5}$$

Assigning states this way is fast and simple. However, it could be the case that the decisions taken by each of the vertices do not fit together. For instance, take function g defined in Example 3. The assignment $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = 0$, $\mathbf{x}_3 = 2$ is an optimal one with value $g(1, 0, 2) = 1$. However, the assignment $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = 1$, $\mathbf{x}_3 = 1$ is also optimal and has value $g(1, 1, 1) = 1$. The values assessed by Equation 2.5 will be equal for both states of variable x_3, but selecting the

values for variables $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = 1$, $\mathbf{x}_3 = 2$ independently will result in a suboptimal solution with value $g(1,1,1) = 0$. Thus, some approaches such as [Givoni and Frey, 2009, Pujol-Gonzalez et al., 2013b] determine the values of the variables in a domain specific manner.

2.2.4 Reducing max-sum computational complexity

So far we have described max-sum as an algorithm to find approximate solutions to an optimization problem; as long as it can be represented by a function that decomposes additively. Unfortunately, since assessing the message in Equation 2.3 takes exponential time in the number of variables, max-sum might perform poorly in terms of computation in problems that contain local terms with large scopes. However, studying the particularities of a specific composite local term can lead to reduce the complexity of assessing its outgoing messages [Tarlow et al., 2010]. The following example illustrates this reduction.

Example 4. Consider the sequence of N-ary variables $X = \langle x_1, \ldots, x_n \rangle$, with each of the variables sharing the same domain $\mathcal{D} = \{1, \ldots, N\}$. Moreover, take the composite local term f with scope $X_f = \langle x_1, \ldots, x_n \rangle$ that takes value 0 whenever all the variables share the same state and $-\infty$ otherwise. Formally,

$$f(\mathbf{x_1}, \ldots, \mathbf{x_n}) = \begin{cases} 0, & \text{if } \mathbf{x_i} = \mathbf{x_{i+1}} \quad 1 \leq i < (n-1) \\ -\infty, & \text{otherwise.} \end{cases}$$

Assessing the max-sum message from local term f to a variable x_i in its scope for state $\mathbf{x_i}$ ($\mu_f^{x_i}(\mathbf{x_i})$) using Equation 2.3 would take exponential time on the number of variables in f's scope. However, if we take a closer look at how this message is computed and the particularities of local term f we can reduce this complexity. Since we are assessing the message for variable's x_i state $\mathbf{x_i}$, any assignment to variables $X_f \setminus \{x_i\}$ other than assigning all the variables to $\mathbf{x_i}$ will result in a message containing a value of $-\infty$. Therefore, in order to maximize the expression in Equation 2.3, all variables in $X_f \setminus \{x_i\}$ must take value $\mathbf{x_i}$. Hence, we can compute the message from local term f to any variable x_i in its domain for a state $\mathbf{x_i}$ as the addition of the messages received by f from its neighboring variables for that same state. More formally, Equation 2.3 for local term f can be assessed as:

$$\mu_f^{x_i}(\mathbf{x_i}) = \sum_{x' \in N(f) \setminus \{x_i\}} \mu_{x'}^{f}(\mathbf{x_i}).$$

Thus, reducing the complexity of assessing the message from local term f to a neighbor x_i from exponential to linear.

In [Givoni and Frey, 2009], the authors provide the first simplified expressions to assess the outgoing messages from composite local terms. Later on, the works in [Tarlow et al., 2010, Pujol-Gonzalez et al., 2013a, Pujol-Gonzalez et al., 2013b] expand the collection of local terms that can have

their output messages assessed in linear (or log-linear) time. These simplifications are usually performed over local terms that operate over binary variables only. However, N-ary variables can be represented by converting variables with N states into N binary variables with a 1-of-N constraint [Tarlow et al., 2010]. Moreover, since each variable is binary, it may appear from Equations 2.3 and 2.4 that the vertices must exchange two-valued messages. Therefore, one might think that introducing additional variables comes at the cost of additional messages being passed since for each possible state of a variable two values would be sent instead of one. In practice, for any message μ_v^w, vertices can just exchange a single-valued message ν_v^w representing the difference between the two values for its possible settings. Formally,

$$\nu_v^w = \mu_v^w(1) - \mu_v^w(0). \tag{2.6}$$

Thus, preserving the amount of information exchanged between vertices. The original two-valued message can be recovered from this scalar up to an additive constant. For instance, we can recover μ_v^w from ν_v^w by defining $\mu_v^w(1) = \nu_v^w$ and $\mu_v^w(0) = 0$. However, this is not important because adding a constant to all values of a message does not alter the output of the algorithm [Givoni and Frey, 2009].

The technique described above will be key in the development of our contributions to decentralized SCF both in peer-to-peer and mediated SCF. That is, in Chapters 4 and 5 we will introduce new types of local terms whose message calculation can be simplified in order to build computationally efficient methods for decentralized SCF.

2.3 Conclusion

In this chapter we have reviewed periodic double auctions as a mechanism to assess the participants in a SC in which there is only one good at trade. Moreover, we have reviewed the max-sum algorithm as a method for approximating optimization problems in a decentralized manner. Both of these tools will be key in understanding the state-of-the-art methods for SCF reviewed in the next chapter. Finally, we have reviewed a technique to reduce the time complexity required by the max-sum algorithm to operate. This technique will be the base of the methods for decentralized SCF that we introduce in later chapters.

Chapter 3

Related Work

The Supply Chain Formation (SCF) problem has been widely studied by the multi-agent systems community. Numerous contributions can be found in the literature where participants are represented by computational agents (e.g. [Davis and Smith, 1983, Walsh et al., 2000, Collins et al., 2002, Walsh and Wellman, 2003, Cerquides et al., 2007, Giovannucci et al., 2008, Winsper and Chli, 2010, Mikhaylov et al., 2011, Winsper and Chli, 2013]). These computational agents act in behalf of the participants during the SCF process [Norman et al., 2004]. Moreover, the agents representing the participants interact with each other for a period of time after which the new Supply Chain (SC) is formed. By employing computational agents it is possible to form SCs in a fraction of the time required by the manual approach [Winsper, 2012].

As discussed in Chapter 1, SCF methods can be classified in three categories depending on the architecture they follow. A first division is to separate SCF into centralized and decentralized architectures. Furthermore, we can separate the decentralized methods into two further categories depending on whether the communication between participants is either direct or mediated. In this chapter we briefly review the contributions within these three categories and provide a description of the state-of-the-art methods for each of them.

3.1 Centralized approaches

A compelling approach to address the SCF problem is to employ a centralized method [Norman et al., 2004]. Unsurprisingly, the majority of the contributions in the literature follow a centralized approach (e.g. [Walsh et al., 2000, Collins et al., 2002, Cerquides et al., 2007, Giovannucci et al., 2008, Mikhaylov et al., 2011]). In a centralized approach, participant agents inform a central authority of their preferences (encoded as offers). After collecting the offers of all participant agents, the central authority determines the resulting SC. Subsequently, the aforementioned central authority notifies the participants whether they are active in the resulting SC and the

terms, such as prices, and with whom they will exchange goods.

Combinatorial Auctions (CAs) [Cramton et al., 2006] are a common mechanism to solve the SCF problem because they are well suited to deal with complementarities between goods. This sort of approaches make use of an auctioneer that acts as the central authority of the system. After collecting the bids from the participants, the auctioneer determines the SC of maximum value and notifies the participants if they are active and the terms of the exchanges. In [Walsh et al., 2000], the authors propose one such method of centralized SCF by means of a CA. Moreover, the authors study the economic impact that participants bidding strategically have in the SC. In [Collins et al., 2002], the authors introduce a model capable of dealing with temporal and precedence constraints. Thus, providing solutions that represent a schedule for the execution of the SC. Later, in [Cerquides et al., 2007], a new bidding language is introduced that increases the expressiveness of standard CAs allowing to express bids that contain multiple copies of the same transformation, transformations that take (or produce) multiple units of the same good, offer different bundles of transformations, and different prices for a transformation depending on how many times it is performed (bulge discounts). In [Giovannucci et al., 2008], the authors provide the means to reduce the computational complexity of solving the Winner Determination Problem of the previous approach through a formal analysis of the topology of the problem. Recently, in [Mikhaylov et al., 2011], the authors propose to solve the SCF problem in a sequential manner. Therefore, the auctioneer first accepts bids for her required goods and, in subsequent iterations, she accepts bids for the required goods of previous iterations. Regardless of their particularities, all the methods described above solve the Winner Determination Problem in a similar manner which we describe next.

3.1.1 Solving the Supply Chain Formation problem

Solving the Winner Determination Problem (WDP) with CAs is usually achieved by means of an Integer Program (IP). The IP receives as input the participants' bids, a set of goods the auctioneer expects to end up with, and a set of the goods that are readily available for the auctioneer. Moreover, the output of the IP is a set of transformations to be executed that correspond to the bids from the participants to be accepted. In the case the IP formulation takes into account the order in which the tasks must be performed, the output would be a sequence of transformations to execute in order to produce the required output [Collins et al., 2002, Cerquides et al., 2007]. Next, we analyze the complexity of the method introduced in [Cerquides et al., 2007].

The WDP proposed in [Cerquides et al., 2007] has a binary decision variable for each task submitted by a participant and position on the solution. Moreover, since the solution is a sequence of transformations to be executed, and all the transformations can be potentially selected to be accepted, there would be as many positions on the solution as tasks in the SC. This encoding is used in order to cope with cyclic transformations (those that produce goods that are used in previous levels of the SC). Therefore, in a problem with N transformations, there

would be N^2 binary variables (one for each transformation-position). Thus, the number of variables in the IP grows quadratically with the number of transformations which hinders its scalability [Cerquides et al., 2007]. Moreover, a valid solution must comply with a set of constraints over the bids and the accepted transformations. That is, the IP needs to enforce, by means of constraints, that it is possible to obtain the requested goods from the goods that are initially available by applying the transformations in the solution in the given order while respecting the bids submitted by the participants. Although it is possible to significantly reduce the number of variables and constraints through formal analysis of the topology of the problem, solving the WDP by means of IP still turns out tot be impractical in high complexity scenarios [Giovannucci et al., 2008].

3.1.2 Analysis

Although solving the SCF in a centralized optimal manner seems to be appealing, it might not be the best strategy [Hayek, 1945]. In general, there are several arguments against solving the SCF problem in a centralized manner and in favor of a decentralized approach. First, centralized approaches rely on a central authority that possesses the private valuations of all agents and determines a feasible allocation. By doing so, these approaches need for all participants to fully trust the central authority. However, there are scenarios where that assumption can not be taken so easily (*e.g.* temporal coalitions among companies without the intervention of a third party mediator). Therefore, due to the decentralized nature of the problem, no central entity might have the allocative authority to perform such operation [Walsh and Wellman, 2003]. Second, even finding a feasible configuration is an NP-HARD problem [Walsh and Wellman, 2000, Fionda, 2009]. Therefore, the SCF problem in large markets results in complex optimization problems [Babaioff and Nisan, 2004] which might render impossible to solve the problem in an exact manner due to computational constraints. Moreover, these methods can also have scalability issues due to their centralized nature. Third, the existence of a centralized entity introduces a single point of failure in the system. Therefore, certain SCF problems (specially those related to large markets) are better addressed by avoiding the use of a central authority. In the next section, we cover the state of the art in decentralized SCF and analyze whether these solutions are economically (Question 1) and computationally (Question 2) efficient.

3.2 Decentralized approaches

Decentralized SCF appears as an alternative to centralized SCF in order to overcome some of its limitations. A special area of interest when facing the SCF problem in a decentralized manner is that of how to distribute the knowledge [Hayek, 1945]. Generally, participants in the SCF process are represented by computational agents [Norman et al., 2004] usually referred to as participant agents. These participant agents act on behalf of the participants during the

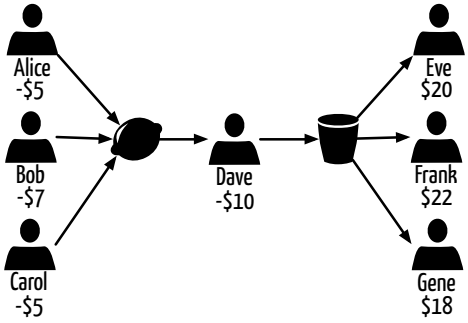


Figure 3.1: Lime juice industry SC described in Example 5.

SCF process. However, one must still decide how the participant agents will interact with each other in order to determine the active participants in the SC and the terms of the exchanges.

As discussed in Section 1.2.2, a first approach to decentralized SCF considers peer-to-peer (P2P) communication. In P2P SCF, participant agents only interact with the participant agents representing their suppliers and consumers, this approach is followed by [Winsper and Chli, 2010, Winsper and Chli, 2013]. A decentralized alternative to P2P SCF is that of mediated SCF. In this setting, participant agents resort to local markets in which the goods they want to sell or buy are being traded [Walsh and Wellman, 2000, Walsh and Wellman, 2003]. In the following sections we elaborate on this distinction and review the state-of-the-art for both P2P SCF and and mediated SCF.

3.2.1 Peer-to-peer Supply Chain Formation

In P2P approaches to SCF, each participant agent communicates directly with the participant agents representing its potential buyers and sellers. That is, the participant agents representing participants providing the goods it requires as inputs and the participant agents representing participants consuming the goods it produces. Therefore, the SCF process takes place between participant agents with no intervention of any third party. Take, for instance, the example of a local lime juice industry described bellow.

Example 5. The participants in the SC for our local lime juice industry are depicted in Figure 3.1. In this setting, we have 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 Gene), each aiming at buying a liter of juice at a given price: Eve offers $20, Frank $22, and Gene $18.

In this setting, lime producers Alice, Bob, and Carol will negotiate with

Dave who will negotiate with Eve, Frank, and Gene. This communication will take place directly between the participants without the intervention of any third party. Next, we describe the state-of-the-art method for P2P SCF, namely Loopy Belief Propagation [Winsper and Chli, 2013], which is based on the max-sum algorithm described in Chapter 2.

Loopy belief propagation

In [Winsper and Chli, 2010], the authors introduce Loopy Belief Propagation (LBP) as a method to solve the SCF in a decentralized manner with P2P communication. 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. Thus, the authors offer the means of converting a SCF problem into a local term graph, as defined in Section 2.2, on which max-sum can operate.

In LBP, the SCF problem is represented by a model in which each of the participants' decisions is encoded in single variable. In Example 5, Alice, Bob, Carol, Dave, Eve, Frank, and Gene decisions would be encoded in variables x_A, x_B, x_C, x_D, x_E, x_F, and x_G. The states of each variable encode the individual decisions that the participant needs to make regarding her exchange relationships plus an inactive state. Moreover, the activation cost for a participant p is encoded by means of a simple term f_p, also called *activation term*. Each of these activation terms has the participant's variable as its scope and takes value zero for the inactive state and the activation cost for any of the active states. For instance, take x_A, x_D, and x_E, the variables encoding Alice, Dave, and Eve in Example 5. Table 3.1 lists the possible states each of these variables can take as well as the value of their activation terms. For Alice's variable, there are two possible states (either sell to Dave or remain inactive) and the activation term takes on value -5 (Alice's activation cost) for the active state and 0 for the inactive state. Notice that the states of x_D (from σ_2 to σ_{11}) encode all possible exchange relationships for Dave.

In order to ensure that decisions are consistent among participants, in LBP, there is a *compatibility term* for each pair of variables representing potential partners. A compatibility term $f_{p_1p_2}$ encodes the compatibility between the decisions of the two participants p_1 and p_2. Two participants are in incompatible states whenever one of them is willing to trade with the other, but the other one does not. Consider participant variable x_D, its state σ_2 is compatible with x_A's state σ_0, but it is incompatible with x_A's σ_1 (Alice does not provide lime to Dave!). If two states are compatible, the value of the compatibility term is zero, otherwise is negative infinity. Thus, considering x_A and x_D, $f_{AD}(\sigma_0, \sigma_2) = 0$ and $f_{AD}(\sigma_1, \sigma_2) = -\infty$.

To summarize, LBP maps the SCF problem into a set of participant variables $X = \{x_1, \ldots, x_N\}$, a set of activation terms $F_A = \{f_1, \ldots, f_N\}$, one per variable, and a set F of compatibility terms. Then, solving the SCF problem amounts to finding a state assignment for the participant variables in X that maximizes the

x_A

value	semantics	$f_A(\mathbf{x_A})$
σ_0	sell to Dave	-5
σ_1	don't sell	0

x_D

value	semantics	$f_D(\mathbf{x_D})$
σ_2	buy from Alice, sell to Eve	10
σ_3	buy from Alice, sell to Frank	10
σ_4	buy from Alice, sell to Gene	10
σ_5	buy from Bob, sell to Eve	10
σ_6	buy from Bob, sell to Frank	10
σ_7	buy from Bob, sell to Gene	10
σ_8	buy from Carol, sell to Eve	10
σ_9	buy from Carol, sell to Frank	10
σ_{10}	buy from Carol, sell to Gene	10
σ_{11}	don't buy, don't sell	10

x_E

value	semantics	$f_E(\mathbf{x_E})$
σ_{12}	buy from Dave	20
σ_{13}	don't buy	0

Table 3.1: Example of states (and values) of agent variables.

following reward function:

$$\mathcal{R}_{\text{LBP}}(\mathbf{X}) = \sum_{x_i \in X} f_i(\mathbf{x_i}) + \sum_{f_{kl} \in F} f_{kl}(\mathbf{x_k}, \mathbf{x_l}). \tag{3.1}$$

Notice that the expression obtained in Equation 3.1 can be decomposed additively. Therefore, it can be mapped into a local term graph over which max-sum can operate in order to find a solution to the SCF problem. Applying the steps described in Section 2.2.1 to the SCF problem illustrated in Example 5 we obtain the local term graph depicted in Figure 3.2. Recall that, for each composite term f_c a composite vertex (depicted by a box) is created and labeled f_c. Moreover, the simple terms which scope is a variable x are represented by a simple vertex (depicted by a circle) and labeled x. In Figure 3.2, each dashed box represents the local terms each of the participant agents needs to be aware of. Note that to determine their compatibility, both Alice and Dave need to keep a copy of compatibility term f_{AD}. Moreover, Alice's participant agent must know both x_A simple term and f_{AD} composite term. On the other hand, Dave's participant agent needs to be aware of x_D simple term and all of f_{AD}, f_{BD}, f_{CD}, f_{DE}, f_{DF}, and f_{DG} composite terms.

Once the SCF problem has been mapped into a local term graph, max-sum can be readily applied to find a solution. In LBP participant agents exchange messages with their potential partners following Equations 2.3 and 2.4. Convergence in LBP occurs when all the participant agents find that their preferred state is the same as in the previous iteration of the algorithm. The preferred state of each variable is determined using independent variable marginals as discussed in

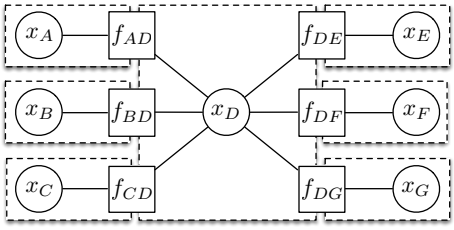


Figure 3.2: Local term graph encoding the lime juice industry SC described in Example 5.

Section 2.2.3. The preferred state extracted using these methods may produce unfeasible SCs, thus LBP includes a final phase in which unfeasible parts of the solution are pruned.

Recall from Section 1.2.3 that we are interested in finding out whether decentralized SCF can be computationally efficient (Question 2). With that goal in mind, in what follows, we provide a computational complexity analysis of LBP.

Example 5 makes us wonder about LBP's memory and communication requirements. Notice that participant variable x_D requires $3^2 + 1$ states and each of Dave's compatibility term requires $2 \cdot (3^2 + 1)$ entries (the product of the number of states of the two participants). If Dave had another input good provided by three other participants, x_D would require $3^3 + 1$ states and compatibility terms with $2 \cdot (3^3 + 1)$ entries. In general, the memory requirements for a participant agent in LBP exponentially grow with the number of goods and neighboring participants. Notice that Equations 2.3 and 2.4 indicate that the size of messages between compatibility and activation terms is as large as the number of states that the participant variable they share can take. In figure 3.2, LBP would employ messages of size $3^2 + 1$ to and from simple term x_D.

From this discussion follows that in markets with high degrees of competition (where goods are either produced or consumed by a wealth of participants), the resulting local term graph is highly demanding in terms of memory, communication, and computational requirements. Next, we assess some upper bounds on the amount of computation, memory, and bandwidth required by LBP at each iteration of the max-sum algorithm. We assume that there are n participants, each participant is connected to at most G goods, and each good is connected to at most P participants. Hence, a participant has at most $G \cdot P$ potential partners. Therefore, the requirements are:

Computation. The costliest operation for a participant agent is to assess the messages from its compatibility terms to the neighboring agents' activation terms. Assessing the values for each compatibility term takes $\mathcal{O}(P^{2G})$ operations. Moreover, each participant is in charge of $G \cdot P$ compatibility terms. Therefore, each of the participants needs to perform $\mathcal{O}(G \cdot P^{2G+1})$ operations on each max-sum iteration.

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

Communication. The messages between a compatibility term and an activation term are of size $\mathcal{O}(P^G)$, the number of states the shared variable can take. Since each participant agent shares compatibility terms with at most $G \cdot P$ neighbors, it consumes $\mathcal{O}(G \cdot P^{G+1})$ bandwidth. Finally, since we consider n participants in the SC, LBP requires $\mathcal{O}(n \cdot G \cdot P^{G+1})$ bandwidth overall per iteration.

Even though LBP is the first P2P approach for decentralized SCF, the work in [Winsper and Chli, 2010, Winsper and Chli, 2013] has some drawbacks. First, the exponential resource requirements of LBP, particularly in markets with high degrees of competition, significantly hinder its scalability. Furthermore, regarding privacy, the message each agent receives from a trading partner also contains information about her competitors. For example, in Figure 3.2 Dave's participant agent would send a message to Alice's that contains all his states and thus Alice's participant agent would be aware of the existence of other lime sellers.

An alternative to P2P communication between participant agents is the introduction of local markets. In this setting, the market for each of the goods in the SCF problem is represented by a computational agent that mediates between the participant agents interested in buying or selling the good it is in charge of. The following section covers these approaches for mediated SCF.

3.2.2 Mediated Supply Chain Formation

Decentralized SCF approaches that employ mediators for goods aim at alleviating part of the responsibility held by the participants in P2P SCF. In order to do so, the concept of local markets is introduced as described in Section 1.2.2 architecture. In this setting, a local market is created for each of the goods at trade in the SC. Moreover, for each local market there is a mediator agent (mediator henceforth) that acts as intermediary for the good at trade in its market. Each of the participant agents has access only to the local markets of the goods it is interested in buying or selling [Kalin, 2000]. The introduction of mediators removes the need for direct communication between participant agents, thus participant agents can reach more potential partners [Kalin, 2000] without incurring in any extra cost in terms of communication or computation. Notice that the mediators can be made profitable in a variety of ways such as charging membership or transaction fees or advertising. Further, more buyers and sellers for a good means that higher profit can be achieved by the mediators even if the fees are low [Kalin, 2000].

In [Walsh and Wellman, 2000], the authors model the SCF problem as a satisfiability problem and propose a method to solve it in a decentralized manner using local markets and mediators. However, solving the SCF problem using this method can turn out to be prohibitively slow even on small

problems [Walsh and Wellman, 2000]. In order to cope with the poor performance of the work in [Walsh and Wellman, 2000], the authors propose a new method in [Walsh and Wellman, 2003]. Although using the same representation as [Walsh and Wellman, 2000], the new method is based on periodic double auctions which turns out to speed up the SCF. In what follows we describe this periodic double auction based method for mediated SCF.

Sequential ascending auctions with simple bidding

In [Walsh and Wellman, 2003], the authors introduced the Simultaneous Ascending (M+1)st Price with Simple Bidding protocol (noted as SAMP-SB henceforth). SAMP-SB aims at solving the SCF problem in a decentralized manner with the use of mediators. In this approach participant agents, rather than exchanging information with each other, exchange information with a mediator agent. In SAMP-SB there is a mediator agent for each of the goods at trade and participant agents communicate exclusively with the mediators of the goods they are interested in. Similarly, mediators only communicate with the participant agents interested in buying or selling the good they are mediating. The SAMP-SB protocol is composed of an auction mechanism along with some bidding policies. In what follows we outline the auction mechanism along with the bidding policies.

A SAMP-SB mechanism comprises a set of parallel auctions, one per good. Each auction is run by a different agent, who plays the role of *mediator* for the good. Each auction runs independently of the other auctions in the SC. However, all auctions run simultaneously. Given a good g, each participant agent interacts with the mediator of the good, m_g, by submitting its offers to buy or sell g. For instance, in Example 5, Dave's participant agent would send a sell offer message to the juice mediator m_J to sell juice and a buy offer message to the lime mediator m_L to buy limes.

Each auction is an increasing periodic double auction (see Section 2.1) with price quotes. When a mediator receives a new bid, it sends each of its bidders a price quote specifying the bid-ask interval (τ^-, τ^+) that would result if the auction ended in the current bid state. The price quote also reports to each participant agent whether it is winning or not. The price quotes are not issued until all initial bids are received, but are subsequently issued immediately on receipt of new bids.

When a participant agent receives a notification from a mediator it replies following a simple, non-strategic bidding policy. Thus, bidding behaviour is purely reactive. The SAMP-SB bidding policies require that, for each auction, the prices of a participant's agent successive buy offers increase by no less than some (generally small) positive number δ_b and the prices of successive sell offers increase by no less than δ_s. Inaction leaves previous bids standing in an auction. Specifically, SAMP-SB distinguishes between participants that produce no output good (consumers) and participants that produce output goods (producers) regardless of whether they have input goods or not. In Example 5, Alice, Bob, Carol, and Dave are said to be producers (they have an output good),

whereas Eve, Frank, and Gene are said to be consumers. The bidding policies for consumers and producers are described bellow.

Consumer bidding policy. The participant agent of a consumer c not winning its input good g will bid by increasing the (M+1)st price (τ^-) by the minimum required increment δ_b (see Equation 3.2). The bid is issued whenever the consumer's gain is non-negative ($C_c - \tau_g^- - \delta_b \geq 0$), otherwise it will stop bidding. [1]

$$\nu_c^g = \tau_g^- + \delta_b. \tag{3.2}$$

Producer bidding policy. Every time the participant agent of a producer p receives a quote, if it is currently winning the auction for its output good and losing the auction for some input good g, it increases its last offer for g by the minimum required increment (δ_b). [1]

$$\nu_p^g = \nu_p^g + \delta_b. \tag{3.3}$$

Furthermore, if the quote is coming from an input good, p's participant agent updates its offer for its output good g (ν_p^g) as[2]:

$$\nu_p^g = \max(\nu_p^g + \delta_s, C_p + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \hat{\tau}_{g'}) \tag{3.4}$$

where $\hat{\tau}'_g$ stands for the perceived cost of input good g' and $\mathcal{N}(p)$ for the goods participant p is interested on buying or selling. If p is currently winning g', $\hat{\tau}'_g$ is τ^-, otherwise $\hat{\tau}'_g = \max(\tau^+, \tau^- + \delta_b)$.

Bidding continues until all messages have been received, no participant agent chooses to revise its bids, and no auction changes its prices, or allocation. At this point, the auctions clear; each bidder is notified of the final prices and how many units she transacts per good. However, SAMP-SB may converge to solutions in which some participants obtain a negative utility [Walsh and Wellman, 2003]. In order to overcome this problem, the authors propose to include a final phase that allows participant agents to decommit. Therefore, the resulting method (named SAMP-SB-D) achieves higher economical efficiency than the original SAMP-SB.

With this description of the algorithm we can already assess the upper bounds on the amount of computation, memory and bandwidth required by the agents in SAMP-SB. Notice that, in SAMP-SB there is no concept of global iteration as there was in LBP since the auctions run simultaneously with no synchronization. However, each agent (be it a participant agent or a mediator) needs to perform some steps each time it receives a message from a neighbor. That is, participant agents must reevaluate and send their bids to mediators and mediators must clear the auction and send the new prices to participant agents. We assume that

[1] Initial offers submitted for inputs goods are set to 0.

[2] The participant agent for a producer places its first output offer only after receiving the first notification for all its inputs.

there are n participants, each participant is connected to at most G goods, and each good is connected to at most P participants. Therefore, the requirements are:

Computation. Recall that the a producer's participant agent that is winning its output good updates its bids for any input good it is losing. Moreover, updating a bid has constant complexity and the number of input goods a participant is connected to is G. Therefore, in the worst case, updating a participant agent's bids will require $\mathcal{O}(G)$ operations. On the other hand, the costliest operation for a mediator is sorting the bids. However, recall that each new bid triggers a clearing of the auction. Thus, mediators need only to worry about inserting new bids in an already sorted set, which can be achieved with $\mathcal{O}(\log P)$ operations [Wurman et al., 1998].

Memory. On the one hand, participant agents need to keep track of the bid-ask interval for each of the goods they are interested in. Therefore, each participant agent requires $\mathcal{O}(G)$ memory. On the other hand, mediators need to keep track of the latest bids received from each of the participants it is connected to. Thus, each mediator requires $\mathcal{O}(P)$ memory.

Communication. In the worst case, a participant agent may have to send new bids to the mediators in charge of all of its input goods. Therefore, a participant agent may incur in $\mathcal{O}(G)$ cost in terms of bandwidth usage. On the other hand, each time a mediator clears the auction it notifies the outcome to the participant agents it is connected to, thus $\mathcal{O}(P)$.

Although SAMP-SB-D can achieve high economic efficiency [Winsper and Chli, 2013], it incurs in a high penalty in terms of communication [Winsper and Chli, 2013]. This penalty in communication comes from the fact that mediators need to re-evaluate and communicate the winners of the running auctions each time they receive a new bid. Moreover, since the updates on the bids made by the participant agents follow a fixed increment, finding the solution can turn out to be slow (for small increments and high prices) or even miss to find a solution (if the increments are too big). In fact, we show that it is an actual drawback in Chapter 5.

3.2.3 Analysis

Decentralized approaches for SCF appear as an interesting alternative to centralized methods for SCF. Moreover, these approaches alleviate some of the issues present in centralized approaches. First, participants in decentralized SCF share their preferences with local trusted parties rather than revealing their preferences to a central authority. Second, decentralized SCF methods are more likely to scale better for sufficiently large-scale scenarios since the computation can be performed in a distributed manner. Third, decentralized SCF methods are more

resilient to failure since they do not rely on a centralized entity. However, although economically efficient approaches exist, the computational requirements for the state-of-the-art decentralized SCF approaches can be higher than for the centralized approaches.

3.3 Conclusion

In this chapter we have reviewed the state of the art on SCF for both centralized and decentralized approaches. Moreover, for the decentralized case, we have reviewed both peer-to-peer and mediated architectures. Although, centralized optimal approaches seem appealing to solve the SCF problem, as discussed in Section 3.1, employing them might not be satisfactory for several reasons. First, centralized approaches rely on a central authority that possesses the private valuations of all participants. However, participants might be reluctant to share this information with any central authority. Second, given the hardness of the SCF problem, centralized optimal solvers might suffer from scalability issues. Third, the existence of a central authority introduces a single point of failure for the SCF process. In order to overcome some of these limitations, as discussed in Section 3.2, decentralized SCF appears as an attractive alternative. In peer-to-peer SCF, participant agents interact directly with each other without the need of any third party. However, the computational requirements of the state of the art in peer-to-peer SCF, LBP, grow exponentially with the size of the problem, thus hindering its scalability. On the other hand, in mediated SCF each of the goods in the SC is represented by a computational agent that mediates between the participant agents interested in buying or selling the good it is in charge of. Although being able to achieve high economic efficiency, the state of the art for mediated SCF, SAMP-SB-D, incurs in a high penalty in terms of communication.

In the following chapters we present our contributions to both peer-to-peer and mediated SCF. These contributions aim at improving on the state of the art in decentralized SCF and at answering the open research questions laid out in Chapter 1.

Chapter 4

RB-LBP: Peer-to-peer Supply Chain Formation

In previous chapters we have argued that there are several reasons to tackle the SCF in a decentralized manner rather than in a centralized optimal one. First, decentralized methods for Supply Chain Formation (SCF) better preserve participants' privacy since they only need to share their preferences with local trusted parties rather than communicating them to a central authority. Second, decentralized SCF offers better scalability for large scenarios due to the fact that each participant is responsible of a small part of the computation. Third, decentralized SCF can be more resilient to failure since the failure of a participant does not hinder the whole SCF process as would happen if the central authority in a centralized approach was to fail. Moreover, peer-to-peer (P2P) methods for SCF present an interesting approach to decentralized SCF since they do not require the intervention of any third party to determine the participants in the Supply Chain (SC). That is, in P2P SCF participants exchange information only with the suppliers of the goods they require and the consumers of the goods they produce. However, as discussed in Section 3.2.1, the state of the art in P2P SCF (LBP) suffers from scalability issues. Therefore, it remains an open research question whether P2P SCF can achieve economically and computationally efficient solutions (Questions 1 and 2 in our introductory chapter).

In this chapter we propose the Reduced Binarized Loopy Belief Propagation algorithm (RB-LBP henceforth), a novel method for P2P SCF with the aim of coping with the scalability issues present in the state of the art for P2P SCF. RB-LBP is based on the max-sum algorithm and simplifies the calculation of max-sum messages through careful analysis of its local terms and the application of the techniques described in Section 2.2.4. Thus, RB-LBP reduces the computation required by the state of the the art in P2P SCF, LBP, to assess SCs from exponential to quadratic, and the memory and communication requirements from exponential to lineal. Thus, RB-LBP is able to save several orders of magnitude in terms of memory, communication and computation with respect to the state

of the art in P2P SCF while the value of the resulting SCs is up to two times higher.

The rest of this chapter is organized as follows. First, in Section 4.1, we propose a novel additive decomposition of the SCF problem and its mapping into a binary local term graph for P2P SCF. In Section 4.2 we describe the operation of the RB-LBP algorithm and provide the means to assess the messages exchanged between participant agents in an efficient manner. In Section 4.3, we provide a complexity analysis of RB-LBP. Furthermore, we show that RB-LBP reduces the state-of-the-art resource requirements from exponential to linear for memory and bandwidth and from exponential to quadratic for the number of operations. In Section 4.4, we conduct an experimental evaluation of our method and compare it against the state of the art in P2P SCF. Our results show that our method is 20 times faster than the state of the art. Similarly, resource requirements are reduced between two and five orders of magnitude with respect to the state of the art in large problems. Furthermore, RB-LBP finds solutions of higher value than those found by the sate of the art. Finally, in Section 4.5, we identify the weaknesses of RB-LBP, diagnose their causes and propose measures to mitigate them.

4.1 A local term graph encoding for Peer-to-peer Supply Chain Formation

In this section we propose a novel additive decomposition of the SCF problem that can be mapped into a local term graph and approximated using the max-sum algorithm. The local terms present in our additive decomposition of the SCF problem are carefully chosen to allow us to apply the techniques described in Section 2.2.4. Applying these techniques to our model will allow us to reduce the computational complexity of our algorithm as we show on Section 4.2.2. Moreover, our variables are binary which is known to facilitate the simplification process [Tarlow et al., 2010]. Furthermore, in our model, each buy and sell decision is decoupled (i.e. encoded in a different variable) from the rest of buy and sell decisions. By decoupling these decisions we are able to reduce the number of combinations to take into account as we show in Section 4.3.

Next, in Section 4.1.1 we describe the variables used in RB-LBP to encode the SCF problem. Moreover, in Section 4.1.2 we introduce the local terms that restrict the values these variables can take in order to assess feasible solutions.

4.1.1 Encoding participants' decisions and their costs

In what follows we describe the encoding of participants' decissions in our additive decomposition by means of an example. Take for instance the lime juice example (Example 5) in Chapter 3. Likewise every participant, Dave has to decide whether to take part in the SC or not. This decision is encoded by means of an *activation variable* x_D that takes value one whenever Dave is to be active and zero otherwise. Moreover, we need to encode Dave's activation cost. For

that end, we include an *activation term* $f_D(\mathbf{x_D})$ that takes value -\$10 (Dave's activation cost) whenever Dave is active ($\mathbf{x_D} = 1$) and zero otherwise.

Furthermore, as seen in Figure 3.1, Dave buys Lime good and sells Juice good. Dave has three choices to buy Lime: he can acquire it either from Alice, Bob, or Carol. Consequently, we employ *option variables* b_{ALD}, b_{BLD}, and b_{CLD} to encode whether to buy from Alice, Bob, or Carol. For instance, if Dave was to buy the Lime from Alice the corresponding option variable would take on value one (i.e. $\mathbf{b_{ALD}} = 1$). Similarly, Dave has three choices to sell Juice: he can sell it to either Eve, Frank or Carol. Therefore, we encode the choice of selling Juice to each of the three possible buyers in variables s_{DJE}, s_{DJF}, and s_{DJG}.

In general, for each participant p taking part in the SC we create two kind of variables. On the one hand, we create an **activation variable** x_p that encodes whether participant p is active ($\mathbf{x_p} = 1$) or inactive ($\mathbf{x_p} = 0$), namely part of the SC configuration or not. Moreover, in order to introduce participants' activation cost, we make use of activation terms. An **activation term** takes as parameter an activation variable and takes as a value the activation cost of the participant when the activation variable takes value one and zero otherwise. Formally, the equation for an activation term f_p for participant p can be expressed as:

$$f_p(\mathbf{x_p}) = \begin{cases} C_p, & \text{if } \mathbf{x_p} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

Furthermore, for each possible buyer p' of each of her input goods g, we create an **option variable** $s_{pgp'}$ that encodes whether p is selling good g to participant p' ($\mathbf{s_{pgp'}} = 1$) or not ($\mathbf{s_{pgp'}} = 0$). Similarly, for each possible seller p' of each p's input goods g, we create an option variable $b_{p'gp}$ that encodes whether p is buying good g from participant p' ($\mathbf{b_{p'gp}} = 1$) or not ($\mathbf{b_{p'gp}} = 0$).

4.1.2 Constraining participants' decisions

It turns out clear that only some combinations of states are acceptable for the variables described above. Thus, if Dave is inactive ($\mathbf{x_D} = 0$), he should not buy Lime and so b_{ALD}, b_{BLD} and b_{CLD} should all be 0. Furthermore, whenever Dave is active, he should buy Lime from only one of his providers, that is, one and only one out of b_{ALD}, b_{BLD}, and b_{CLD} should be 1. We encode whether a set of values is acceptable by means of a local term f_S that takes as input variables x_D, b_{ALD}, b_{BLD}, and b_{CLD}. This local term takes value zero for valid combinations and negative infinity otherwise. Since this local term guarantees that only one of the providers is selected, we call it a *selection term*. Likewise, there will be a selection term f_S with input variables x_D, s_{DJE}, s_{DJF}, and s_{DJG} to ensure that Dave can select one and only one buyer when active and no buyer otherwise. Therefore, selection terms $f_S(\mathbf{x_D}, \mathbf{b_{ALD}}, \mathbf{b_{BLD}}, \mathbf{b_{CLD}})$ and $f_S(\mathbf{x_D}, \mathbf{s_{DJE}}, \mathbf{s_{DJF}}, \mathbf{s_{DJG}})$ guarantee Dave's internal coherence as a decision maker.

Furthermore, we need to guarantee that the decisions made for different participants remain coherent with each other. For instance, in order for Dave to

effectively buy Lime from Alice (encoded in variable b_{ALD}), he needs that Alice decides to sell Lime to him (encoded in variable s_{ALD}). Therefore, we add a local term $f_E(\mathbf{s_{ALD}}, \mathbf{b_{ALD}})$ that takes value zero when both variables take the same value and minus infinity otherwise, thus enforcing coherence among participants. Since this local term enforces both variables to take the same value, we call it *equality term*.

In general, in order to guarantee that only one of the providers of a given good is selected, we make use of selection terms. Given a participant p offering good g, a **selection term** links the activation variable from the participant (namely x_p) with the different choices for that good (namely b_{*gp}), and enforces that one and only one option variable takes on value one if the activation variable is active and that all option variables take on value zero otherwise. Note that the role of the activation variable on a selection term is different from that of the option variables. Formally, the equation for a selection term f_S joining the activation variable x_p and option variables $o_1, \ldots, o_n$ can be expressed as:

$$f_S(\mathbf{x_p}, \mathbf{o_1}, \ldots, \mathbf{o_n}) = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} \mathbf{o_i} = \mathbf{x_p} \\ -\infty, & \text{otherwise} \end{cases} \tag{4.2}$$

Furthermore, in order to guarantee coherent decisions between participants, we make use of equality terms. An **equality term** links buy and sell variables regarding the same transaction (e.g. b_{ALD} and s_{ALD}) and enforces that both variables take the same value. Formally, the equation for an equality term f_E joining variables b and s can be expressed as:

$$f_E(\mathbf{b}, \mathbf{s}) = \begin{cases} 0, & \text{if } \mathbf{b} = \mathbf{s} \\ -\infty, & \text{otherwise.} \end{cases} \tag{4.3}$$

Notice that, contrarily to what happened with LBP, there is no need to keep a table for these local terms (see Table 3.1). It is possible to simply calculate the output of selection and equality terms using Equations 4.2 and 4.3. This allows us to save large amounts of memory in local terms with large domains.

At this point we are ready to provide the additive decomposition that will be used by RB-LBP to tackle the SCF problem following a P2P architecture. Our objective function can be expressed as:

$$\mathcal{R}_{\text{RB-LBP}}(\mathbf{X}) = \sum_{x_p \in X_p} f_p(\mathbf{x}_p) + \sum_{f_S \in F_S} f_S(\mathbf{X}_{f_S}) + \sum_{f_E \in F_E} f_E(\mathbf{X}_{f_E}), \tag{4.4}$$

where X_p is the set of participant variables, F_S is the set of selection terms, and F_E is the set of equality terms. With this additive decomposition in hand, it is possible to map the SCF into a binary local term graph. Recall from Section 2.2.1 that building a local term graph from an additive decomposable function amounts to: (i) creating a simple vertex for each variable (that summarizes all the simple terms with that variable as their scope); (ii) creating a composite vertex for each composite term; and (iii) connecting with an edge each simple and composite vertex that share a variable. Thus, applying

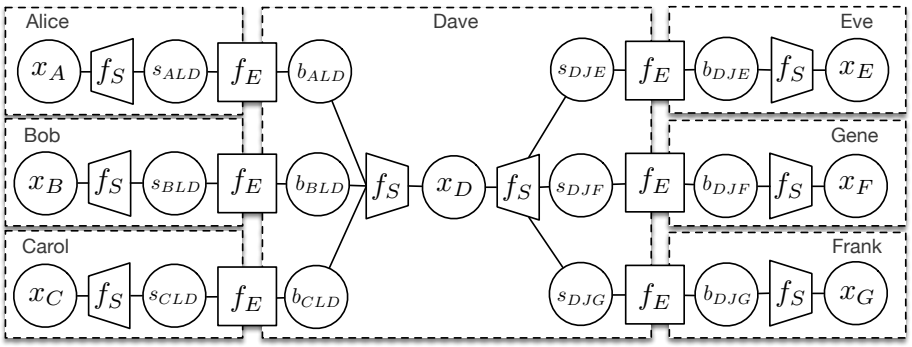


Figure 4.1: Supply chain as a binary local term graph.

this procedure to the SCF problem in Example 5 we obtain the local term graph in Figure 4.1. Notice that the local terms inside each of the dashed boxes correspond to the local terms that encode the decisions of each of the participants in the SC.

In this section he have proposed a new encoding for the SCF formation problem that employs only binary variables. Moreover, it is possible to map our encoding into a local term graph over which the max-sum algorithm can operate since we encode the problem by means of a function that can be decomposed additively. In the next section we describe RB-LBP, a P2P SCF algorithm that operates on the local term graph introduced above.

4.2 The RB-LBP algorithm

Reduced Binarized Loopy Belief Propagation (RB-LBP) is a message-passing algorithm that is based on the max-sum algorithm applied to the local term graph described in the previous section. Recall from Section 2.2 that max-sum exchanges messages between the vertices in a local term graph. Therefore, in order to be able to provide a P2P method for SCF based on max-sum we need to assign the vertices in the local term graph to some computational agents representing the participants in the SC. In RB-LBP, there is a participant agent for each participant in the SC. Each participant agent is in charge of the vertices representing the local terms that encode its participant's decisions. For instance, back in Figure 4.1, Dave's agent will be in charge of Dave's activation (x_D), option (b_{ALD}, b_{BLD}, b_{CLD}, s_{DJE}, s_{DJF}, and s_{DJG}), selection, and equality vertices that connect Dave's option vertices with his potential partners' option vertices. Thus, the participant agents representing each pair of potential partners will be connected through the equality vertices that join their option variables. For instance, Alice's participant agent is connected with Dave's through the equality

vertex that joins option vertices s_{ALD} and b_{ALD}.

Next, Section 4.2.1 contains RB-LBP's algorithmic details. In Section 4.2.2 we describe the messages exchanged between participant agents.

4.2.1 Algorithm description

Participant agents in RB-LBP follow a protocol that has two phases. During the first one, each participant agent uses the max-sum algorithm to send messages to its neighbors in the local term graph. As a result of this first phase, each participant agent finds out its preference for being active in the SC. During the second phase, participant agents use the information obtained from the first phase to decide which participants are going to be active in the resulting SC. In what follows we describe both phases.

Algorithm 3 Algorithm run by a participant p in RB-LBP.

// Assess the participant's preferred state.
1: Initialize all messages to zeros.
2: **while not** convergence **and not** reached maximum number of iterations **do**
3: Internal propagation.
4: Send max-sum message to each potential partner p'.
5: Receive max-sum message from each potential partner p'.
6: **end while**
7: Assess the preferred state $\mathbf{x}_\mathbf{p}^*$ according to Equation 2.5.
// Assess the SC configuration.
8: Ensure internal consistency.
9: **while** agent state **is not** consistent **do**
10: Notify potential partners of its willingness to collaborate.
11: Receive potential partners' willingness to collaborate.
12: Ensure collaboration consistency.
13: **end while**

Assessing participant's preferred states

In order to assess their preferred states, participant agents exchange messages following the max-sum algorithm. Initially, all participant agents initialize their messages to zero. Subsequently, participant agents exchange messages according to the max-sum algorithm. Recall that participant agents exchange messages through the equality vertices. Therefore, each participant agent needs only to communicate with its potential partners. That is, the message exchange between its vertices can be done internally without consuming any bandwidth resources. This message exchange continues until convergence or reaching a maximum number of steps. After that, each participant agent assesses its preferred state by means of the independent variable marginals method described in Section 2.2.3.

The procedure followed by a participant agent during the first phase of RB-LBP is described in lines 1-7 of Algorithm 3. However, it is possible that the

solution obtained after this first phase does not satisfy some of the constraints (i.e. some of the local terms evaluates to negative infinity). Therefore, RB-LBP includes a second phase to ensure the feasibility of the solution.

Assessing the Supply Chain configuration

In order to ensure that the decissions made by different participants are coherent (i.e. if participant p is willing to collaborate with p', p' must be willing to collaborate with p), RB-LBP includes a second phase in which the preferred states obtained in the first phase are revised (and possibly mended). The mending process has two main steps: during the first one, each participant agent ensures that all of his internal constraints are satisfied; during the second one, participant agents ensure that suppliers and consumers agree on their decision to collaborate. That is, that equality constraints are satisfied.

At the first step, in order to guarantee *internal consistency*, each participant agent checks each of its selection terms. When a selection term breaks (i.e. evaluates to negative infinity), there are three possible cases:

- The activation variable is set to zero and some option variables set to one.
- The activation variable is set to one and all option variables are set to zero.
- The activation variable is set to one and there are more than one option variables set to one.

In the first two cases, the participant agent will set all of its variables to zero. In the third case, the participant agent will randomly select one of the option variables and set it to one while setting the remaining ones to zero. While this first step guarantees internal consistency, it could be the case that two participant agents decide to take states that are not consistent with each other. For instance, it could be the case that Dave's agent decides to buy Lime from Alice but Alice does not sell it to Dave.

At the second step, in order to guarantee *collaboration consistency* participant agents need to organize as a spanning tree to be able to act in sequence. This can be done distributedly and efficiently [Attiya and Welch, 2004]. After building the tree, the procedure continues as follows. First, simultaneously, each agent sends to each of its neighbours its decision to collaborate with it or not. Second, sequentially, each active agent will determine (based on the information received at the previous step) whether all of its selected partners want to collaborate with it. If that is not the case, it will set all of its variables to 0 and send this information to its neighbours. This second step is repeated until no agent changes its states.

Notice that, since the participant agents can only go from being active to being inactive, the process of assessing the solution is guaranteed to finish in a number of steps no greater than the graph diameter. The procedure followed by a participant agent during the second phase of RB-LBP is described in lines 8-13 of Algorithm 3.

Recall that the assessment of max-sum messages for a composite term with k binary variables takes time $\mathcal{O}(2^k)$ (Section 2.2.4). Therefore, asssessing RB-LBP messages can turn to be computanionally ineficient. To reduce the complexity of the first phase, in the next section we introduce a simplification of the computation of RB-LBP's max-sum messages that leads to an efficient message-passing protocol for P2P SCF.

4.2.2 Efficient message updates

In RB-LBP, each vertex exchanges single-valued messages that represent its preferences for its shared variable to take value one over taking value zero. We use ν_v^w to denote the single-valued message from vertex v to vertex w. Moreover, we reduce the complexity of the computation of messages. This simplification stems from taking benefit of the fact that our local terms represent constraints and the alternatives that do not satisfy the constraint can directly be discarded. A complete description of this simplifications is given in Appendix A. After simplifications, messages in RB-LBP can be proved to be the following ones:

Message from activation vertex to selection vertex. The message sent from agent's p activation vertex x_p to one of its selection vertices f_S contains a single value $\nu_p^{f_S}$ that can be assessed by:

$$\nu_p^{f_S} = C_p + \sum_{f_S{}' \in \mathcal{N}(x_p) \setminus \{f_S\}} \nu_{f_S{}'}^p, \tag{4.5}$$

where $\mathcal{N}(x_p) \setminus \{f_S\}$ is the set of selection vertices neighboring activation vertex x_P excluding f_S, and C_p corresponds to the activation cost of the participant to whom corresponds activation variable x_p.

Message from option vertex to selection vertex. The message sent from an option vertex o to a selection vertex f_S contains a single value $\nu_o^{f_S}$ that can be assessed by:

$$\nu_o^{f_S} = \nu_{f_E}, \tag{4.6}$$

where ν_{f_E} is the equality vertex that option vertex o is connected to.

Message from option vertex to equality vertex. The message sent from an option vertex o to an equality vertex f_E contains a single value $\nu_o^{f_E}$ that can be assessed by:

$$\nu_o^{f_E} = \nu_{f_S}, \tag{4.7}$$

where ν_{f_S} is the selection vertex that option vertex o is connected to.

Message from selection vertex to activation vertex. The message sent from a selection vertex f_S to its activation vertex x_p contains a single value $\nu^p_{f_S}$ that corresponds to the largest message received by f_S from any of its neighboring option vertices. Formally,

$$\nu^p_{f_S} = \max_{o \in \mathcal{N}(f_S) \setminus \{x_p\}} \nu^{f_S}_o, \tag{4.8}$$

where $\mathcal{N}(f_S) \setminus \{x_p\}$ is the set of option vertices neighboring composite vertex f_S.

Message from selection vertex to option vertex. The message sent from a selection vertex f_S to one of its option vertices o contains a single value $\nu^o_{f_S}$ that can be asssessed by:

$$\nu^o_{f_S} = \min(\nu^{f_S}_p, -\max_{o' \in \mathcal{N}(f_S) \setminus \{x_p, o\}} \nu^{f_S}_{o'}), \tag{4.9}$$

where x_p is the activation vertex connected to f_S, and $\mathcal{N}(f_S) \setminus \{x_p, o\}$ is the set of option vertices neighboring composite vertex f_S excluding o.

Message from equality vertex to option vertex. The message sent from an equality vertex f_E to one of its option vertices o contains a single value $\nu^o_{f_E}$ that corresponds to the last message received from its other option variable o'. Formally,

$$\nu^o_{f_E} = \nu^{f_E}_{o'}. \tag{4.10}$$

Note that these simplified expressions to asssess the messages greatly reduce the computational requirements needed by a participant agent. In the case of the selection vertices we are reducing standard max-sum's computation time from $\mathcal{O}(2^k)$ to $\mathcal{O}(k)$, where k is the number of variables in the domain of the selection term. Importantly, as we formally prove in Appendix A, we are not approximating the max-sum messsages but providing a praticularly 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, the first phase of RB-LBP is an instance of max-sum, and thus, it inherits all max-sum's convergence and quality properties.

In Section 2.2.3 we argued that the solution assessed when each agent independently determines the values of its variables can sometimes break constraints. In many cases this happens because there is a tie in the agent's preferences. Consider the example in Figure 3.1. There Dave has three possible providers, Alice and Carol, selling Limes at the same price. The preferences of Dave's participant agent for each of the two producers will be exactly the same, and both variables b_{ALD} and b_{CLD} can be set to 1, breaking the selection constraint.

Here we propose a simple strategy to reduce the number of ties in selection constraints. The main idea is to establish an ordering among providers/consumers linked by a selection constraint in case that the expected

benefit of collaborating with them is the same. For example, in the case above, before agents start exchanging messages, Dave's participant decides that it prefers to buy from Alice rather than from Carol, provided that both make the same offer. Each agent can accomplish this by assigning an economically negligible random quantity to the value of collaborating with each of its potential partners. That is, each agent, for each of its option variables o_i, selects a small value ε_{o_i} and the messages in Equations 4.8 and 4.9 are modified as follows:

$$\nu^p_{f_S} = \max_{o \in \mathcal{N}(f_S) \setminus \{x_p\}} (\nu^{f_S}_o + \varepsilon_o), \tag{4.11}$$

$$\nu^o_{f_S} = \varepsilon_o + \min(\nu^{f_S}_p, - \max_{o' \in \mathcal{N}(f_S) \setminus \{x_p, o\}} \nu^{f_S}_{o'}), \tag{4.12}$$

4.3 Complexity analysis

Along the analysis conducted in Section 3.2.1 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. Moreover, we compare these results with LBP's complexity and argue that RB-LBP provides a significant reduction on the resources required to solve the SCF problem following a P2P architecture. We assume that the maximum number of goods that each participant is connected to is G, and that the maximum number of participants connected to a single good is P. That is, each participant will need to negotiate for at most G goods that can be traded with at most P other participants each. Therefore, a participant will be linked with $G \cdot P$ neighbors in the worst case.

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

Memory. Each RB-LBP participant agent needs to store a real number per variable in order to maintain preferences over a variable's states. Since each participant agent interacts with at most $G \cdot P$ other participant agents, the amount of memory to store its preferences is $\mathcal{O}(G \cdot P)$. Regarding local terms, each RB-LBP participant agent only needs to store its activation term (it can avoid storing equality and selection terms thanks to the equations obtained in Section 4.2.2). Hence, the memory that each RB-LBP participant agent needs is in $\mathcal{O}(G \cdot P)$.

Communication. Each RB-LBP participant can be interested in buying or selling at most G goods. Moreover, for each of these goods there would be at most P other participants interested in trading that good with him. Therefore,

Measure	LBP	RB-LBP
Operations (per iteration)	$\mathcal{O}(G \cdot P^{2G+1})$	$\mathcal{O}(G \cdot P^2)$
Memory	$\mathcal{O}(G \cdot P^{2G+1})$	$\mathcal{O}(G \cdot P)$
Bandwidth	$\mathcal{O}(G \cdot P^{G+1})$	$\mathcal{O}(G \cdot P)$

Table 4.1: LBP and RB-LBP worst case resource requirements.

since participant agents exchange a single-valued message for each of these potential trades, a participant agent worst case communication requirements are $\mathcal{O}(G \cdot P)$.

Table 4.1 compares the resouces required by the LBP and RB-LBP algorithms. RB-LBP reduces the communication and bandwidth requirements from exponential to linear with respect to LBP. This reduction is achieved by decoupling the participant's buy and sell decisions. Moreover, RB-LBP reduces the computation requirements from exponential to quadratic with respect to LBP. This reduction is achieved thanks to the simplified expressions to assess the max-sum messages exchanged during the execution of the algorithm. In the next section we empirically evaluate LBP and RB-LBP.

4.4 Experimental evaluation

In this section, we describe a series of experiments designed and performed in order to quantify RB-LBP's savings in terms of computation, memory and communication with respect to the state of the art and to assess the performance of RB-LBP in terms of the quality of the solution. First, we describe RB-LBP and LBP implementation details and which tools were used in the process. Second, we detail the experimental design. Last, we describe the metrics that were used to measure the performance of both methods and provide a thorough analysis of the collected data.

4.4.1 Implementation of peer-to-peer Supply Chain Formation algorithms

Both LBP and RB-LBP were implemented as an extension of libDAI [Mooij, 2010]. libDAI is an open source C++ library that implements various probabilistic inference methods. libDAI was a perfect starting point for our implementation since both LBP and RB-LBP are based on the max-sum algorithm, which is already implemented in libDAI.

In order to implement LBP we used libDAI's max-sum implementation with two extensions: (i) LBP's convergence criterion that slightly differs from the one in use in libDAI and (ii) LBP's post-convergence process for eliminating incompatibilities described in Section 4.2.1.

The implementation of RB-LBP was more challenging since it is a specialization of the standard max-sum algorithm. Our main modifications to libDAI

were:

1. **Local terms as functions**. One of the strengths of RB-LBP is that it does not need tables to store its local terms since they can be calculated from their inputs, saving memory in the process. Therefore, our first change in the implementation was to extend the local terms in libDAI (which are stored as tables) to allow such behaviour.

2. **Reduced messages**. The introduction of binary variables allows RB-LBP to greatly simplify the calculation of messages and the amount of information exchanged between local terms. Therefore, we had to modify the logic related to message calculation and message passing.

3. **Local terms with preferences**. The inclusion of preferences as tie-breaking mechanism in RB-LBP resulted in a slight modification of the implementation of the local terms in libDAI.

Finally, in our implementation of RB-LBP, in order to avoid ties, we generated values of ε_{o_i} (see Section 4.2.2) by sampling a uniform distribution in the range $[-0.00005, 0.00005]$. Next, we describe the datasets used to conduct the empirical evaluation. Our implementation of LBP and RB-LBP can be freely downloaded from [Penya-Alba et al., 2012e].

4.4.2 Experimental design

In this section we describe the design of the experiments conducted to evaluate the performance of RB-LBP. In [Winsper and Chli, 2010], Winsper and Chli evaluate their method in the SCs described by Walsh and Wellman in [Walsh and Wellman, 2003]. These SCs are relatively small (33 participants at most), which makes them poor candidates to evaluate an algorithm's scalability. However, these SCs have been studied in detail by Winsper et al. in [Winsper and Chli, 2010] and represent a good reference point for a more detailed comparison of LBP and RB-LBP.

For these small SCs we follow the same initialization procedure as described in [Walsh and Wellman, 2003] and followed in [Winsper and Chli, 2010]. Selling prices for producers are drawn randomly from a uniform distribution U(0,1). Moreover, buying prices for final consumers are fixed to the value that appears below each consumer in Appendix C. These prices were calculated in [Walsh and Wellman, 2003] to ensure the existence of a solution in which a profitable SC configuration exists in 90% of the instances.

Our main concern is to evaluate how RB-LBP scales compared to LBP and to quantify the savings in terms of resources required to find a solution. For that purpose, we need large-scale SCs. In order to generate larger networks we resorted to the test-suite for Mixed Multi-Unit Combinatorial Auctions (MMUCATS) described in [Vinyals et al., 2008]. MMUCATS is specifically designed to mimic real-world SCF problems. We generate SCs with 50 goods and a number of participants ranging from 40 to 500. Since the number of goods is fixed

across scenarios, the degree of competition in the SCs grows with the number of participants. The problems used to evaluate RB-LBP can be freely downloaded from [Penya-Alba et al., 2012c].

For each scenario, we generate 100 problems. We solve problems using the implementations for RB-LBP and LBP described above. 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. For each problem we record the following values:

Maximum memory. Measured as the maximum amount of memory needed by any participant agent in the problem to store both its preferences and its local terms.

Maximum bandwidth. Measured by the maximum quantity transmitted and received by any participant agent in an iteration of the problem. Note that in LBP and RB-LBP participant agents has the same network usage for all the iterations of the algorithm.

Problem solving time. Time taken to solve the problem. This time includes the time necessary for the post-processing of the allocations (Section 4.2.1) both for LBP and RB-LBP.

Value benefit over LBP. Calculated by dividing the value of the SC assessed by RB-LBP by the value of the supply chain assessed by LBP. The values of the supply chains are assessed using Equation 3.1 and Equation 4.4 for LBP and RB-LBP respectively.

4.4.3 Empirical evaluation

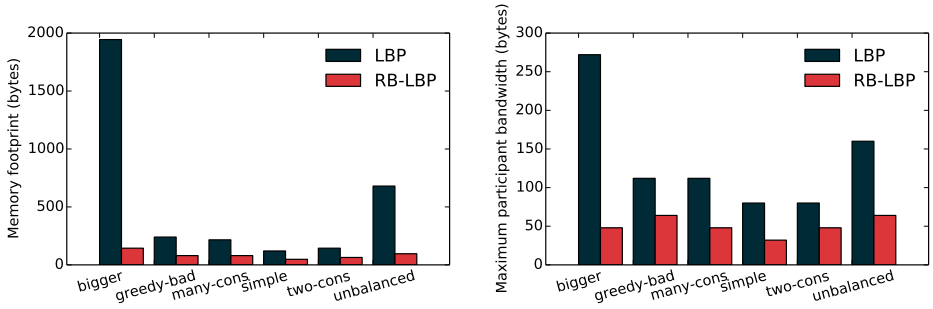
In this section we analyze the results obtained after evaluating separately SCs from [Walsh and Wellman, 2003] and larger networks with higher degrees of competition generated with MMUCATS. Hardness in the problems we generated using MMUCATS presented high variability. Therefore, the distribution of the results we obtained was long-tailed and non-symmetrical.

In order to improve readability of the plots and maintain consistency in how results are reported, we use median values instead of mean values [Wilcox and Keselman, 2003] both for Walsh and Wellman's SCs and large-scale network structures. Moreover, we provide plots in logarithmic scale for those measures in which the difference in performance between RB-LBP and LBP was large.

Small network structures

In this section we turn our attention to the SCs from [Walsh and Wellman, 2003]. Figure 4.2a shows that LBP requires from 2 up to 13 times more memory than RB-LBP depending on the SC. This difference is specially large for the SCs named

bigger and *unbalanced* that represent markets with high competition. Figure 4.2b shows that the maximum bandwidth consumed by a participant agent during an LBP iteration is up to 5 times larger than RB-LBP's. Again, the difference is more obvious for *bigger* and *unbalanced* SCs since network usage and memory requirements are tightly coupled in LBP.



(a) Maximum memory per participant.

(b) Maximum bandwidth per participant.

Figure 4.2: RB-LBP vs LBP in Walsh's SCs.

The values of the SC configurations obtained by our implementation of LBP match the results reported in [Winsper and Chli, 2010]. RB-LBP's average SC values are identical to LBP's. Due to the small size of the networks, both LBP and RB-LBP converge to a SC configuration in the order of the millisecond, being negligible the difference between both methods.

Large network structures

In this section we focus on the scalability of RB-LBP and LBP. In Sections 3.2.1 and 4.2.1 we mention that LBP and RB-LBP are not guaranteed to converge to a feasible solution and that, if convergence is not reached, both methods halt after a certain number of iterations. Contrarily to what happened in the previous section, problems with high degrees of competition did not always converge. Therefore, in our experiments, problems run either until convergence or for a maximum of 250 iterations. Moreover, due to computational constraints, problems in which the memory requirements for an agent exceed 100MB or the memory needed by the whole network exceeded 1GB were discarded. Note that, since RB-LBP agents have much lower memory requirements than LBP agents, we only encountered such cases for LBP.

Figure 4.3 shows the median values over 100 runs of the maximum memory required per agent for both RB-LBP and LBP. Observe that LBP's memory requirements grow exponentially with the number of participants while memory requirements for RB-LBP grow linearly. Moreover, for networks with higher de-

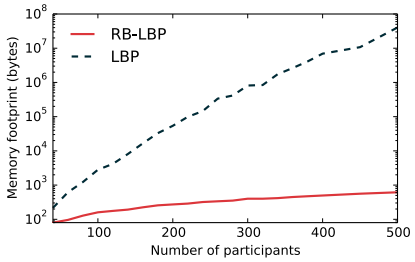


Figure 4.3: Maximum memory requirement.

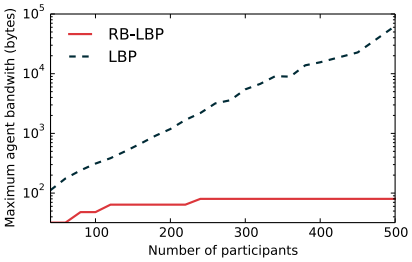


Figure 4.4: Maximum bandwidth per agent.

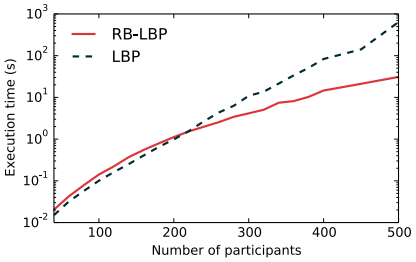


Figure 4.5: Median problem solving time.

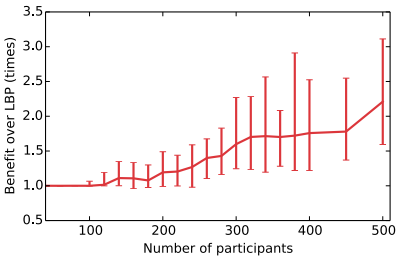


Figure 4.6: RB-LBP benefit over LBP.

grees of competition, the memory requirements for LBP are up to 5 orders of magnitude (10^5 times) greater than for RB-LBP.

Figure 4.4 shows the median value of the maximum bandwidth required per agent during one iteration for both RB-LBP and LBP. In this case, bandwidth usage also displays an exponential growth for LBP while it is lineal for RB-LBP. Bandwidth usage for LBP is up to 787 times greater than for RB-LBP.

Figure 4.5 shows the median value of the time required for RB-LBP and LBP to provide a solution for the problems. In this case, the exponential behaviour of LBP is confirmed as well as the polynomial behaviour of RB-LBP. Moreover, RB-LBP is up to 20 times faster than LBP for the problems tested.

Finally, it is worth noting that none of those benefits come at the expense of the quality of the solution. Moreover, Figure 4.6 shows that the value of the SC configurations obtained by RB-LBP is almost never smaller and eventually more than 2 times larger than those obtained by LBP.

These results place RB-LBP as the best performing method for P2P SCF. On the one hand, RB-LBP is able to assess SCs of higher value than the state-of-the-art for P2P SCF while running up to 20 times faster, requiring up to

787 less bandwidth, and reducing the memory requirements up to five orders of magnitude. However, the quality of the SCs assessed by LBP and RB-LBP gets further from the optimal as the complexity of the SC rises. Figure 4.7 shows the optimality of the SCs assessed by LBP and RB-LBP[1]. In the next Section we argue that the low performance in terms of solution quality of the P2P SCF methods in the state of the art (including RB-LBP) might be due to the model and implementation chosen.

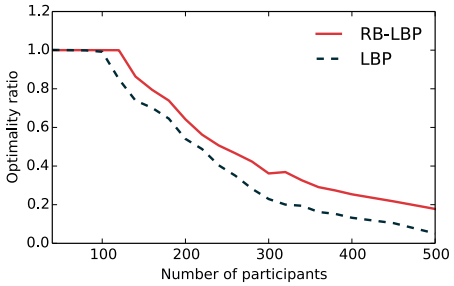


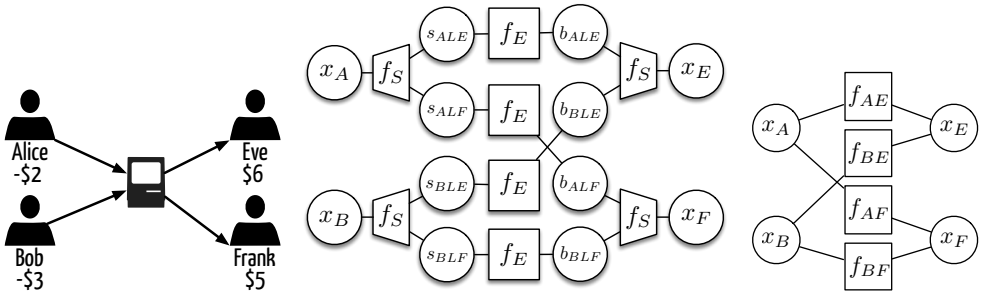
Figure 4.7: Optimality of the solutions assessed by RB-LBP.

4.5 Conclusion

In this chapter we have proposed RB-LBP, a P2P method for SCF. RB-LBP is based on a novel additive decomposition of the SCF, its mapping into a local term graph, and a computationally efficient application of the max-sum algorithm. Moreover, the local terms in RB-LBP's additive decomposition are carefully selected in order to be able to introduce optimizations in max-sum's message calculation. These simplifications reduce the state-of-the-art for P2P SCF resource requirements from exponential to linear for memory and bandwidth, and from exponential to quadratic as to the number of operations. Experimentally, these simplifications allow RB-LBP to find solutions to the SCF problem 20 times faster than LBP, the state of the art in P2P SCF. Similarly, bandwidth and memory requirements are reduced over 700 times and 5 orders of magnitude in RB-LBP over LBP in large scale problems. Furthermore, the reduction in resource requirements does not come at the expense of solution quality. That is, RB-LBP produces SCs whole values are up to two times of higher than those produced by LBP. However, the quality of the solutions found by both methods (LBP and RB-LBP) decreases as the number of participants in the SC grows. Next, we argue that this is a consequence of applying the max-sum algorithm to the P2P architecture.

Take into consideration the scenario in which two sellers and two buyers are interested in exchanging a good. As Figure 4.8 shows, both methods for P2P

[1]The optimal value was computed using a linear program.



(a) Cycle-free SCF. (b) RB-LBP's cyclic local term graph. (c) LBP's cyclic local term graph.

Figure 4.8: P2P cyclic local term graphs from a cycle-free SCF problem.

SCF generate local term graphs that contain cycles even though the original SCF problem contained no cycles. In general, both RB-LBP and LBP mappings into a local term graph introduce cycles that are not part of the original SCF problem whenever there is a good with more than one buyer and more than one seller. Recall form Section 2.2.2 that max-sum is guaranteed to converge to the optimal solution when the underlying local term graph is a tree. However, in cyclic local term graphs max-sum optimality is not guaranteed and it may even fail, to converge thus providing potentially unfeasible solutions [Weiss and Freeman, 2001]. Hence the number of cycles in the underlying local term graph is a determining factor for the quality of the solutions found by max-sum based algorithms. Therefore, RB-LBP and LBP solution quality will be affected by max-sum's suboptimality in cyclic local term graphs. In the next chapter, we introduce a new algorithm for decentralized SCF that addresses this problem by introducing mediators for goods.

Chapter 5

CHAINME: Mediated Supply Chain Formation

In previous chapters we have argued that decentralized approaches can be better suited to solve the Supply Chain Formation (SCF) problem for several reasons. First, in decentralized SCF, participants share their preferences with trusted parties in a local environment rather than trusting this information to a global central authority. Second, by distributing the computation among all the participants in the SC they are more likely to better scale to large-scale scenarios. Third, decentralized SCF methods are more resilient to failure since they do not rely on a centralized entity. The state-of-the-art approaches for decentralized SCF can be classified into two architectures: peer-to-peer (P2P), and mediated.

In P2P SCF each participant is represented by a participant agent that communicates directly with the participant agents representing the sellers of the goods it is interested in buying and the buyers of the goods its participant is producing. In Chapter 4, we have shown that the state of the art algorithm for P2P SCF, LBP [Winsper and Chli, 2013], suffers from high computational requirements issues. Moreover, in the same chapter we have proposed RB-LBP a method for P2P SCF that lowers the computational resources required to find solutions to the SCF problem with respect to LBP while obtaining better-valued solutions. However, the quality of the solutions found by both of these methods for P2P SCF is further from the optimal as the complexity of the SC rises.

Decentralized methods for SCF with mediators for goods benefit from introducing the concept of local markets for each of the goods in the SC. Each local market is represented by a mediator agent that interacts with the buyers and sellers of the good it is in charge of. Thus, the mediator is in charge of relaying communication and task allocation. Therefore, markets allow participants to reach new potential partners [Kalin, 2000] with no extra cost in terms of communication or computation. Although economically efficient approaches exist, the state of the art method for mediated SCF, SAMP-SB-D [Walsh and Wellman, 2003], suffers from high communication requirements as

discussed in [Winsper and Chli, 2013].

In this chapter we propose the CHaining Agents IN Mediated Environments (CHAINME) algorithm, a novel method for mediated SCF. With CHAINME we aim at providing a method for mediated SCF that provides economically efficient solutions while keeping a low profile in terms of the computational requirements required to find the SC configurations. CHAINME is based on the max-sum algorithm. However, CHAINME's computation is significantly reduced with respect to standard max-sum's through careful analysis of the local terms employed by CHAINME and the application of the message simplification techniques described in Section 2.2.4.

The rest of this chapter is organized as follows. First, in Section 5.1, we propose a novel additive decomposition of the SCF problem into a local term graph for mediated SCF. In Section 5.2, we describe the operation of the CHAINME algorithm and provide computationally efficient expressions to assess the messages exchanged between CHAINME agents. In Section 5.3, we provide a complexity analysis of CHAINME. In Section 5.4, we conduct an experimental evaluation of our method and compare it against the state-of-the-art for decentralized SCF (both P2P and mediated). The results show that CHAINME is able to produce economically efficient solutions while reducing the computational requirements from one to four orders of magnitude with respect to the state-of-the-art methods. Finally, in Section 5.5, we conclude.

5.1 A local term graph encoding for mediated Supply Chain Formation

In this section we provide a novel additive decompositon of the SCF that can be mapped into a local term graph and approximated using the max-sum algorithm following a mediated architecture. Our model will contain only binary variables and the local terms are chosen in order to be able to apply the techniques described in Section 2.2.4 in order to simplify message calculation. Next, in Section 5.1.1 we describe the encoding of the participants' decisions in binary varibales. Then, in Section 5.1.2, we describe the constraints introduced in order to ensure feasible solutions.

5.1.1 Encoding participants' decisions and their costs

In what follows we describe the encoding of participants' decisions in CHAINME's additive decomposition of the SCF problem. Take the lime juice industry in Example 5. In CHAINME's encoding there is a binary variable for each of the participants in the SC. Each **participant variable** x_p encodes whether participant p is active ($\mathbf{x_p} = 1$) or not ($\mathbf{x_p} = 0$) in the SC. Furthermore, in order to encode participants' activation cost, we introduce a simple term (f_p) for each participant p with its scope limited to the activation variable representing that participant. The **activation term** $f_p(\mathbf{x_p})$ for participant p can be described by

$$f_p(\mathbf{x_p}) = \begin{cases} C_p & \text{, if } \mathbf{x_p} = 1 \\ 0 & \text{, if } \mathbf{x_p} = 0, \end{cases} \tag{5.1}$$

where C_p is the activation cost for participant p. That is, for the active state the term takes on value the participants activation cost and for the inactive state it takes on value zero.

Take the SC described in Example 5. In CHAINME's mapping, there is an activation variable and an activation term per participant. There is an activation variable x_A and an activation term $f_A(\mathbf{x_A})$ for Alice, an activation variable x_B and an activation term $f_B(\mathbf{x_B})$ for Bob, and so on for the rest of participants.

Notice that, in order to encode a particiant's decisions, CHAINME's encoding only needs one variable whereas RB-LBP needed as many variables as possible partners for the participant plus the activation variable. Therefore, CHAINME requires less memory than RB-LBP in order to encode participants' decisions, since all variables in both CHAINME and RB-LBP are binary.

So far, we have described how participants' decisions are encoded in CHAINME's additive decomposition of the SCF problem. In the next section, we describe how participants' decisions are constrained so that a SC configuration remains feasible.

5.1.2 Constraining participants' decisions

In Chapter 1, we argued that a solution to a SCF problem is feasible whenever all participants are able to buy their input goods and sell their output goods. Notice that this is equivalent to saying that a solution to a SCF problem is feasible whenever the number of active sellers for each good is equal to the number of active buyers. In CHAINME, we say that a good is at equilibrium whenever this condition is met. Thus, in a feasible solution, all the goods in the SC will be at equilibrium.

In order to constrain participants' decisions so that the solution remains feasible we introduce an *equilibrium term* for each of the goods in the SC. An **equilibrium term** for good g is a composite term noted as m_g whose domain contains the variables related to its sellers ($\mathcal{S}_g$) and buyers ($\mathcal{B}_g$). We use $S_g = \{x_p | p \in \mathcal{S}_g\}$ to denote the variables of the sellers of g, and $B_g = \{x_p | p \in \mathcal{B}_g\}$ to denote the variables of the buyers of g. Formally, an equilibrium term can be defined as:

$$m_g(\mathbf{S_g}, \mathbf{B_g}) = \begin{cases} 0 & \text{, if } \sum_{x_s \in S_g} \mathbf{x_s} = \sum_{x_b \in B_g} \mathbf{x_b} \\ -\infty & \text{, otherwise} \end{cases} \tag{5.2}$$

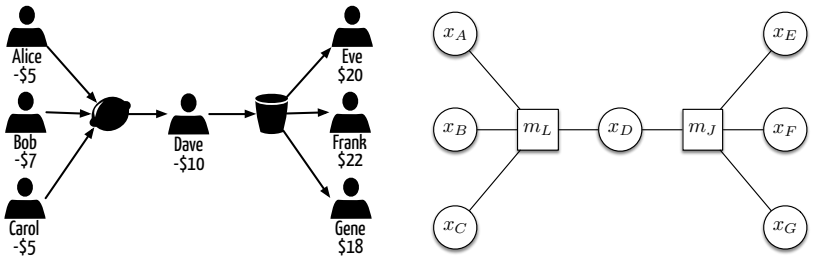
In the additive decomposition of the SC in Example 5, where there were two goods are at trade (Lime and Juice), we be two equilibrium terms $m_L(\mathbf{S_L}, \mathbf{B_L})$ and $m_J(\mathbf{S_J}, \mathbf{B_J})$.

At this point we are ready to provide the additive decomposition that will be used by CHAINME to tackle the SCF problem in a decentralized manner following a mediated architecture. Therefore, CHAINME's objective function can be

expressed as:

$$\mathcal{R}_{\text{CHAINME}}(\mathbf{X_p}) = \sum_{x_p \in X_p} f_p(\mathbf{x_p}) + \sum_{m_g \in M_g} m_g(\mathbf{S_g}, \mathbf{B_g}) \tag{5.3}$$

With this additive decomposition we can now map the SCF into a local term graph following the steps listed in Section 2.2.1. Thus, applying this procedure we can obtain the local term graph corresponding to the SCF problem described in Example 5 depicted in Figure 5.1. Notice that, in contrast with RB-LBP, in CHAINME's decomposition there is a single simple term for each of the participants and a single composite term for each of the goods at trade. Moreover, CHAINME's decomposition produces local term graphs that contain less cycles than those produced by LBP and RB-LBP which are known to negatively affect the solutions assessed by the max-sum algorithm [Weiss and Freeman, 2001]. Take for instance the SCF problem depicted in Figure 5.2a. CHAINME will produce the local term graph in Figure 5.2d that contains no cycles whereas LBP and RB-LBP produced cyclic graphs (Figures 5.2b and 5.2c). In general, CHAINME will produce local term graphs that contain cycles only if the undirected version of the original SC contained cycles. This is because we are basically substituting each participant by a simple term and each good by a composite term.



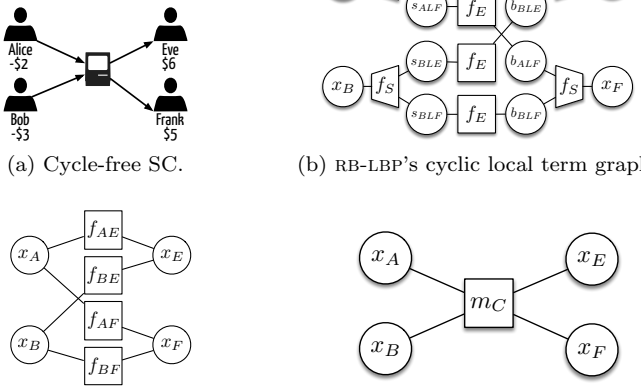
(a) Lime juice industry SC. (b) Local term graph.

Figure 5.1: CHAINME local term graph for the SC in Example 5.

In this section we have provided a way to map the SCF problem into a local term graph over which max-sum can operate. Now, in the next section we introduce the CHAINME algorithm that distributes the local term graph resulting from this mapping among different autonomous agents and subsequently approximates the SCF problem encoded by Equation 5.3 via message-passing over the graph.

5.2 The CHAINME algorithm

CHAINME is a max-sum based message-passing algorithm involving participants and mediators. Recall from Section 2.2 that max-sum exchanges messages between vertices of a local term graph. Thus, we need to distribute the vertices in



(a) Cycle-free SC.

(b) RB-LBP's cyclic local term graph.

(c) LBP's cyclic local term graph.

(d) CHAINME's cycle-free local term graph.

Figure 5.2: P2P cyclic local term graphs from a cycle-free SCF problem.

the local term graph provided in the previous section among computational agents in order to provide a decentralized, mediated algorithm for SCF. In CHAINME, there is a *participant agent* for each of the participants in the SC. Each participant agent is responsible for the simple vertex representing its participant's activation variable. Furthermore, for each of the goods at trade there is an agent that will act as mediator for that good. Each *mediator agent* is responsible for the equilibrium term representing its good. Figure 5.3 depicts the distribution of CHAINME's local term graph vertices into participant and mediator agents. Therefore, CHAINME is a message-passing protocol in which messages flow back and forth from participant agents to mediator agents. Thus, each participant agent in CHAINME communicates only with the mediator agents of the goods it wants to buy or sell. Likewise, each mediator agent only communicates with the participant agents willing to buy or sell the good it mediates.

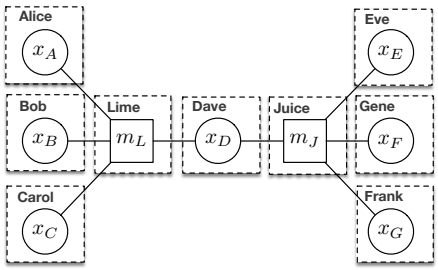


Figure 5.3: CHAINME's local term distribution among agents.

Section 5.2.1 contains CHAINME's algorithmic details. In Section 5.2.2 we describe the messages exchanged between participant and mediator agents.

5.2.1 Algorithm Description

CHAINME agents follow a established protocol that has two main phases. During the first phase, CHAINME agents run the max-sum algorithm over the CHAINME local term graph (introduced in Section 5.1). As a result of this first phase, each participant finds out how valuable it is for the SC as a whole when it is active (i.e. buying or selling goods) and hence, its preferred state. Based on that information, during the second phase, CHAINME agents decide which participants are going to take part in the SC. Next, we describe both phases in detail.

Algorithm 4 Algorithm run by a participant agent p in CHAINME.

`// Assess the participant's preferred state.`
For each mediator m_g neighbor of p, initialize message μ_p^g to zeros.
while not convergence **and not** reached maximum number of iterations **do**
    Send max-sum message to each mediator m_g neighbor of p.
    Receive max-sum message from each mediator m_g neighbor of p.
end while
Assess the preferred state $\mathbf{x_p^*}$ according to Equation 2.5.
`// Assess the SC configuration.`
Send $\mathbf{x_p^*}$ to each mediator m_g neighbor of p.
while not convergence **and** participant p is available **do**
    Receive from each of its good's mediators whether participant p should be active ($\mathbf{x_p^*} = 1$) or not ($\mathbf{x_p^*} = 0$).
    Set participant p to be activate if all of its good's mediators want it to be active.
    Send the new state $\mathbf{x_p^*}$ to each of its good's mediators.
end while

Assessing participants' preferred states

In order to assess the participants' preferred states, participant and mediator agents exchange messages following the max-sum algorithm. When the algorithm starts all participant and mediator agents initialize their messages to zero. Subsequently, each participant agent p sends a message to each of the mediator agents of the goods it is interested in. The message from a participant agent to a mediator agent encodes the participant's preference for each of its states (active or inactive). Participant agents assess the max-sum messages over their activation terms. Similarly, each mediator agent m_g sends a message to each of the participant agents interested in buying or selling the good it is mediating. The message from a mediator agent to a participant agent encodes the preference of the mediator agent for that participant to be included in or excluded from the

Algorithm 5 Algorithm run by a mediator m_g in CHAINME.

```
   // Assess the participants' preferred state.
```
1: For each participant p neighbor of m_g, initialize message μ_g^p to zeros.
2: **while not** convergence **and not** reached maximum number of iterations **do**
3: Send max-sum message to each participant p neighbor of m_g.
4: Receive max-sum message from each participant p neighbor of m_g.
5: **end while**
```
   // Assess the SC configuration.
```
6: Receive states from each neighboring participant.
7: **while not** convergence **do**
8: Define inactive participants as $I = \{p \in \mathcal{N}(g) | \mathbf{x}_\mathbf{p}^* = 0\}$.
9: Remove inactive participants from the neighbors ($\mathcal{N}(g) = \mathcal{N}(g) \setminus I$).
10: Determine preferred participants states $\mathbf{S}_g^*, \mathbf{B}_g^*$ according to Equation 5.4.
11: Send to each neighbor in $\mathcal{N}(g)$ whether it should be active or not.
12: Receive state updates from each neighboring active participants.
13: **end while**

solution. Mediator agents assess their max-sum messages over their equilibrium terms.

This process runs iteratively until convergence (or a maximum number of iterations is reached). After that, each participant agent assesses its preferred state by means of Equation 2.5. The procedure followed by a participant agent during the first phase is described in lines 1-6 of Algorithm 4, whereas the procedure followed by a mediator agent is described in lines 1-5 of Algorithm 5. However, it is possible that the preferred states chosen by the participant agents in this first phase lead to an unfeasible solution. Therefore, CHAINME includes a second phase to ensure the feasibility of the solution that we describe next.

Assessing the SC configuration

The decisions taken individually by the participant agents as a result of the previous phase may not correspond to a feasible SC configuration. Hence, during this phase, mediator and participant agents follow a protocol to ensure that the final SC configuration is feasible. This is achieved by iterating a two-step process. During the first step, participant agents communicate their preference for the active state to the mediator agents. During the second step, mediators communicate to participants whether they are eligible to be active.

At the first step each participant agent sends its preferred state to all of its neighboring mediator agents. Once a participant agent decides that it is inactive, it will no further change its state. Accordingly, each mediator agent, after receiving the preferred states from all its neighboring agents, removes the inactive agents from the set of potential buyers and sellers for its good. After that, each mediator recomputes the preferred states for its remaining neighbors following Equation 5.4 below, which can be derived from an extension of Equation 2.5 for composite local terms. Subsequently, each mediator agent sends to each of its

participant agents whether it is required to be active or inactive.

$$\mathbf{X}^*_{\mathbf{m_g}} = \arg\max_{\mathbf{S_g},\mathbf{B_g}} \left\{ m_g(\mathbf{S_g},\mathbf{B_g}) + \sum_{s\in\mathcal{S}_g} \mu^g_s(\mathbf{x_s}) + \sum_{b\in\mathcal{B}_g} \mu^g_b(\mathbf{x_b}) \right\}. \tag{5.4}$$

When a participant agent receives the state from each of its neighboring mediators, it decides to be remain active only if all of its neighboring mediator agents requested it to be active. Again, each participant agent sends its preferred state to all the mediator agents it is connected to.

Participant and mediator agents continue exchanging messages in an interleaved manner until no participant agent changes its preferred state. Participant agents who still prefer to be active after this process finishes will be the active participants in the SC and will compose the SC configuration. Notice that, similarly to what happened with RB-LBP, this phase is guaranteed to finish in a number of steps no greater than the graph diameter since participant agents can only go from being active to being inactive. The process to determine which participants are active is described in lines 6-12 of Algorithm 4 and lines 6-13 of Algorithm 5 for participant and mediator agents respectively.

Note that the assessment of messages from mediator to participant agents in line 3 of Algorithm 5 is particularly inefficient. Conventionally, the assessment of the max-sum message for a local term over k binary variables takes time $\mathcal{O}(2^k)$. Thus, computational complexity of mediator agents is exponential in the number of participant agents it is connected to. To reduce the complexity of this phase, in the next section, we introduce a simplification of the computation of CHAINME's max-sum messages that leads to a computationally efficient message-passing protocol between participant and mediator agents.

5.2.2 Efficient message updates

In CHAINME, agents exchange single-valued messages that encode the preference for the active state against the inactive one. We use ν^g_p to denote the message from participant agent p to the mediator agent of good g, and ν^p_g to denote the message from the mediator agent of good g to participant agent p. In this section we provide equations to assess this messages in a computationally efficient manner. This simplification stems from taking benefit of the fact that our equilibrium term represents a constraint and the alternatives that do not satisfy the constraint can directly be discarded. A complete description of this simplifications is given in Appendix B. After simplifications, messages in CHAINME can be proven to be the following ones:

Message from participant to mediator agent The message from participant agent p to the mediator agent of good g contains a single value ν^g_p. It is computed as the addition of all messages received by p, excluding the message

received from g, plus the participant's activation cost. Formally,

$$\nu_p^g = C_p + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{g'}^p, \tag{5.5}$$

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

Message from mediator to participant agent To assess the messages sent from the mediator agent of a good to a participant agent we build upon the bid-ask interval introduced in Section 2.1.2. Specifically, the mediator agent of good g computes the bid-ask interval (τ^+, τ^-), and the sets of active buyers ($\mathcal{B}_g^a$) and active sellers ($\mathcal{S}_g^a$) by executing Algorithm 1. In so doing, the messages received from the buyers ($\nu_{b_1}^g, \ldots, \nu_{b_{|\mathcal{B}_g|}}^g$) are used as buy offers, whereas the messages received from the sellers ($\nu_{s_1}^g, \ldots, \nu_{s_{|\mathcal{S}_g|}}^g$) are used as sell offers.

Now, the messages from a mediator to a participant agent can be proven to be the following ones.

- The message from the **mediator of good g to a seller** s contains a single value ν_g^s that is assessed as:

$$\nu_g^s = \begin{cases} \tau^+, & \text{if } s \in \mathcal{S}_g^a \\ \tau^-, & \text{otherwise .} \end{cases} \tag{5.6}$$

- The message from the **mediator of good g to a buyer** b contains a single value ν_g^b that is assessed as:

$$\nu_g^b = \begin{cases} -\tau^-, & \text{if } b \in \mathcal{B}_g^a \\ -\tau^+, & \text{otherwise.} \end{cases} \tag{5.7}$$

Hence, the messages from mediators to participants coincide (disregarding signs) with the bid (τ^-) and ask (τ^+) values in a periodic double auction that takes the participants' messages as bids. Furthermore, the message from a mediator to a participant agent does not depend on the particular message received from that participant agent, but only on whether the participant is active or inactive. Finally, the participants that are preferred as active by a mediator are those in her set of active buyers and sellers.

Note that the assessment of messages from mediators to participants is particularly efficient. Recall that the assessment of the max-sum message for a local term over k binary variables takes time $\mathcal{O}(2^k)$. Conversely, CHAINME's messages help us reduce this time to $\mathcal{O}(k \cdot \log k)$ (the time taken to order messages) for the equilibrium term. Importantly, as we formally prove in Appendix B, we are not approximating the max-sum messages, but providing 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, the first phase of CHAINME is an instance of max-sum, and thus, it inherits all max-sum convergence and quality properties. Additionally, as we argue next, this simplified versions of the messages give semantics to the standard max-sum messages.

Unraveling the semantics of CHAINME's messages

As we show in Section 5.2.2, after simplifications the computations of messages from mediators to participants at a given iteration requires to compute a bid-ask interval for a particular periodic double auction. This makes us think that there is a relationship between the messages exchanged in CHAINME and other concepts used in other disciplines such as auction theory and economics. In this section we elaborate on these connections by expressing the semantics of the messages in CHAINME in terms of the social value of participants in a double auction. Intuitively, messages from mediator to participant agents indicate their local social value, and participant agents use the social value to guide their local decision making.

We define the local social value for a good g of a participant p as the difference between the benefit for the remaining participants of having p at trade versus not having her at trade. Consider as example the mediation scenario shown in Example 2, to assess the local social value of Alice for that good (say g), first, we need to assess the largest possible benefit *for the other agents* when Alice is active (υ^*_A) and when she is not active (υ^*_{-A}). Therefore, the local social value for good g of Alice is $SV_g(A) = \upsilon^*_A - \upsilon^*_{-A}$. Recall that, in a periodic double auction, π^* denotes the largest possible surplus, and we refer to a π^*-configuration as a set of buyers and sellers that achieve surplus π^*. Since Alice is active in the π^*-configuration, υ^*_A is π^* minus the contribution of Alice, namely her offer (ν^g_A). Thus, $\upsilon^*_A = \pi^* - \nu^g_A$ = \$6 - (\$-2) = \$8. To assess υ^*_{-A} we remove Alice offer. The best configuration will definitely have Bob and Eve (since Eve is paying more than Frank and Bob was happy with that what Frank was paying). Furthermore, since Frank's offer covers the price requested by Carol, the best configuration also includes Frank and Carol, and thus υ^*_{-A} is \$4, setting the local social value for good g of Alice for good g to $SV_g(A) = \upsilon^*_A - \upsilon^*_{-A}$ =\$4.

Having defined what a participant local social value is for a given good and taking the messages from participants as offers, next we make two connections:

- The message sent from a mediator to participant in CHAINME approximates the participant's local social value for that mediator.
- The message sent from a participant to a mediator in CHAINME approximates the marginal social value of that participant.

We detail these two connections in the rest of this section.

Messages to participant agents approximate their local social value. To show how the message from a mediator to a participant approximates her

local social value, first we detail how a mediator for good g computes the local social value for every participant it is linked to depending on whether the agent is active in the π^*-configuration or not. A general mediation scenario is described in Section 2.1. Consider first the case of any seller s^k active in the π^*-configuration. The benefit when s^k is active, $v^*_{s^k}$, is simply $\pi^* - \nu^g_{s^k}$. Notice that we subtract the message from the seller to the good's mediator ($\nu^g_{s^k}$) because it should not be considered benefit to the other participants but to s^k itself. To assess $v^*_{-s^k}$ we need to remove offer $\nu^g_{s^k}$ and recompute the best configuration. All pairs (s^i, b^i) with $i < k$ are profitable because they were profitable before removing seller s^k. Furthermore, all pairs (s^{i+1}, b^i) with $k \leq i < \eta$ are also profitable, since each pair (s^{i+1}, b^{i+1}) was profitable before removing s^k and the offer for b^i is at least as good as that of b^{i+1}. To assess the best attainable benefit after removing s^k, we must consider whether: (i) to add a new seller (namely $s^{\eta+1}$); or (ii) to remove one of the current buyers (namely b^η). Thus, we have that $v^*_{-s^k} = \pi^* - \nu^g_{s^k} - \min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta})$. Moreover, the local social value for an active seller s^k for good g is $SV_g(s^k) = \min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta})$. Therefore, a mediator can compute the active seller's local social value as the ask price τ^+.

Following this line of reasoning, a mediator for good g can assess the local social value for any seller s as:

$$SV_g(s) = \begin{cases} \min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta}) = \tau^+, & \text{if } s \in \mathcal{S}^a_g \\ \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}) = \tau^-, & \text{otherwise,} \end{cases} \tag{5.8}$$

and the local social value for any buyer b as:

$$SV_g(b) = \begin{cases} -\max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}) = -\tau^-, & \text{if } b \in \mathcal{B}^a_g \\ -\min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta}) = -\tau^+, & \text{otherwise.} \end{cases} \tag{5.9}$$

Recall that $\mathcal{S}^a_g$ and $\mathcal{B}^a_g$ stand for the set of active sellers and buyers for good g respectively.

Notice that there is a direct correspondence between equations 5.8 and 5.9 and the equations 5.6 and 5.7 used to compute messages from mediators to participants in CHAINME.

Messages from participants approximate their marginal value. To show how the messages from a participant to a mediator agent approximate the participant's marginal value for that good, first we detail how a participant computes her value when there are complementarities over a set of goods (the activation cost of a participant depends on acquiring a set of input goods and selling a set of output goods). In this case, we define the value of a participant p as her activation cost (C_p) plus the local social values of that participant for each good is connected to. Formally,

$$V_p = C_p + \sum_{g \in \mathcal{N}(p)} SV_g(p) \tag{5.10}$$

In this case, since there is more than one good involved, we can also define the *marginal value* for a good as the participant value without the value contributed by g. The message from a participant to a good's mediator is her *marginal value* for that good, namely her value without the value contributed by g, to signal her significance in the supply chain excluding g. This amounts to subtracting the local social value of p for g from the agent's value. Therefore, the marginal value of a participant p for a good g is:

$$V_{p-g} = V_p - SV_g(p) \tag{5.11}$$

Notice that, taking the messages from mediators as local social values, there is a direct correspondence between Equation 5.11 and Equation 5.5 used to compute messages from a participant to a mediator in CHAINME.

5.3 Complexity analysis

In this section, in order to complete the complexity analyses provided in Sections 3.2.1 and 4.3, we provide worst-case bounds on the amount of memory, the size of the messages exchanged at each iteration, and the computation time needed by CHAINME agents. Moreover, we compare CHAINME's complexity with the state of the art for mediated SCF, SAMP-SB-D, and our own contribution for P2P SCF, RB-LBP. Hereafter, we 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.

Computation. At each iteration, each participant needs to assess messages for G mediator agents. Moreover, if we make use of Equation 5.11, each participant requires only $\mathcal{O}(G)$ operations to assess those messages. On the other hand, the costliest operation for a mediator agent is ordering the messages received from the participant agents, which takes $\mathcal{O}(P \cdot \log P)$ operations per iteration.

Memory. Each participant agent needs to store the last message received from each of the mediator agents it is connected to plus its activation cost. Therefore, the memory requirements for a CHAINME participant agent are $\mathcal{O}(G)$. On the other hand, each mediator agent needs to store the messages received from each of the participant agents it is connected to, namely $\mathcal{O}(P)$ memory.

Communication. Each participant agent communicates exclusively with the mediator agents for the goods it is interested in buying or selling. Moreover, a participant is interested in at most G goods and each message exchanged contains a single value. Therefore, each participant agent needs to send $\mathcal{O}(G)$ messages per iteration. Analogously, each mediator agent is connected to at most P participant agents with whom it exchanges a single value. Therefore, each mediator agent needs to send $\mathcal{O}(P)$ in the worst case.

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 5.1: Worst case resource requirements.

Table 5.1 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 $\mathcal{O}(\log P)$ [Wurman et al., 1998]. Finding other values for SAMP-SB-D is direct from its description in [Walsh and Wellman, 2003]. Values for RB-LBP are those provided in Chapter 4. CHAINME closely matches memory and communication worst case requirements of both SAMP-SB-D and RB-LBP. Moreover, CHAINME requires less resources than RB-LBP in terms of number of operations and a factor of P more resources than SAMP-SB-D. However, as we show in the next section, CHAINME needs less iterations to find a solution. This results in CHAINME finding solutions to the SCF using significantly less resources than both SAMP-SB-D and RB-LBP.

5.4 Experimental evaluation

In this section we benchmark CHAINME against the state-of-the-art decentralized SCF algorithms: SAMP-SB-D, and RB-LBP. We aim at providing a quantification of CHAINME's resource requirements as well as the quality of the solutions that our method finds and to contrast these results with the worst case analysis conducted in the previous version. First, we describe SAMP-SB-D and CHAINME implementation details and the tools used in the process. Then, we detail our experimental design. Finally, we describe the metrics that we used to measure the performance of these methods for decentralized SCF, and provide a thorough analysis of the collected data.

5.4.1 Implementation mediated Supply Chain Formation algorithms

The implementation of the CHAINME algorithm was conducted in a similar way as that for RB-LBP described in Section 4.4.1. That is, CHAINME was implemented as an extension of libDAI [Mooij, 2010], an open source C++ library for probabilistic inference. Since CHAINME is a specialization of the standard max-sum algorithm implemented in libDAI, some features were added to the

library. Namely,

1. **Local terms as functions.** While libDAI's terms need to be stored in memory as tables, CHAINME's do not since they can be expressed as functions. Therefore, our first extension to libDAI was to implement CHAINME local terms as functions, thus reducing memory requirements.

2. **Simplified message calculation.** All the messages from a mediator agent to all the participant agents it is connected to depend on the same two values, namely τ^+ and τ^-. Therefore, instead of calculating each message separately as happens with libDAI's local terms, our implementation of CHAINME's equilibrium terms includes the precalculation of τ^+ and τ^- prior to sending the messages to each of the participants. Thus, by extracting the part of message calculation that is common for all the neighbors of an equilibrium term, we reduce CHAINME's computational requirements with respect to its standard max-sum counterpart.

3. **Solution assessment.** Recall from Section 5.2 that the assessment of solutions after CHAINME's max-sum phase has finished is specific to this algorithm. As a consequence of this, we implemented CHAINME's solution assessment as an additional procedure after the termination of the max-sum phase.

Finally, SAMP-SB-D algorithm was implemented in C++ following the description of the algorithm in [Walsh and Wellman, 2003]. The code for all three algorithms can be freely downloaded from [Penya-Alba et al., 2014a].

5.4.2 Experimental design

We are interested in evaluating the performance of CHAINME against SAMP-SB-D, the state of the art in mediated SCF. Moreover, we are interested in evaluating the benefits, if any, of mediated SCF with respect to P2P SCF. Our experimental design follows that of Section 4.4.2. That is, we use the test-suite described in [Vinyals et al., 2008] to generate SCF problems. Moreover, we generate SCs with 50 goods and a number of participants ranging from 40 to 500. For each scenario we generate 100 different SCs. The generated problems can be downloaded from [Penya-Alba et al., 2014a].

Each of the problems is solved with the implementations of SAMP-SB-D, RB-LBP, and CHAINME described in the previous section. Our tests were run on an Intel(R) Xeon(TM) CPU running at 3.20GHz with 2GB of RAM on linux-2.6 x86 64. For each problem we record the following values:

Bandwidth. While CHAINME and RB-LBP agents exchange single-valued messages, SAMP-SB-D's mediator agents send multiple-valued messages to the participant agents they are connected to. Thus, for each agent involved in the SCF process, be it a participant or a mediator agent, we record the number of messages sent times the size of the message.

Computation. We measure the computation needed by an agent as the number of operations the agent performs[1].

Solution quality. We normalize solution quality to the 0-1 scale by dividing the value of the SCs found by the different methods by the value of the optimal solution.[2] Hence, a quality of one means that the solution found is optimal.

Recall form Section 5.3 that the memory requirements for agents in all the three algorithms compared is proportional to the number of neighbors the agent is connected to. Moreover, the memory used by these agents does not increase during the execution of each of the algorithms. Therefore, we do not record the memory requirements since they will produce similar results for all three methods. In what follows, we provide an analysis of the results obtained after running the experiments and recording the results obtained.

5.4.3 Empirical evaluation

In this section we study the scalability of CHAINME and compare it with that of SAMP-SB-D and RB-LBP. Following the empirical evaluation in Section 4.4.3 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 in this protocol [Walsh and Wellman, 2003].

The distributions obtained for the measures described in the previous section are long-tailed and skewed. Therefore we use the median instead of the mean as a measure of central tendency following the recommendations in [Wilcox and Keselman, 2003]. Where possible we do also show the 20th and 80th percentile as a measure of dispersion. First, we analyze the resource requirements of each algorithm. Then, we analyze the quality of the solution obtained (in terms of the value of the SC) by each algorithm.

Resource requirements

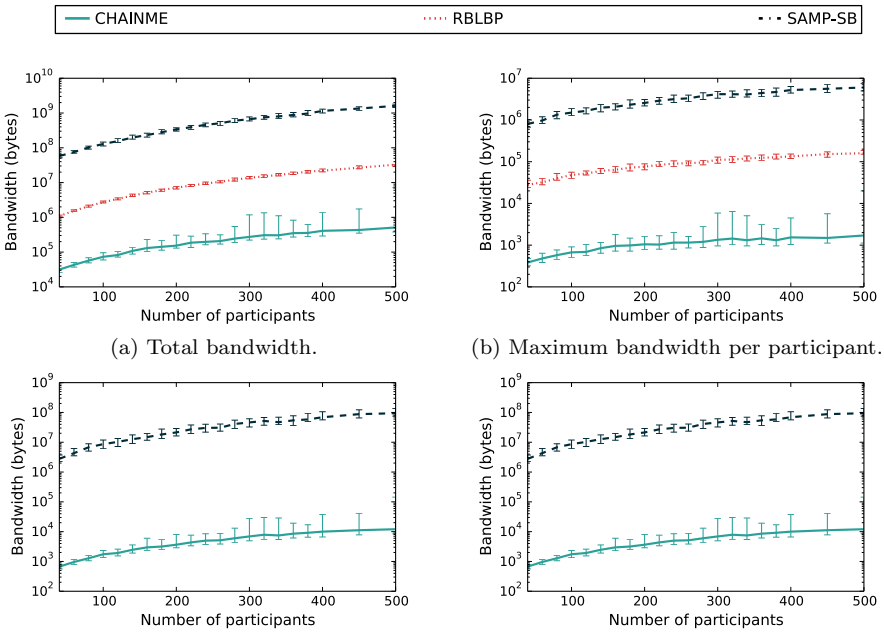
Recall that CHAINME and SAMP-SB-D are mediated algorithms whilst RB-LBP is not. That is, CHAINME and SAMP-SB-D employ mediator agents alongside participant agents, whereas in RB-LBP there are only participant agents. For that reason we benchmark bandwidth usage along four different dimensions: total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

[1] A fair assessment of the amount of computation performed by mediators is difficult. The costliest operation for both SAMP-SB-D and CHAINME mediators is assessing the bid-ask interval. CHAINME uses algorithm 1 whose worst-case complexity is $\mathcal{O}(P \cdot \log P)$ (P stands for the number of participants a mediator is connected to). Thus, we record $P \cdot \log P$ operations each time the bid-ask interval is assessed in CHAINME. On the other hand, the assessment of the bid-ask interval in SAMP-SB-D can be done in $\mathcal{O}(\log P)$ following [Wurman et al., 1998]. Hence, we record $\log P$ operations each time the bid-ask interval is assessed in SAMP-SB-D.

[2] The optimal solutions are found using a centralized mixed integer programming solver.

total bandwidth used by mediators, and maximum bandwidth used by any mediator.

Figure 5.4 shows how the different algorithms performed in terms of bandwidth usage. Note that RB-LBP is left out of Figures 5.4c and 5.4d due to its lack of mediator agents. In Figure 5.4a we see that CHAINME consumes at most 1/60th of the bandwidth used by RB-LBP and that the savings when compared to SAMP-SB-D are of at least three orders of magnitude. Moreover, as shown in Figure 5.4b, these savings remain even when considering only participant agents. Furthermore, the savings increase from three to four orders of magnitude when considering only mediator agents (Figures 5.4c and 5.4d).



(a) Total bandwidth.

(b) Maximum bandwidth per participant.

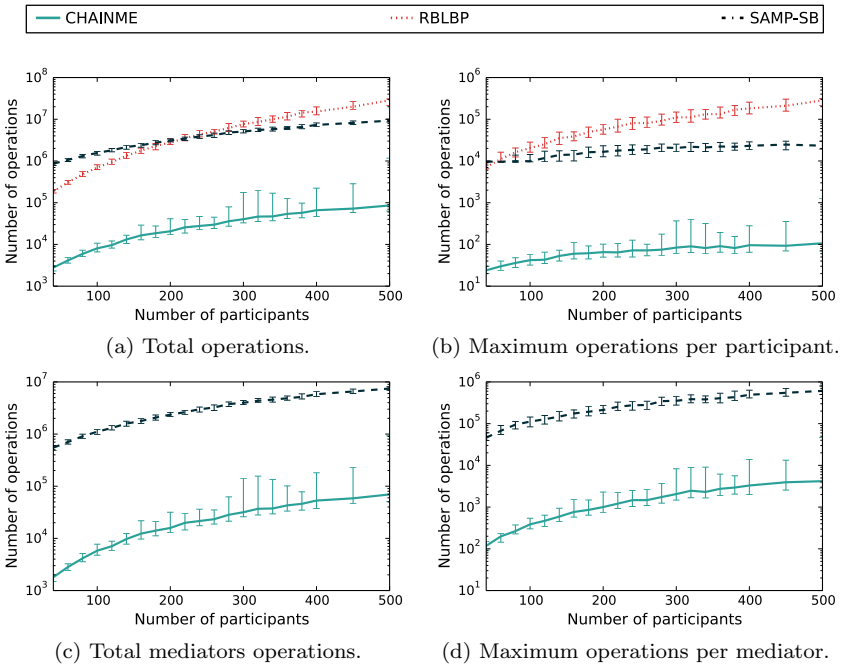
(c) Total mediators bandwidth.

(d) Maximum bandwidth per mediator.

Figure 5.4: CHAINME, RB-LBP, and SAMP-SB-D bandwidth requirements. Plots use a log-scale for the y axis.

Next, we turn our attention to the computational requirements of the three algorithms at hand (SAMP-SB-D, RB-LBP and CHAINME). Figure 5.5 shows how the different algorithms performed in terms of computation. Again, RB-LBP is left out of Figures 5.5c and 5.5d due to its lack of mediator agents. In Figures 5.5a and 5.5b we see that the number of operations performed by CHAINME is at least 2 orders of magnitude smaller than that of the runner-up. This difference is confirmed when we only consider mediators in Figures 5.5c and 5.5d.

Given the complexity results provided in Section 5.3 it might seem unexpected that CHAINME performs better than SAMP-SB-D in terms of computational requirements. However, SAMP-SB-D message update policies can slow down the



(a) Total operations.

(b) Maximum operations per participant.

(c) Total mediators operations.

(d) Maximum operations per mediator.

Figure 5.5: CHAINME, RB-LBP, and SAMP-SB-D computations. Plots use a log-scale for the y axis.

algorithm by increasing the number of iterations needed to find a solution as discussed in Section 3.2.2. This increase on the number of iterations can be observed in Figure 5.6. Notice that SAMP-SB-D requires more than one thousand more iterations than CHAINME to find a solution. Moreover, RB-LBP fails to find a solution within the limit of 250 iterations whereas CHAINME finds solutions well bellow this limit. This phenomenon can be explained by the fact that CHAINME's additive decomposition produces local term graphs that contain less cycles than those produced by RB-LBP as argued in Section 5.1 (which is a known factor for max-sum convergence [Weiss and Freeman, 2001]).

Solution quality

In this section we focus our attention on the quality of the solutions assessed by CHAINME. Figure 5.7a shows the median and dispersion of the solution quality for CHAINME, RB-LBP, and SAMP-SB-D. We observe that both mediated methods (CHAINME and SAMP-SB-D) outperform RB-LBP (which implements a P2P architecture). Moreover, the quality of the solutions assessed by RB-LBP rapidly decreases as the number of participants in the SC increases. On the other hand, both CHAINME and SAMP-SB-D find solutions that are close to the optimal one although the later shows a slight decrease on the quality of the solution as the number of participants increases.

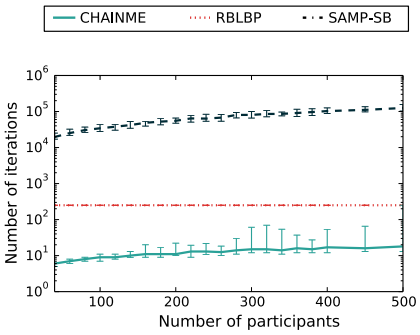
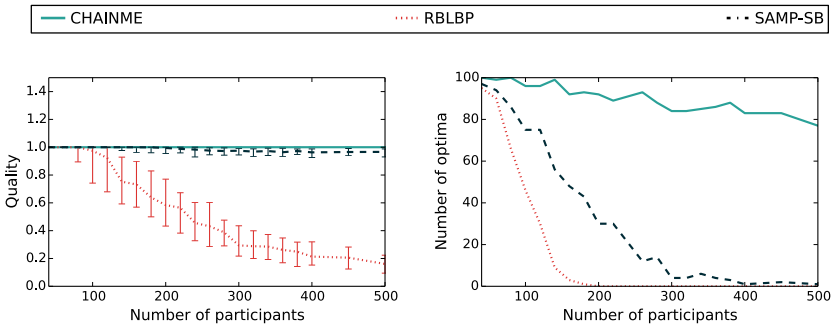


Figure 5.6: CHAINME, RB-LBP, and SAMP-SB-D iterations to convergence.



(a) Median quality.

(b) Number of optimal solutions.

Figure 5.7: CHAINME, RB-LBP, and SAMP-SB-D solution quality.

Figure 5.7b plots the number of problems for which each method was able to find the optimal solution. The number of problems optimally solved by SAMP-SB-D and RB-LBP decreases very rapidly as the number of participants increases. By contrast, CHAINME is able to find the optimal solution in most of the problems, even in scenarios with a large number of participants. In the 500 participants scenario, CHAINME converges to the optimal solution in more than 70% of the problems, whereas the other methods almost never find it.

These experiments place CHAINME as the best performing algorithm for decentralized SCF. On the one hand, it requires less computational resources than the state-of-the-art. Moreover, CHAINME reduces the bandwidth required to find a solution to no more than 1/60th of the runner-up. Furthermore, the number of operations required by CHAINME to assess solutions is at least two orders of magnitude smaller than those required by the next best performing algorithm. On the other hand, CHAINME is able to find solutions that are of higher quality than those found by the other methods. Moreover, CHAINME finds the optimal solution in over 70% of the instances even for problems with 500 participants.

5.5 Conclusion

In this chapter we have introduced CHAINME, a novel method for mediated SCF. CHAINME is based on an additive decomposition of the SCF problem that can be approximated by the max-sum algorithm. Moreover, we have provided computationally efficient expressions to assess max-sum messages that significantly lower the resources required by CHAINME with respect to standard max-sum. The experimental evaluation of the CHAINME algorithm shows that it is able to find solutions requiring less than 1/60th of the bandwidth required by RB-LBP (our method for P2P SCF), and that the savings in terms of bandwidth are of at least three orders of magnitude when compared with SAMP-SB-D (the state of the art for mediated SCF). Moreover, CHAINME savings in terms of number of operations are of at least two orders of magnitude when compared with RB-LBP and SAMP-SB-D. Finally, these savings do not come at the expense of the solution quality. That is, CHAINME is able to find solutions that are closer to the optimal one than the other methods and finds the optimal solution more often even for lager problems.

Chapter 6

Conclusions and Future Work

In this chapter, we draw some conclusions about the work developed in this dissertation and we show some paths open to future development.

6.1 Conclusions

The Supply Chain Formation (SCF) problem has been widely studied in the multi-agent systems literature. Moreover, as we discussed in Chapter 1, the majority of contributions tackle the SCF problem in a centralized manner. However, as argued in Section 3.1, employing a centralized solution might not be satisfactory for the following reasons:

- Centralized approaches rely on a central authority that possesses the private valuations of all participants. However, participants might be reluctant to share this information with any central authority.
- Centralized optimal solvers might suffer from scalability issues given the hardness of the SCF problem.
- The existence of a central authority introduces a single point of failure for the SCF process.

In order to overcome of these limitations, as discussed in Section 3.2, decentralized SCF appears as an attractive alternative. Decentralized approaches in the literature to the SCF problem can be categorized under two categories: peer-to-peer (P2P) and mediated. In P2P SCF, participant agents interact directly with each other without the need of any third party. On the other hand, in mediated SCF each of the goods in the Supply Chain (SC) is represented by a computational agent that mediates between the participant agents interested in buying or selling the good it is in charge of.

Although solving the SCF problem in a decentralized manner has been studied in the literature both in a P2P and in a mediated manner, current state-of-the-art methods for decentralized SCF incur in high computational penalties. Moreover, there is no comparison in terms of economical efficiency of these methods. With this background, in Chapter 1, we posed the following open research questions:

- Can decentralized SCF methods be economically efficient?
- Can decentralized SCF methods be computationally efficient?

With the goal of answering these questions, in this thesis, we have contributed to the state of the art with two algorithms for decentralized SCF that are computationally efficient and provide higher quality solutions to the SCF problem than the current approaches in the state of the art. One of the main contributions of this thesis was RB-LBP, an algorithm to solve the SCF problem following a P2P architecture. On the other hand, we contribute to the state of the art in mediated SCF with CHAINME, an algorithm to solve the SCF problem in a decentralized manner with the use of mediators for goods. Both contributions are based on the max-sum algorithm and exploit the particularities of our encodings in order to reduce computational complexity. Moreover, the methodology we applied to both RB-LBP and CHAINME has proven to be effective signaling its potential for other coordination problems.

Next, in Section 6.1.1, we provide a summary of the methodology followed to develop both RB-LBP and CHAINME. Then, in Section 6.1.2, we provide a detailed summary of RB-LBP, our contribution to P2P SCF. Finally, in Section 6.1.3, we summarize in more detail CHAINME, our contribution to mediated SCF.

6.1.1 A methodology for the design of decentralized decision making algorithms

We decided to tackle the decentralized SCF problem by means of decentralized decision making algorithms based on max-sum that are run by a multi-agent system. We set our goal to develop a methodology to build a local term graph over which the max-sum algorithm could run in a computationally efficient manner in order to approximate an objective function that decomposes additively. Recall from Section 2.2 that max-sum is a message passing algorithm that has shown good empirical performance in a wide range of multi-agent coordination scenarios. Unfortunately, assessing standard max-sum messages takes exponential time. However, it is possible to reduce its time complexity by studying the particularities of specific local terms as described in Section 2.2.4. In order to design and implement computationally efficient solutions to the SCF problem both in P2P and mediated scenarios we adopted a common methodology. Our methodology consists in the following steps:

1. **Identify decision makers.** Each decision maker in a SC is represented by a computational agent.

2. **Encode decisions in binary variables.** For each of the decision makers identified in the previous step, break down its decisions into binary ones. By encoding each decision in a binary variable we avoid the problem of coupling several decisions into a single variable.

3. **Ensure coherence.** Introduce constraints in the form of local terms in order to ensure that only feasible combinations of variables' states remain acceptable.

4. **Assign variables and local terms to agents.** Assign the responsibility of sending and receiving the max-sum messages for each variable and local term to the corresponding agent identified at step 1.

5. **Efficient computation of messages.** Apply the technique described in Section 2.2.4 to the local terms obtained at step 3 in order to reduce standard max-sum's exponential time complexity.

6. **Decode max-sum's solution.** Since max-sum can converge to unfeasible solutions or even fail to converge, devise a domain specific method for removing unfeasibilities in the solutions found by the max-sum algorithm.

Notice that the steps described above are not exclusive to the SCF problem and can be applied to more general optimization problems. Therefore, we believe that applying the methodology followed in this work can reduce the design and implementation time of solutions to other multi-agent system coordination problems.

In the next sections we provide a summary of how we applied this methodology to the SCF problem and the results it yielded.

6.1.2 Peer-to-peer Supply Chain Formation

As described in Section 3.2.1, in P2P SCF there is a participant agent representing each of the participants in the SC. Moreover, each participant agent communicates only with the agents representing the sellers of the goods its participant is interested in buying and the buyers of the goods its participant is interested in selling.

As we show in Section 3.2.1, the computational requirements of LBP, the state of the art in P2P SCF, grow exponentially with the size of the problem, thus hindering the scalability of this method.

In order to provide a computationally efficient algorithm for P2P SCF, in Chapter 4, we proposed RB-LBP. RB-LBP is designed following the methodology described in the previous section as we detail next.

The first step in our methodology is that of identifying decision makers. Since we are dealing with a P2P SCF, it turns out clear that the decision makers in our scenario are the SC participants themselves.

To encode decisions in binary variables (step 2), we encode each participant's decision to take part in the SC in a binary variable. Moreover, we decouple each buy and sell decision into a separate variable.

Furthermore, we ensure internal coherence (step 3) of each participant by means of selection terms and coherence among participants by means of equality terms. That is, each selection term relates to the buy or sell decision of a certain good and ensures that an active participant chooses only one partner for the good in question and each equality term ensures that potential partners agree on whether to exchange a certain good or not.

In order to apply the max-sum algorithm to a multi-agent system, it is necessary to assign each variable and local term to a computational agent (step 4). Therefore, in RB-LBP, there is a participant agent representing each of the participants in the SC. Moreover, each participant agent, is responsible for the variables encoding its participant's decisions and the local terms constraining them. The local terms ensuring coherence among each pair of potential partners are shared by both participant agents.

Applying the technique described in Section 2.2.4 to reduce the computational complexity of max-sum message assessment (step 5) results in an improvement in worst case resource requirements over LBP, the state-of-the-art method for P2P SCF. Specifically, with RB-LBP, we reduce memory and bandwidth requirements from exponential to linear, and the number of operations per iteration from exponential to quadratic.

Finally, in order to decode max-sum's solutions (step 6), in RB-LBP, we simply remove from the solution any participant that is not able to buy all of its input goods and sell its output goods.

With the purpose of evaluating RB-LBP's performance we conducted an experimental analysis. The results show that RB-LBP is able to find higher-valued solutions while reducing the memory, bandwidth and computational resources required by several orders of magnitude with respect to LBP, the state-of-the-art method for P2P SCF. However, as a result of the experiments, we observed that the quality of the solutions found by both LBP and RB-LBP degrades as the number of participants in the SC increases. Moreover, we linked this behaviour to the number of cycles in the local term graphs produced by both methods. Furthermore, we argued that this is a consequence of applying the max-sum algorithm to the P2P architecture.

Summary of RB-LBP contributions

To summarize, our main contributions to the state of the art in P2P SCF are:

- A novel encoding of the SCF into a local term graph containing only binary variables.
- A mapping of the local term graph into a multi-agent system that allows to solve the SCF problem following a P2P architecture.
- A set of expressions to assess the messages exchanged between the participant agents in a computationally efficient manner.

- A P2P algorithm for SCF that requires several orders of magnitude less memory, bandwidth and computational resources, and finds better valued solutions than LBP, the state of the art for P2P SCF.

Thus, with RB-LBP, we answer the first of our research questions by showing that decentralized SCF can be computationally efficient in the P2P setting. However, the question about economical efficiency remains open. We try to address this question with our contribution to mediated SCF.

6.1.3 Mediated Supply Chain Formation

As described in Section 3.2.2, in mediated SCF there is a participant agent representing each participant in the SC. Moreover, for each of the goods in the SC, there is a mediator agent that acts as a local market for that good. Therefore, participant agents communicate only with the mediator agents for the goods they are interested to buy or sell.

Although in Chapter 4 we provided a computationally-efficient method for decentralized SCF, the quality of the solutions found by this method degrades as the size of the problem increases. On the other hand, in Section 3.2.2, we argued that, although being able to achieve high economic efficiency, SAMP-SB-D, the state of the art for mediated SCF, incurs in a high penalty in terms of communication.

Thus, in Chapter 5, we proposed CHAINME, a mediated method for SCF. CHAINME is designed following the methodology described in Section 6.1.1 as we detail next.

The first step in our methodology is that of identifying decision makers (step 1). Since CHAINME is targeted at mediated SCF, it turns out clear that the decision makers in our scenario are both the participants in the SC and the mediators for each of the goods in the SC.

To encode participant's decisions (step 2) we introduce a binary variable for each of the participants encoding whether she is active in the SC or not. On the other hand, we ensure coherence (step 3) by means of equilibrium terms, one per each good at trade, that enforce that the number of active sellers is equal to the number of active buyers for each of the goods at trade.

In order to solve the SCF problem by means of a max-sum based multi-agent system, it is necessary to assign each variable and local term to a computational agent (step 4). Therefore, in CHAINME, there is a participant agent representing each of the participants in the SC. Moreover, each participant agent is responsible for its participant's variable. On the other hand, for each good in the SC there is a mediator agent that is responsible for the equilibrium term constraining the exchanges of that good.

Applying the technique described in Section 2.2.4 to reduce the computational complexity of max-sum's message assessment (step 5) results in CHAINME requiring the same worst case memory and communication resources as SAMP-SB-D the state-of-the-art method for mediated SCF, and RB-LBP. Moreover, CHAINME reduces the operations required per iteration with respect to RB-LBP.

Finally, to decode max-sum's solutions (step 6), each mediator agent in CHAINME matches the best buyers and sellers for the good it is representing and marks them to be active. Then, each participant agent decides to be active in the SC if it is able to buy all of its input goods and sell all of its output goods.

With the purpose of quantifying CHAINME's savings in terms of computational resources and evaluating the quality of the SC configurations found by CHAINME we conducted an experimental analysis. The results show that, when compared against both RB-LBP and SAMP-SB-D, CHAINME requires a fraction of the bandwidth an operations needed by the next best performing algorithm and has the same memory requirements. Moreover, CHAINME finds SC configurations of higher value than the competition. Furthermore, the value of the solutions found by CHAINME is within 98% of the optimal value.

Summary of CHAINME contributions

To summarize, our main contributions to the state of the art in mediated SCF are:

- A novel encoding of the SCF into a local term graph containing only binary variables.
- A mapping of the local term graph into a multi-agent system that allows to solve the SCF problem following a mediated architecture.
- An efficient computation of messages exchanged between the participant agents and the mediator agents.
- A method that allows to find solutions to the SCF problem in a decentralized manner reducing the memory requirements to 1/60th, and the number of operations by two orders of magnitude with respect to the state of the art while finding solutions that are close to the value of the optimal one.

Thus, with CHAINME, we answer both of our research questions by showing that decentralized SCF can be computationally and economically efficient in a mediated setting. However, as we argue in the next section there are many paths for future development in the area of decentralized SCF.

6.2 Future work

In this thesis we have tried to make headway in the decentralized assessment of Supply Chains (SCs) both in peer-to-peer (P2P) and mediated scenarios. This thesis also reveals several paths to future developments both on decentralized Supply Chain Formation (SCF) and on improving the results obtained by applying the max-sum algorithm to other coordination problems. Next, we elaborate on both lines of future work.

6.2.1 Decentralized Supply Chain Formation

Although we have contributed to the state of the art on decentralized SCF with two computationally efficent algorithms, there is room from improvement on the quality of the solutions found by current P2P SCF methods. Recall that the value of the SC configurations found by both LBP [Winsper and Chli, 2013] and RB-LBP (Chapter 4) degrades as the number of participants in the SC increases. Both methods are based on the max-sum algorithm and, as we argued in Section 4.5, applying this algorithm to P2P SCF results in graphs that contain cycles which is known to affect negatively max-sum's performance. Therefore, we think it would be interesting to study how other techniques from the Decentralized Constraint Optimization community perform in the SCF problem. Moreover, both LBP and RB-LBP ensure feasibility of the solutions by just removing participants from the solution until it is feasible. Thus, more sophisticated decoding methods could lead to better valued solutions.

In general, all the methods reviewed in this thesis (including RB-LBP and CHAINME) that tackle the SCF problem in a decentralized manner lack the expressivity found in centralized approaches such as [Collins et al., 2002], [Cerquides et al., 2007], and [Witzel and Endriss, 2012]. That is, none of the reviewed methods takes into account features such as temporal constraints or allowing participants to express bids that contain multiple copies of the same transformation, transformations that take (or produce) multiple units of the same good, offer different bundles of transformations, and different prices for a transformation depending on how many times it is performed (bulge discounts). Therefore, we think it would be interesting to study how to add expressiveness to current decentralized methods for SCF while keeping their computational and economical efficiency. For instance, it would be possible to add this features by means of additional constraints to the methods we proposed.

Finally, in this thesis we have provided the means to solve the Winner Determination Problem in a decentralized manner. However, since no payment function has been defined, the design of a mechanism would require the definition of one such payment function. Therefore, the design of this function and the analysis of the properties of the corresponding mechanisms should be pursued as future work.

6.2.2 Improving the performance of max-sum

The decoding of the solutions found by the max-sum algorithm is a key aspect to consider when using max-sum to solve an optimization problem since max-sum may converge to solutions that break some constraints. So far, max-sum based methods in the literature [Givoni and Frey, 2009, Kim et al., 2010, Winsper and Chli, 2013] tend to include a domain specific decoding phase in order to turn the solutions found by the max-sum algorithm into feasible ones. The design of this decoding phase is not trivial and the quality of the final solution can be greatly affected by the method chosen. Therefore, max-sum based methods would benefit from the study of general solution decoding methods.

Finally, the design and implementation of both of our contributions to decentralized SCF follow the methodology described in Section 6.1.1. Although our methodology proved to be valid for the design of SCF algorithms, its generality makes it appear as a promising candidate for other multi-agent coordination problems. A key aspect of our methodology is providing computationally efficient message computation for max-sum local terms. Moreover, in this thesis we have provided computationally efficient message computation for three new local terms. Furthermore, we can find in the literature more examples of these computationally efficient local terms [Givoni and Frey, 2009, Tarlow et al., 2010, Pujol-Gonzalez et al., 2013a, Pujol-Gonzalez et al., 2013b]. Since these local terms can be understood as logical operators (e.g. select one variable, all variables must take the same value, etc), we think it could be possible to build a general framework to solve coordination problems following our methodology and using these local terms as building blocks. That is, a designer could build a solver to a coordination problem just by "plugging" together these building blocks without the need to understand the inner workings of the max-sum algorithm, thus speeding up the design of solvers. An interesting take on building such a framework is the Binary MaxSum library [Pujol-Gonzalez and Penya-Alba, 2014]. However, as of today, this library is limited to a collection of computationally efficient local terms. We think it would require the implementation of a general method for decoding solutions and a design tool to specify problems in order to turn this library in the framework described above.

Appendix A

RB-LBP Efficient Message Computation

We devote this chapter to provide computationally efficient expressions to assess the value of the messages exchanged in RB-LBP's local term graph described in Chapter 4. Therefore, in Section A.1, we provide a description of the expressions used in standard max-sum to assess the messages exchanged between vertices in the local term graph. Then, in Section A.2, we study the particularities of RB-LBP's local terms. Finally, in Section A.3 we provide computationally efficient expressions to assess the single-valued messages exchanged between RB-LBP's local terms.

A.1 Message exchange in standard max-sum

The max-sum algorithm consists of a series of message exchanges between simple and composite vertices in a *local term graph*. Recall from Section 2.2.1 that a simple vertex is a vertex representing a local term whose scope is a single variable, whereas a composite vertex is a vertex representing a local term whose scope is two or more variables. Moreover, the message exchange between vertices is repeated until the algorithm either converges or is stopped. Furthermore, recall from Section 2.2.2 that, in standard max-sum, the message from a composite vertex f to a simple vertex x is assessed by Equation A.1.

$$\mu_f^x(\mathbf{x}) = \max_{\mathbf{X_{f\setminus\{x\}}}} \left\{ f(\mathbf{X}_f) + \sum_{x' \in N(f)\setminus\{x\}} \mu_{x'}^f(\mathbf{x}') \right\} \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{A.1}$$

where $\mathbf{X}_{f\setminus\{x\}}$ can take every possible state of every variable in the scope of local term f except x, f is the local term associated to the composite vertex, and $N(f) \setminus \{x\}$ is the set of neighbors of f excluding x.

On the other hand, the message from a simple vertex x to a composite vertex

f (μ_x^f) is given by Equation A.2.

$$\mu_x^f(\mathbf{x}) = f_x(\mathbf{x}) + \sum_{f' \in N(x) \setminus \{f\}} \mu_{f'}^x(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{A.2}$$

where f_x is the simple term associated to x and $N(x) \setminus \{f\}$ is the set of neighbors of x excluding f.

Notice that these messages depend not only on the messages received from neighboring vertices but also on the value taken by the local term they represent. Moreover, the computation of messages from composite vertex to simple vertex takes exponential time. However, as argued in Section 2.2.4 this complexity can be reduced by studying the particularities of each particular local term. In the next section, we study the particularities of RB-LBP's local terms and provide slightly simplified expressions to assess the messages between RB-LBP's local term graph vertices.

A.2 Message computation

In this section we study the messages computed by the selection and equality terms employed by RB-LBP. Each of these local terms encode a hard constraint. That is, some of the possible assignments to the variables of a local term are unfeasible and can be discarded. Therefore, it is possible to simplify the maximum in Equation A.1 by not considering the assignments that break the constraint. Consequently, we provide slightly simplified versions of Equation A.1 for equality and selection terms both for the active and inactive states of a variable. For simplicity, we start with equality terms.

Computing the message from equality vertex to option vertex. Consider equality term f_E joining option variables o and o' described by the following equation:

$$f_E(\mathbf{o}, \mathbf{o}') = \begin{cases} 0, & \text{if } \mathbf{o} = \mathbf{o}' \\ -\infty, & \text{otherwise.} \end{cases}$$

Moreover, consider that the last messages received by vertex f_E from vertex o' is $\mu_{o'}^{f_E}$. We can readily assess the new message from f_E to o ($\mu_{f_E}^o$):

$$\mu_{f_E}^o(0) = \max_{\mathbf{o}'} \left(f_E(\mathbf{o}', 0) + \mu_{o'}^{f_E}(\mathbf{o}') \right).$$

Since $f_E(o, 0)$ has a value of $-\infty$ if $\mathbf{o}' = 1$, we can assure that $\mathbf{o}' = 0$ is the assignment that maximizes the expression, and thus

$$\begin{aligned} \mu_{f_E}^o(0) &= f_E(0, 0) + \mu_{o'}^{f_E}(0) \\ &= \mu_{o'}^{f_E}(0). \end{aligned} \tag{A.3}$$

In a similar way we obtain:

$$\mu_{f_E}^o(1) = \mu_{o'}^{f_E}(1). \tag{A.4}$$

Note that the message from f_E to o is the last message received by f_E from o'. Since the equality term is symmetrical, the message from f_E to o' is the last message received by f_E from o. We reproduce bellow the equations for the message sent from an equality vertex to an option vertex, both for the inactive and the active states.

$$\boxed{\begin{array}{l}\mu^{o}_{f_E}(0) = \mu^{f_E}_{o'}(0)\\ \mu^{o}_{f_E}(1) = \mu^{f_E}_{o'}(1)\end{array}}$$

Computing the message from selection vertex to activation vertex. Consider a generic selection term f_S, joining an activation variable a with option variables $o_1, \ldots, o_n$ described by the following equation:

$$f_S(\mathbf{a}, \mathbf{o_1}, \ldots, \mathbf{o_n}) = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} \mathbf{o_i} = \mathbf{a} \\ -\infty, & \text{otherwise.} \end{cases}$$

Moreover, consider that the last messages received by vertex f_S from vertices $a, o_1, \ldots, o_n$, are respectively $\mu^{f_S}_{a}, \mu^{f_S}_{o_1}, \ldots, \mu^{f_S}_{o_n}$. We start by assessing the message from f_S to a, analyzing the case when $a = 0$:

$$\mu^{a}_{f_S}(0) = \max_{\mathbf{o_1},\ldots,\mathbf{o_n}} \big(f_S(0, \mathbf{o_1}, \ldots, \mathbf{o_n}) + \sum_{i=1}^{n} \mu^{f_S}_{o_i}(\mathbf{o_i})\big).$$

Since the activation variable is 0, the only possible combination of values returning a non-infinite value for f_S is setting all option variables o_i to 0 and thus

$$\mu^{a}_{f_S}(0) = \sum_{o \in \mathcal{N}(f_S) \setminus \{a\}} \mu^{f_S}_{o}(0), \tag{A.5}$$

where $\mathcal{N}(f_S) \setminus \{a\}$ is the set of option vertices connected to selection vertex f_S.

On the other hand, the message for $a = 1$ is:

$$\mu^{a}_{f_S}(1) = \max_{\mathbf{o_1},\ldots,\mathbf{o_n}} \big(f_S(1, \mathbf{o_1}, \ldots, \mathbf{o_n}) + \sum_{i=0}^{n} \mu^{f_S}_{o_i}(\mathbf{o_i})\big).$$

Since the activation variable is 1, the selection term is only satisfied when one of its option variables takes on value 1 and the remaining ones take on value 0. It follows that:

$$\mu^{a}_{f_S}(1) = \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \big(\mu^{f_S}_{o}(1) + \sum_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \mu^{f_S}_{o'}(0)\big), \tag{A.6}$$

where $\mathcal{N}(f_S) \setminus \{a, o\}$ is the set of option vertices connected to selection vertex f_S but option vertex o. We reproduce bellow the equations for the message sent from an selection vertex to an activation vertex, both for the inactive and the active states.

$$\mu^a_{f_S}(0) = \sum_{o \in \mathcal{N}(f_S)\setminus\{a\}} \mu^{f_S}_o(0)$$
$$\mu^a_{f_S}(1) = \max_{o \in \mathcal{N}(f_S)\setminus\{a\}} \big(\mu^{f_S}_o(1) + \sum_{o' \in \mathcal{N}(f_S)\setminus\{a,o\}} \mu^{f_S}_{o'}(0)\big)$$

Computing the message from selection vertex to option vertex. The message sent from vertex f_S to each option vertex o_i is assessed in the same way. We start with the case $o_i = 0$. For compactness we note O_{-i} the set of all option variables attached to local term f_S except o_i.

$$\mu^{o_i}_{f_S}(0) = \max_{\mathbf{a},\mathbf{O_{-i}}} \big(f_S(\mathbf{a}, \mathbf{o_i} = 0, \mathbf{O_{-i}}) + \mu^{f_S}_a(\mathbf{a}) + \sum_{o \in O_{-i}} \mu^{f_S}_o(\mathbf{o})\big).$$

There are two ways in which the selection constraint can be satisfied. Either by setting every variable to 0 or by setting activation variable a to 1 and selecting exactly one of the option variables to take on value 1. Therefore, the equation to assess the message from a selection vertex to an option variable will follow the pattern:

$$\mu^{o_i}_{f_S}(0) = \max(AllZeros, \max_{o' \in O_{-i}} Select(o')). \tag{A.7}$$

On the one hand, the expression obtained after setting all variables to zero, $AllZeros$, can be expressed as:

$$AllZeros = \mu^{f_S}_a(0) + \sum_{o' \in O_{-i}} \mu^{f_S}_{o'}(0). \tag{A.8}$$

On the other hand, the expression obtained after setting activation variable to one and only one selection variable to zero, $Select(o')$, can be expressed as:

$$Select(o') = \mu^{f_S}_a(1) + \mu^{f_S}_{o'}(1) + \sum_{o'' \in O_{-i}\setminus\{o'\}} \mu^{f_S}_{o''}(0). \tag{A.9}$$

Finally, in case $o_i = 1$, we have that:

$$\mu^{o_i}_{f_S}(1) = \max_{\mathbf{a},\mathbf{O_{-i}}} \big(f_S(\mathbf{a}, \mathbf{o_i} = 1, \mathbf{O_{-i}}) + \mu^{f_S}_a(\mathbf{a}) + \sum_{o \in O_{-i}} \mu^{f_S}_o(\mathbf{o})\big).$$

The only way to satisfy the selection constraint is by setting a to 1 and all the option variables other than o_i to 0. Thus,

$$\mu^{o_i}_{f_S}(1) = \mu^{f_S}_a(1) + \sum_{o' \in O_{-i}} \mu^{f_S}_{o'}(0). \tag{A.10}$$

Therefore, the messages from a selection vertex to an option vertex both for the inactive and the active states can be assessed by the following equations.

$$\boxed{\begin{aligned}\mu^{o_i}_{f_S}(0) &= \max(AllZeros, \max_{o' \in O_{-i}} Select(o')) \\ \mu^{o_i}_{f_S}(1) &= \mu^{f_S}_{a}(1) + \sum_{o' \in O_{-i}} \mu^{f_S}_{o'}(0)\end{aligned}}$$

Note that to assess the messages from composite to simple vertices participant agents do not need to store the tables representing composite terms. Since the size of a selection term grows exponentially with the number of options, the equations above yield large savings in memory requirements.

A.3 Message simplification

In this section we take the expressions to assess standard max-sum messages from the previous section and derive computationally efficient expressions to assess the equivalent single-valued messages exchanged between RB-LBP vertices. The derivation of the expressions for each message follows the same pattern. First, we take the difference between the message for the active state and the inactive state of a variable obtained in the previous section. Then, we remove terms that cancel out. Finally, we apply the following simplification:

$$\nu^{w}_{v} = \mu^{w}_{v}(1) - \mu^{w}_{v}(0). \tag{A.11}$$

We start by deriving the efficient messages from simple vertices to composite vertices. In general, the message from a simple vertex to a composite vertex can be assessed by means of the equation in the following lemma.

Lemma 1. *The single-valued message from a simple vertex x to a composite vertex g can be assessed as:*

$$\nu^{g}_{x} = f_x(1) - f_x(0) + \sum_{g' \in \mathcal{N}(x) \setminus \{g\}} \nu^{x}_{g'}. \tag{A.12}$$

Proof. We begin by taking the difference between the messages sent by simple vertex x to composite vertex f (Equation 2.4) for the active and the inactive states of x. From there, it is just a matter of expanding both messages and applying Equation A.11.

$$\begin{aligned}\nu_x^f &= \mu_x^f(1) - \mu_x^f(0) \\ &= \left(f_x(1) + \sum_{f' \in \mathcal{N}(x)\setminus\{f\}} \mu_{f'}^x(1)\right) - \left(f_x(0) + \sum_{f' \in \mathcal{N}(x)\setminus\{f\}} \mu_{f'}^x(0)\right) \\ &= f_x(1) - f_x(0) + \sum_{f' \in \mathcal{N}(x)\setminus\{f\}} \mu_{f'}^x(1) - \sum_{f' \in \mathcal{N}(x)\setminus\{f\}} \mu_{f'}^x(0) \\ &= f_x(1) - f_x(0) + \sum_{f' \in \mathcal{N}(x)\setminus\{f\}} \nu_{f'}^x.\end{aligned}$$

□

In what follows, we provide computationally efficient expressions to assess the single-valued message sent from an activation vertex to a selection vertex, from an option vertex to a selection vertex, and from an option vertex to an equality vertex.

Theorem 1. *The single-valued message from a participant's p activation vertex x_p to a selection vertex f_S can be assessed as:*

$$\nu_{x_p}^{f_S} = C_p + \sum_{f_S' \in \mathcal{N}(x_p)\setminus\{f_S\}} \nu_{f_S'}^{x_p}.$$

Proof. The simple term f_a associated to vertex a is the addition of all the simple terms that have variable a in their scope (see Section 2.2.1). In the case of an activation vertex x_p this simple term corresponds to p's activation term f_p. Recall that, in RB-LBP, an activation term takes on the value of participant's p activation cost for $\mathbf{x_p} = 1$ and zero otherwise (see Section 4.1.2). Therefore, using Equation A.12 from Lemma 1 we have that

$$\begin{aligned}\nu_{x_p}^{f_S} &= f_p(1) - f_p(0) + \sum_{f' \in \mathcal{N}(x_p)\setminus\{f\}} \nu_{f'}^{x_p} \\ &= C_p - 0 + \sum_{f' \in \mathcal{N}(x_p)\setminus\{f\}} \nu_{f'}^{x_p} \\ &= C_p + \sum_{f_S' \in \mathcal{N}(x_p)\setminus\{f_S\}} \nu_{f_S'}^{x_p}.\end{aligned}$$

□

Theorem 2. *The single-valued message from an option vertex o to a selection vertex f_S can be assessed as:*

$$\nu_o^{f_S} = \nu_{f_E}^o.$$

Proof. We begin from Equation A.12. Since option variables are not in the scope of any simple terms, we can cancel the first two terms in Equation A.12. Moreover, since an option vertex is connected only to a selection vertex f_S and an equality vertex f_E we can simplify the summation. Therefore, we have that

$$\begin{aligned}\nu_o^{f_S} &= f_o(1) - f_o(0) + \sum_{f' \in \mathcal{N}(o) \setminus \{f\}} \nu_{f'}^x \\ &= \sum_{f' \in \mathcal{N}(o) \setminus \{f\}} \nu_{f'}^o \\ &= \nu_{f_E}^o.\end{aligned}$$

□

Theorem 3. *The single-valued message from an option vertex o to an equality vertex f_E can be assessed as:*

$$\nu_o^{f_E} = \nu_{f_S}^o.$$

Proof. Follows the same reasoning as Theorem 2. That is we begin from Equation A.12. Then, we cancel the first two terms from Equation A.12 since option variable o is not in the scope of any simple term. Last, we simplify the summation. Therefore, we have that

$$\begin{aligned}\nu_o^{f_E} &= f_o(1) - f_o(0) + \sum_{f' \in \mathcal{N}(o) \setminus \{f\}} \nu_{f'}^x \\ &= \sum_{f' \in \mathcal{N}(o) \setminus \{f\}} \nu_{f'}^o \\ &= \nu_{f_S}^o.\end{aligned}$$

□

Next, we turn our attention to the more challenging case of deriving efficient messages from composite vertices to simple vertices.

Theorem 4. *The single-valued message from an equality vertex f_E, which is connected to option vertices o and o', to option vertex o is exactly the last message received by the equality vertex from option vertex o'. Formally,*

$$\nu_{f_E}^o = \nu_{o'}^{f_E}.$$

Proof. We begin by taking the difference between the messages sent by equality vertex f_E to option vertex o for the active and the inactive state of option variable o. From there, it is just a matter of applying Equations A.3, A.4, and A.11.

$$\begin{aligned}\nu_{f_E}^o &= \mu_{f_E}^o(1) - \mu_{f_E}^o(0) \\ &= \mu_{o'}^{f_E}(1) - \mu_{o'}^{f_E}(1) \\ &= \nu_{o'}^{f_E}.\end{aligned}$$

□

Theorem 5. *The single-valued message from a selection vertex f_S to its activation vertex a can be assessed as the maximum message received from its option vertices $o_1, \ldots, o_n$. Formally,*

$$\nu^a_{f_S} = \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \nu^{f_S}_o .$$

Proof. We begin by taking the difference between the messages sent by selection vertex f_S to the activation vertex a for its active and the inactive state. Next, we apply Equations A.6 and A.5. Then, we simplify the terms that cancel out. In the final step, we apply Equation A.11.

$$\begin{aligned}
\nu^a_{f_S} &= \mu^a_{f_S}(1) - \mu^a_{f_S}(0) \\
&= \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \big(\mu^{f_S}_o(1) + \sum_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \mu^{f_S}_{o'}(0)\big) - \sum_{o \in \mathcal{N}(f_S) \setminus \{a\}} \mu^{f_S}_o(0) \\
&= \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \big(\mu^{f_S}_o(1) - \mu^{f_S}_o(0)\big) \\
&= \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \nu^{f_S}_o .
\end{aligned}$$

□

Theorem 6. *The single-valued message from a selection term f_S to any of its option terms can be assessed from the message received from the activation vertex a and the best message received from any of the option vertex $(o_1, \ldots, o_n)$. Formally,*

$$\nu^o_{f_S} = \min(\nu^{f_S}_a, -\max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \nu^{f_S}_{o'}).$$

Proof. We begin by taking the difference between the messages sent by a selection term f_S to the option variable o for its active and the inactive state. Next, we apply Equations A.10 and A.7. Then, we expand the expressions $AllZeros$ and $Select(o')$ according to Equations A.8 and A.9. Afterward, we simplify the terms that cancel out. In the final step, we apply Equation A.11.

$$\begin{aligned}
\nu^o_{f_S} &= \mu^o_{f_S}(1) - \mu^o_{f_S}(0) \\
&= \mu^{f_S}_a(1) + \sum_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \mu^{f_S}_{o'}(0) - \max(AllZeros, \max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} Select(o')) \\
&= \mu^S_a(1) \; + \sum_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \mu^S_{o'}(0) \\
&\qquad - \max\Big(\mu^{f_S}_a(0) + \sum_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \mu^{f_S}_{o'}(0), \\
&\qquad\qquad \max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \big(\mu^{f_S}_a(1) + \mu^{f_S}_{o'}(1) + \sum_{o'' \in \mathcal{N}(f_S) \setminus \{a,o,o'\}} \mu^{f_S}_{o''}(0)\big)\Big) \\
&= \min\Big(\; \mu^{f_S}_a(1) - \mu^{f_S}_a(0), - \max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \big(\mu^{f_S}_{o'}(1) - \mu^{f_S}_{o'}(0)\big)\Big) \\
&= \min\Big(\; \nu^{f_S}_a, - \max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \nu^{f_S}_{o'}\Big).
\end{aligned}$$

□

Summarizing, the single-valued messages sent from RB-LBP simple vertices to composite vertices can be assessed by means of the following expressions.

$$\boxed{\begin{aligned} \nu_{x_p}^{f_S} &= C_p + \sum_{f_S' \in \mathcal{N}(x_p) \setminus \{f_S\}} \nu_{f_S'}^{x_p} \\ \nu_{o}^{f_S} &= \nu_{f_E}^{o} \\ \nu_{o}^{f_E} &= \nu_{f_S}^{o} \end{aligned}}$$

Moreover, the single-valued messages sent from RB-LBP composite vertices to simple vertices can be assessed by means of the following expressions.

$$\boxed{\begin{aligned} \nu_{f_E}^{o} &= \nu_{o'}^{f_E} \\ \nu_{f_S}^{a} &= \max_{o \in \mathcal{N}(f_S) \setminus \{a\}} \nu_{o}^{f_S} \\ \nu_{f_S}^{o} &= \min(\nu_{a}^{f_S}, -\max_{o' \in \mathcal{N}(f_S) \setminus \{a,o\}} \nu_{o'}^{f_S}) \end{aligned}}$$

Notice that, in the worst case, we have reduced the number of operations required to assess the message from a composite vertex to a simple vertex from exponential to linear. Moreover, the messages exchanged between vertices contain a single value rather than two values as would be required for binary variables in standard max-sum. Therefore, in this chapter we have provided expressions to assess the max-sum messages exchanged between RB-LBP's vertices in a computationally efficient manner.

Appendix B

CHAINME Efficient Message Computation

We devote this chapter to provide computationally efficient expressions to assess the value of the messages exchanged in CHAINME's local term graph. Therefore, in Section B.1, we review the expressions used in standard max-sum to assess the messages exchanged between the vertices in the local term graph. Then, in Section B.2, we study the particularities of CHAINME local terms. Finally, in Section B.3 we provide computationally efficient expressions to assess the single-valued messages exchanged between CHAINME's local terms.

B.1 Message exchange in standard max-sum

The max-sum algorithm consists of a series of message exchanges between simple and composite vertices in a *local term graph*. Recall from Section 2.2.1 that a simple vertex is a vertex representing a local term whose scope is a single variable, whereas a composite vertex is a vertex representing a local term whose scope is two or more variables. Moreover, the message exchange between vertices is repeated until the algorithm either converges or it is stopped. Furthermore, recall from Section 2.2.2 that, in standard max-sum, the message from a composite vertex f to a simple vertex x is assessed by Equation B.1.

$$\mu_f^x(\mathbf{x}) = \max_{\mathbf{X_f} \setminus \{\mathbf{x}\}} \left\{ f(\mathbf{X}_f) + \sum_{x' \in N(f) \setminus \{x\}} \mu_{x'}^f(\mathbf{x}') \right\} \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{B.1}$$

where $\mathbf{X}_{f \setminus \{x\}}$ can take every possible state of every variable in the scope of local term f except x; f is the local term associated to the composite vertex, and $N(f) \setminus \{x\}$ is the set of neighbors of f excluding x.

On the other hand, the message from a simple vertex x to a composite vertex

f (μ_x^f) is given by Equation B.2.

$$\mu_x^f(\mathbf{x}) = f_x(\mathbf{x}) + \sum_{f' \in N(x) \setminus \{f\}} \mu_{f'}^x(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{D}_x \tag{B.2}$$

where f_x is the simple term associated to x and $N(x) \setminus \{f\}$ is the set of neighbors of x excluding f.

Notice that these messages depend not only on the messages received from neighboring vertices but also on the value taken by the local term they represent. Moreover, the computation of messages from composite vertex to simple vertex takes exponential time. However, as argued in Section 2.2.4 this complexity can be reduced by studying the particularities of each particular local term.

B.2 Message computation

In this section we study the particularities of CHAINME equilibrium terms. Moreover, we provide simplified expressions to assess the max-sum message sent from an equilibrium vertex to an activation vertex for both the active and the inactive state. This simplification is possible since equilibrium terms represent hard constraints which allows us to reduce the number of combinations to take into account when assessing the maximum in Equation B.1.

Following standard max-sum equations, the message sent from the equilibrium vertex of the mediator agent for good g to the activation vertex of participant agent s representing a seller of good g for the state $\mathbf{x_s}$ is assessed as

$$\mu_g^s(\mathbf{x_s}) = \max_{\mathbf{S_{g-s}}, \mathbf{B_g}} \left(m_g(\langle \mathbf{S_{g-s}}, \mathbf{x_s} \rangle, \mathbf{B_g}) + \sum_{p \in \mathcal{S}_{g-s} \cup \mathcal{B}_g} \mu_p^g(\mathbf{x_p}) \right), \tag{B.3}$$

where $\mathcal{S}_{g-s}$ denotes the sellers for good g excluding seller s, and S_{g-s} denotes the variables corresponding to the sellers in $\mathcal{S}_{g-s}$. Similarly, $\mathcal{B}_g$ denotes the buyers for good g and B_g denotes the participant variables of the buyers of good g.

Similarly, the message from the mediator agent for good g to the participant agent b representing one buyer of g for the state $\mathbf{x_b}$ can be assessed as

$$\mu_g^b(\mathbf{x_b}) = \max_{\mathbf{S_g}, \mathbf{B_{g-b}}} \Big(m_g(\mathbf{S_g}, \langle \mathbf{B_{g-b}}, \mathbf{x_b} \rangle) + \sum_{p \in \mathcal{S}_g \cup \mathcal{B}_{g-b}} \mu_p^g(\mathbf{x_p}) \Big), \tag{B.4}$$

where $\mathcal{B}_{g-b}$ denotes the buyers for good g excluding buyer b, and $\mathbf{B_{g-b}}$ denotes the variables corresponding to the sellers in $\mathcal{B}_{g-b}$. Similarly, $\mathcal{S}_g$ denotes the sellers for good g and $\mathbf{S_g}$ denotes the participant variables of the sellers of good g.

In order to provide the simplified version for the assessment of messages sent from a mediator to a participant agent we need first to introduce a series of definitions and lemmas that will be used as building blocks for our proofs.

Definition 1. We define the **value** of an assignment $\langle \mathbf{S_g}, \mathbf{B_g} \rangle$ for good g, w_g, as

$$w_g(\mathbf{S_g}, \mathbf{B_g}) = m_g(\mathbf{S_g}, \mathbf{B_g}) + \sum_{p \in \mathcal{S}_g \cup \mathcal{B}_g} \mu_p^g(\mathbf{x_p}), \tag{B.5}$$

where $\mu_p^g(\mathbf{x_p})$ denotes the value of the last message received by g from neighbor p for state $\mathbf{x_p}$, and $\mathcal{S}_g$ and $\mathcal{B}_g$ are the sellers and buyers of good g.

Definition 2. The optimal value for a good g, w_g^*, is the maximum attainable value for any assignment. That is,

$$w_g^* = \max_{\mathbf{S_g}, \mathbf{B_g}} \big(w_g(\mathbf{S_g}, \mathbf{B_g}) \big). \tag{B.6}$$

Definition 3. Optimal assignment, $\langle \mathbf{S_g^*}, \mathbf{B_g^*} \rangle$, for a good g's sellers and buyers, is an assignment that has the optimal value.

Definition 4. The number of exchanges in the optimal assignment, $k(\mathcal{S}_g, \mathcal{B}_g)$, is the number of seller variables taking value one in the optimal assignment.

Note that, by the definition of the equilibrium term, the number of seller variables taking value one is the same as the number of buyer variables taking value one.

Recall that ν_p^g is the difference between two states, namely $\nu_p^g = \mu_p^g(1) - \mu_p^g(0)$. Let $\mathcal{S}_g^\succ = \langle s_1^\succ, \ldots, s_{|\mathcal{S}_g|}^\succ \rangle$ be a sequence of the sellers for good g ordered decreasingly by the value of the last message sent to good g. Similarly, let $\mathcal{B}_g^\succ = \langle b_1^\succ, \ldots, b_{|\mathcal{B}_g|}^\succ \rangle$ be a sequence of the buyers of good g ordered decreasingly by the value of the last message sent to good g.

Lemma 2. *In the optimal assignment, either $s_1^\succ$ and $b_1^\succ$ are active or all participants are inactive.*

Proof. Let $\langle \mathbf{S_g^*}, \mathbf{B_g^*} \rangle^*$ be an optimal assignment with $k(\mathcal{S}_g, \mathcal{B}_g) > 0$ in which $s_1^\succ = 0$. Making any of the active sellers inactive and making $s_1^\succ$ active instead will increase the value of w_g^* since it will equal to $w_g^* - \nu_{s_i}^g + \nu_{s_1^\succ}^g$, and $\nu_{s_1^\succ}^g > \nu_{s_i}^g$ for any $s_i \neq s_1^\succ$, and we have a contradiction. The same reasoning can be applied to $b_1^\succ$. □

Lemma 3. *The optimal value for good g can be assessed as*

$$w_g^* = \sum_{j=1}^{\eta} \Big(\mu_{s_j}^g(1) + \mu_{b_j}^g(1) \Big) + \sum_{j=\eta+1}^{|\mathcal{S}_g|} \mu_{s_j}^g(0) + \sum_{j=\eta+1}^{|\mathcal{B}_g|} \mu_{b_j}^g(0), \tag{B.7}$$

where $\eta = \max\{j | \nu_{s_j}^g + \nu_{b_j}^g \geq 0\}$, $s_j \in \mathcal{S}_g$, and $b_j \in \mathcal{B}_g$.

Proof. By induction on η.

Base case. Taking $\eta = 0$, in Equation B.7 we get

$$w_g^* = \sum_{j=1}^{r} \mu_{s_j^\succ}^g(0) + \sum_{j=1}^{t} \mu_{b_j^\succ}^g(0). \tag{B.8}$$

Assume there exists another assignment with a value higher than w_g^*. By Lemma 2, such assignment must contain $s_1^\succ$ and $b_1^\succ$. But removing $s_1^\succ$ and $b_1^\succ$ from that assignment will increase the value of the assignment (since $\nu^g_{s_1^\succ} + \nu^g_{b_1^\succ} < 0$ for $\eta = 0$), contradicting the existence of such assignment.

Induction case. Assuming that the lemma holds for $\eta - 1$. Let $\langle \mathbf{S_g^*}, \mathbf{B_g^*} \rangle$ be an optimal assignment with $\eta > 0$. By Lemma 2 we know that $x_{s_1^\succ} = 1$ and $x_{b_1^\succ} = 1$. Taking $\mathcal{S}'_g = \mathcal{S}_g \setminus \{s_1^\succ\}$ and $\mathcal{B}'_g = \mathcal{B}_g \setminus \{b_1^\succ\}$, we have that $k(\mathcal{S}'_g, \mathcal{B}'_g) = k(\mathcal{S}_g, \mathcal{B}_g) - 1 = \eta - 1$ since we have just removed $s_1^\succ$ and $b_1^\succ$.

Hence, by induction, in a scenario without seller $s_1^\succ$ and buyer $b_1^\succ$, the value of the optimal assignment will be:

$$w_g^* = \sum_{j=2}^{\eta} \left(\mu^g_{s_j^\succ}(1) + \mu^g_{b_j^\succ}(1) \right) + \sum_{j=\eta+1}^{|\mathcal{S}_g|} \mu^g_{s_j^\succ}(0) + \sum_{j=\eta+1}^{|\mathcal{B}_g|} \mu^g_{b_j^\succ}(0). \tag{B.9}$$

Adding seller $s_1^\succ$ and buyer $b_1^\succ$ back, we have that

$$\begin{aligned} w_g^* &= \mu^g_{s_1^\succ}(1) + \mu^g_{b_1^\succ}(1) + \sum_{j=2}^{\eta} \left(\mu^g_{s_j^\succ}(1) + \mu^g_{b_j^\succ}(1) \right) + \sum_{j=\eta+1}^{|\mathcal{S}_g|} \mu^g_{s_j^\succ}(0) + \sum_{j=\eta+1}^{|\mathcal{B}_g|} \mu^g_{b_j^\succ}(0) \\ &= \sum_{j=1}^{\eta} \left(\mu^g_{s_j^\succ}(1) + \mu^g_{b_j^\succ}(1) \right) + \sum_{j=\eta+1}^{|\mathcal{S}_g|} \mu^g_{s_j^\succ}(0) + \sum_{j=\eta+1}^{|\mathcal{B}_g|} \mu^g_{b_j^\succ}(0). \end{aligned} \tag{B.10}$$

□

Note that η relates to the number of active buyers and sellers that would take place in the optimal assignment. In other words, $\eta = k(\mathcal{S}_g, \mathcal{B}_g)$. Furthermore, we can define the set of active buyers and the set of active sellers in the optimal assignment as follows.

Definition 5. We define the set of active sellers of good g in the optimal assignment, $\mathcal{S}_g^a$, as those participants that are sellers of good g and active in the optimal assignment. More formally,

$$\mathcal{S}_g^a = \{p_i | p_i \in \mathcal{S}_g, p_i \in \{s_1^\succ, \ldots, s_\eta^\succ\}\}. \tag{B.11}$$

Definition 6. We define the set of active buyers for good g in the optimal assignment, $\mathcal{B}_g^a$, as those participants that are buyers of good g and active in the optimal assignment. More formally,

$$\mathcal{B}_g^a = \{p_i | p_i \in \mathcal{B}_g, p_i \in \{b_1^\succ, \ldots, b_\eta^\succ\}\}. \tag{B.12}$$

At this point we have all the tools we need to provide more efficient expressions to assess the message sent from a mediator agent to a participant agent for both its active and its inactive state. First, in Lemma 4, we provide the

expression to assess the message from mediator g to the participant agent representing a seller s for the inactive state. Then, in Lemma 5, we provide the expression to assess the message for the active state. Finally, in Lemmas 6 and 7, we provide the expression to assess the messages from mediator g to buyer b for its the inactive and inactive respectively.

Lemma 4. *The message from mediator g to one of its sellers $s \in \mathcal{S}_g$ for its inactive state can be assessed as*

$$\mu_g^s(0) = \begin{cases} w_g^* - \mu_s^g(1) + \max(\nu^g_{s^{\eta+1}}, -\nu^g_{b^\eta}), & \text{if } s \in \mathcal{S}_g^a \\ w_g^* - \mu_s^g(0) & \text{otherwise.} \end{cases} \tag{B.13}$$

Proof. We want to obtain the expression for the message from the mediator of good g to its seller s for the inactive state, $\mu_g^s(0)$, from the expression of the optimal value for good g, w_g^*. In order to do so, we will explore two scenarios, one in which the seller is part of the optimal assignment and one in which she is not. For each of the scenarios we will find the differences between the expression for the optimal value (Equation B.6) and the equation used in standard max-sum to assess the message from good g to seller s for the inactive state (Equation B.3). After assessing these differences, proving the lemma will be straightforward.

Assume that s is not part of the optimal assignment ($s \notin \mathcal{S}_g^a$). The expression for the optimal value and the expression for the message from good g to seller s for the inactive state are almost identical. In fact, the only difference is that the latter lacks the last message received by the mediator for good g from seller s for the inactive state. Therefore, if s is not part of the optimal assignment, the message from the mediator for good g to seller s for the inactive state can be assessed by subtracting $\mu_s^g(0)$ from the value of the optimal assignment.

Now, assume that s is part of the optimal assignment ($s \in S_g^a$). In this case, the expression for the optimal value and the expression for the message from the mediator for good g to seller s will differ in two things. First, similarly as happened in the previous case, the expression for the optimal value will contain the last message received from s for the active state (s is active in the optimal assignment). Second, by forcing s to be inactive, we are removing s from the optimal assignment, thus breaking the equilibrium constraint defined by m_g. To mend it, we need to either add a new seller or remove a buyer from the assignment. The best seller we can add to the assignment is be $s^{\succ}_{\eta+1}$. And the best buyer we can remove is be $b^{\succ}_{\eta}$. The expression follows directly from this reasoning. □

Lemma 5. *The message from mediator g to one of its sellers s for its active state can be assessed as*

$$\mu_g^s(1) = \begin{cases} w_g^* - \mu_s^g(1), & s \in \mathcal{S}_g^a \\ w_g^* - \mu_s^g(0) + \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}), & \text{otherwise.} \end{cases} \tag{B.14}$$

Proof. Follows the same reasoning as Lemma 4. We want to obtain the expression for the message from the mediator of good g to its seller s for the active

state, $\mu_g^s(1)$, from the expression of the optimal value for good g, w_g^*. In order to do so, we will explore two scenarios, one in which the seller is part of the optimal assignment and one in which she is not. For each of the scenarios we will find the differences between the expression for the optimal value (Equation B.6) and the equation used in standard max-sum to assess the message from good g to seller s for the inactive state (Equation B.3). After assessing these differences, proving the lemma will be straightforward.

Assume that s is part of the optimal assignment ($s \in \mathcal{S}_g^a$). The expression for the optimal value and the expression for the message from good g to seller s for the active state are almost identical. In fact, the only difference is that the latter lacks the last message received by the mediator for good g from seller s for the active state. Therefore, if s is part of the optimal assignment, the message from the mediator for good g to seller s for the active state can be assessed by subtracting $\mu_s^g(1)$ from the value of the optimal assignment.

Now, assume that s is not part of the optimal assignment ($s \notin S_g^a$). In this case, the expression for the optimal value and the expression for the message from the mediator for good g to seller s will differ in two things. First, similarly as happened in the previous case, the expression for the optimal value will contain the last message received from s for the inactive state (s is inactive in the optimal assignment). Second, by forcing s to be active, we are adding s to the optimal assignment, thus breaking the equilibrium constraint defined by m_g. To mend it, we need to either add a new buyer or remove a seller from the assignment. The best buyer we can add to the assignment is be $b_{\eta+1}^{\succ}$. And the best seller we can remove is be $s_{\eta}^{\succ}$. The expression follows directly from this reasoning. □

Lemma 6. *The message from mediator g to one of its buyers $b \in \mathcal{B}_g$ for its inactive state can be assessed*

$$\mu_g^b(0) = \begin{cases} w_g^* - \mu_b^g(1) + \max(\nu_{b^{\eta+1}}^g, -\nu_{s^\eta}^g), & \text{if } b \in \mathcal{B}_g^a \\ w_g^* - \mu_b^g(0) & \text{otherwise.} \end{cases} \tag{B.15}$$

Proof. The demonstration is identical to that of Lemma 4 exchanging sellers for buyers. □

Lemma 7. *The message from mediator g to one of its buyers $b \in \mathcal{B}_g$ for its active state can be assessed as*

$$\mu_g^b(1) = \begin{cases} w_g^* - \mu_b^g(1), & b \in \mathcal{B}_g^a \\ w_g^* - \mu_b^g(0) + \max(-\nu_{b^\eta}^g, \nu_{s^{\eta+1}}^g), & \text{otherwise.} \end{cases} \tag{B.16}$$

Proof. The demonstration is identical to that of Lemma 5 exchanging sellers for buyers. □

B.3 Message simplification

In this section we provide simplified equations to assess the single-valued messages exchanged between mediator and participant agents in CHAINME in a com-

putationally efficient manner. Moreover, we prove that the expressions provided in Chapter 5 are equivalent to standard max-sum's. We begin by proving in Theorem 7 that the expression used by CHAINME to assess the message sent from a participant agent to a mediator agent (Equation 5.5) is equivalent to standard max-sum's. Then, we prove the equivalence of the expressions to assess the message sent from a mediator agent to a participant agent representing a seller (Equation 5.6) and to a participant agent representing a buyer (Equation 5.7) respectively in Theorems 8 and 9.

Theorem 7. *The single-valued message from participant agent p to the mediator for good g (ν_p^g) can be assessed as the addition of all messages received by p from its neighboring mediators, excluding the message received from g, plus p's activation cost. More formally,*

$$\nu_p^g = C_p + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{g'}^p. \tag{B.17}$$

Proof. The simple term f_p associated to vertex p is the addition of all the simple terms that have variable x_p in their scope (see Section 2.2.1). In the case of an activation vertex x_p, this simple term corresponds to p's activation term. Recall that, in CHAINME, an activation term takes on the value of participant's p activation cost (C_p) for $\mathbf{x_p} = 1$ and zero otherwise. Recall from Lemma 1 that the single-valued message from a simple term to a composite term can be assessed by the following expression:

$$\nu_x^f = f_x(1) - f_x(0) + \sum_{f' \in \mathcal{N}(x) \setminus \{f\}} \nu_{f'}^x. \tag{B.18}$$

Therefore, using Equation B.18, we have that

$$\begin{aligned}
\nu_p^g &= f_p(1) - f_p(0) + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{f'}^x \\
&= C_p - 0 + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{g'}^p \\
&= C_p + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{g'}^p.
\end{aligned}$$

□

Next, in order to provide the efficient expressions for the messages sent from a mediator to a seller and from a mediator to a buyer, we will make use of the bid-ask interval (τ^-, τ^+) from Section 2.1.2. Recall that these values can be assessed as follows:

$$\tau^- = \max(-\nu_{s^\eta}^g, \nu_{b^{\eta+1}}^g) \tag{B.19}$$

$$\tau^+ = \min(-\nu_{s^{\eta+1}}^g, \nu_{b^\eta}^g). \tag{B.20}$$

Theorem 8. *The single-valued message from the mediator of good g to its seller s can be assessed as:*

$$\nu_g^s = \begin{cases} \tau^+, & \textit{if } s \in \mathcal{S}_g^a \\ \tau^-, & \textit{otherwise} \end{cases} \tag{B.21}$$

Proof. We begin by taking the difference between the messages sent by mediator for good g to a seller s for the inactive and the active state (Equations B.13 and B.14 from Lemmas 4 and 5). From there, it is just a matter of expanding both messages and applying the definition of τ^+ and τ^- given by Equations B.19 and B.20.

$$\begin{aligned}
\nu_g^s &= \mu_g^s(1) - \mu_g^s(0) \\
&= \begin{cases} w_g^* - \mu_s^g(1) - \Big(w_g^* - \mu_s^g(1) + \max(\nu^g_{s^{\eta+1}}, -\nu^g_{b^\eta})\Big), & s \in \mathcal{S}_g^a \\ w_g^* - \mu_s^g(0) + \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}) - \Big(w_g^* - \mu_s^g(0)\Big), & \text{otherwise.} \end{cases} \\
&= \begin{cases} -\max(\nu^g_{s^{\eta+1}}, -\nu^g_{b^\eta}), & s \in \mathcal{S}_g^a \\ \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}), & \text{otherwise.} \end{cases} \\
&= \begin{cases} \min(-\nu^g_{s^{\eta+1}}, \nu^g_{b^\eta}), & s \in \mathcal{S}_g^a \\ \max(-\nu^g_{s^\eta}, \nu^g_{b^{\eta+1}}), & \text{otherwise.} \end{cases} \\
&= \begin{cases} \tau^+, & \text{if } s \in \mathcal{S}_g^a \\ \tau^-, & \text{otherwise} \end{cases}
\end{aligned} \tag{B.22}$$

□

Next, we focus on the messages sent from a mediator to the buyers for the good she is mediating.

Theorem 9. *The single-valued message from the mediator of good g to her buyer b can be assessed as:*

$$\nu_g^b = \begin{cases} -\tau^-, & \textit{if } b \in \mathcal{B}_g^a \\ -\tau^+, & \textit{otherwise} \end{cases} \tag{B.23}$$

Proof. Follows the same reasoning as that applied in Theorem 8. We begin by taking the difference between the messages sent by mediator for good g to a buyer b for the inactive and the active state (Equations B.15 and B.16 from Corollaries 6 and 7). From there, it is just a matter of expanding both messages

and applying the definition of τ^+ and τ^- given by Equations B.19 and B.20.

$$\begin{aligned}
\nu_g^b &= \mu_g^b(1) - \mu_g^b(0) \\
&= \begin{cases} w_g^* - \mu_b^g(1) - \Big(w_g^* - \mu_b^g(1) + \max(\nu_{b^{\eta+1}}^g, -\nu_{s^\eta}^g)\Big), & \text{if } b \in \mathcal{B}_g^a \\ w_g^* - \mu_b^g(0) + \max(-\nu_{b^\eta}^g, \nu_{s^{\eta+1}}^g) - \Big(w_g^* - \mu_b^g(0)\Big), & \text{otherwise} \end{cases} \\
&= \begin{cases} -\max(\nu_{b^{\eta+1}}^g, -\nu_{s^\eta}^g), & \text{if } b \in \mathcal{B}_g^a \\ \max(-\nu_{b^\eta}^g, \nu_{s^{\eta+1}}^g), & \text{otherwise} \end{cases} \\
&= \begin{cases} -\max(\nu_{b^{\eta+1}}^g, -\nu_{s^\eta}^g), & \text{if } b \in \mathcal{B}_g^a \\ -\min(\nu_{b^\eta}^g, -\nu_{s^{\eta+1}}^g), & \text{otherwise} \end{cases} \\
&= \begin{cases} -\tau^-, & \text{if } b \in \mathcal{B}_g^a \\ -\tau^+, & \text{otherwise} \end{cases}
\end{aligned} \tag{B.24}$$

□

In this chapter we have provided expressions to assess the max-sum messages exchanged between CHAINME's vertices in a computationally efficient manner. Bellow, we provide a summary of these expressions.

$$\nu_p^g = C_p + \sum_{g' \in \mathcal{N}(p) \setminus \{g\}} \nu_{g'}^p$$

$$\nu_g^s = \begin{cases} \tau^+, & \text{if } s \in \mathcal{S}_g^a \\ \tau^-, & \text{otherwise} \end{cases}$$

$$\nu_g^b = \begin{cases} -\tau^-, & \text{if } b \in \mathcal{B}_g^a \\ -\tau^+, & \text{otherwise} \end{cases}$$

Notice that, in the worst case, we have reduced the number of operations required to assess the message from a composite vertex to a simple vertex from exponential to log-linear (the time required to sort the messages received by the equilibrium term).

Appendix C

Walsh and Wellman's Supply Chain Formation problems

In this chapter collect the SCs described by Walsh and Wellman in [Walsh and Wellman, 2003]. These SCs are relatively small (33 participants at most), therefore they are not good candidates to test the scalability of SCF methods. However, these SCs have been studied in detail by Winsper et al. in [Walsh and Wellman, 2003, Winsper and Chli, 2010, Winsper and Chli, 2013] and represent a good reference point for a more detailed for comparing SCF methods.

Each figure represents a different SC network. Moreover, boxes represent participants, whereas circles represent goods at trade. Take for instance the SC represented in Figure C.1. In this SC there are give participants ($p_1, p_2, p_3, p_4,$ and c_2) and three goods at trade ($g_1, g_2,$ and g_3). Furthermore, arrows represent potential flows of goods. For instance, in Figure C.1, participant p_1 is willing to sell good g_1 to participant p_3. Finally, the label bellow the rightmost participants in each SC represents its activation cost. These costs are calculated in order to ensure the existence of a solution in which a profitable SC configuration exists in 90% of the instances when the activation costs for the rest of the participants are drawn from a uniform distribution U(0,1) [Walsh and Wellman, 2003]. Back in Figure C.1, the activation cost for participant c_2 is set to 1.216.

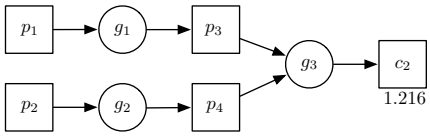


Figure C.1: Supply Chain Formation problem SIMPLE.

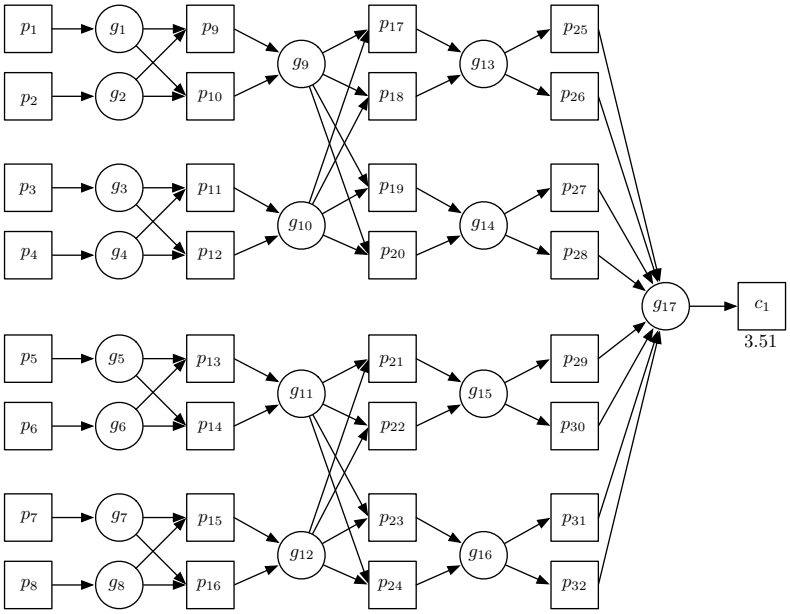


Figure C.2: Supply Chain Formation problem BIGGER.

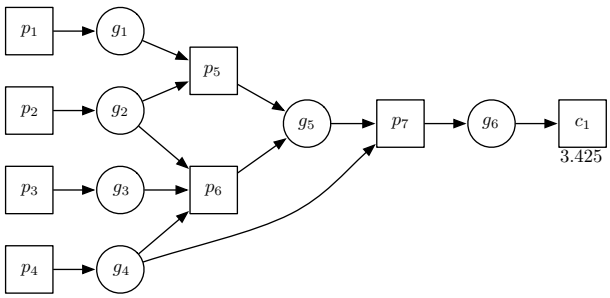


Figure C.3: Supply Chain Formation problem GREEDY-BAD.

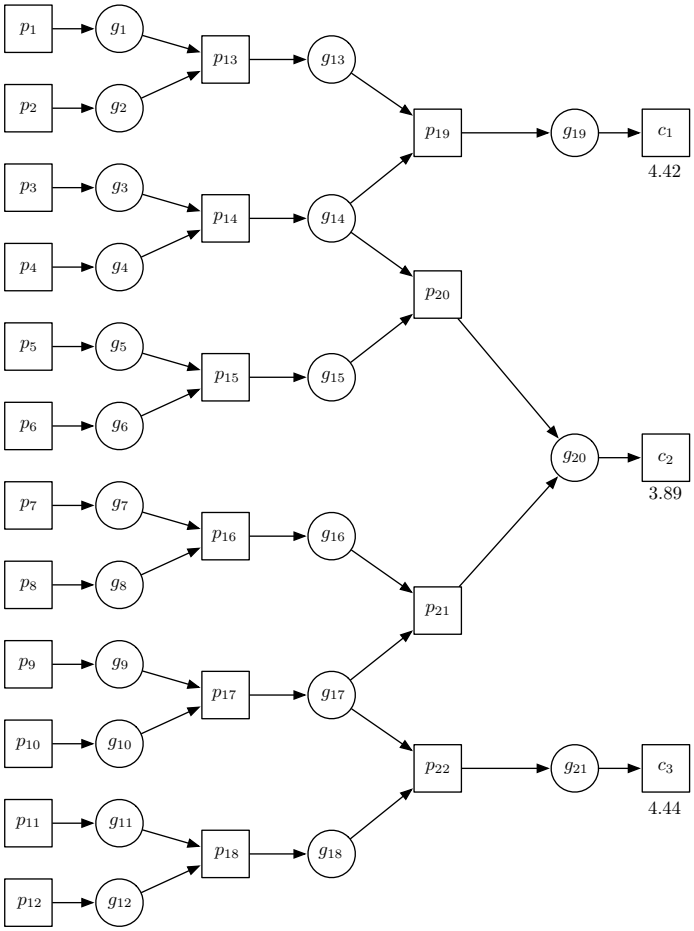


Figure C.4: Supply Chain Formation problem MANY-CONS.

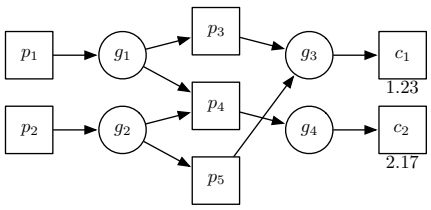


Figure C.5: Supply Chain Formation problem TWO-CONS.

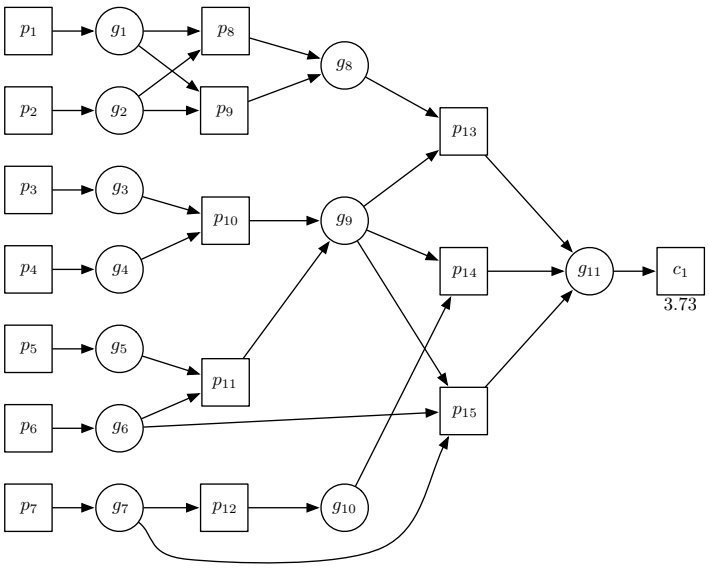


Figure C.6: Supply Chain Formation problem UNBALANCED.

Bibliography

[Aji and McEliece, 2000] Aji, S. M. and McEliece, R. J. (2000). The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343.

[Attiya and Welch, 2004] Attiya, H. and Welch, J. (2004). *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience.

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

[Bishop et al., 2006] Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 1. springer New York.

[Borenstein and Saloner, 2001] Borenstein, S. and Saloner, G. (2001). Economics and electronic commerce. *Journal of Economic Perspectives*, pages 3–12.

[Cerquides et al., 2007] Cerquides, J., Endriss, U., Giovannucci, A., and Rodríguez-Aguilar, J. A. (2007). Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *IJCAI*, pages 1221–1226. Morgan Kaufmann Publishers Inc.

[Collins et al., 2002] Collins, J., Ketter, W., Gini, M., and Mobasher, B. (2002). A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *International Journal of Electronic Commerce*, 7:35–58.

[Cramton et al., 2006] Cramton, P., Shoham, Y., and Steinberg, R. (2006). Combinatorial auctions.

[Davis and Smith, 1983] Davis, R. and Smith, R. G. (1983). Negotiation as a metaphor for distributed problem solving. *Artificial intelligence*, 20(1):63–109.

[Farinelli et al., 2008] Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R. (2008). Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 639–646. International Foundation for Autonomous Agents and Multiagent Systems.

[Fionda, 2009] Fionda, V. (2009). Charting the Tractability Frontier of Mixed Multi-Unit Combinatorial Auctions. *21st International Joint Conference on Artificial Intelligence IJCAI09*, pages 134–139.

[Forney Jr, 1973] Forney Jr, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

[Fudenberg and Tirole, 1991] Fudenberg, D. and Tirole, J. (1991). Game theory, 1991. *Cambridge, Massachusetts*.

[Giovannucci et al., 2008] Giovannucci, A., Vinyals, M., Rodriguez-Aguilar, J. A., and Cerquides, J. (2008). Computationally-efficient winner determination for mixed multi-unit combinatorial auctions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 1071–1078. International Foundation for Autonomous Agents and Multiagent Systems.

[Givoni and Frey, 2009] Givoni, I. E. and Frey, B. J. (2009). A Binary Variable Model for Affinity Propagation. *Neural Computation*, 1600:1589–1600.

[Hayek, 1945] Hayek, F. A. (1945). The use of knowledge in society. *The American economic review*, pages 519–530.

[Kalin, 2000] Kalin, S. (2000). Trading places. *CIO Magazine*, 13(9):96–103.

[Kim et al., 2010] Kim, Y., Krainin, M., and Lesser, V. (2010). Application of max-sum algorithm to radar coordination and scheduling. In *Workshop on Distributed Constraint Reasoning*.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[Mccabe et al., 1990] Mccabe, K. A., Rassenti, S. J., and Smith, V. L. (1990). Auction Institutional Design : Theory and Behavior of Simultaneous Multiple-Unit Generalizations of the Dutch and English Auctions. *The American Economic Review*, 80(5):1276–1283.

[Mikhaylov et al., 2011] Mikhaylov, B., Cerquides, J., and Rodriguez-Aguilar, J. A. (2011). Solving sequential mixed auctions with integer programming. In *Advances in Artificial Intelligence*, pages 42–53. Springer.

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

[Norman et al., 2004] Norman, T. J., Preece, A., Chalmers, S., Jennings, N. R., Luck, M., Dang, V. D., Nguyen, T. D., Deora, V., Shao, J., Gray, W. A., et al. (2004). Agent-based formation of virtual organisations. *Knowledge-based systems*, 17(2):103–111.

[Penya-Alba, 2013] Penya-Alba, T. (2013). From Supply Chain Formation to Multi-agent Coordination. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*.

[Penya-Alba et al., 2012a] Penya-Alba, T., Mikhaylov, B., Pujol, M., Rosell, B., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012a). Tecnologia de subastas para la formacion automatizada de cadenas de suministro. *Novatica*, (218).

[Penya-Alba et al., 2013a] Penya-Alba, T., Mikhaylov, B., Pujol-Gonzalez, M., Esteva, M., Rosell, B., Cerquides, J., Rodriguez-Aguilar, J. A., Sierra, C., Carrascosa, C., Julian, V., Rebollo, M., Rodrigo, M., and Vasirani, M. (2013a). An environment to build and track agent-based business collaborations. In et al., S. O., editor, *Agreement Technologies (Law, Governance and Technology Series)*, pages 609–622. Springer.

[Penya-Alba et al., 2011] Penya-Alba, T., Pujol-Gonzalez, M., Esteva, M., Rosell, B., Cerquides, J., Rodriguez-Aguilar, J. A., Sierra, C., Carrascosa, C., Julian, V., Rebollo, M., Rodrigo, M., and Vasirani, M. (2011). ABC4MAS: Assembling Business Collaborations for MAS. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2011)*, number 3, pages 431–432. IEEE.

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

[Penya-Alba et al., 2012c] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012c). Rb-lbp experiments. `http://iiia.csic.es/~tonipenya/rblbp_experiments.zip`.

[Penya-Alba et al., 2012d] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012d). RB-LBP: Scaling Up Decentralized supply chain formation. In *5th International Workshop on Optimization in Multi-Agent Systems @AAMAS (OPTMAS 2012)*, Valencia.

[Penya-Alba et al., 2012e] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012e). Rb-lbp source code. `http://iiia.csic.es/~tonipenya/rblbp_code.zip`.

[Penya-Alba et al., 2012f] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2012f). Scalable decentralized supply chain formation through binarized belief propagation. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1275–1276, Valencia.

[Penya-Alba et al., 2013b] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2013b). CHAINME: Fast Decentralized Finding of Better Supply Chains. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*.

[Penya-Alba et al., 2013c] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2013c). Social Value Propagation for Supply Chain Formation. In *6th International Workshop on Optimization in Multi-Agent Systems @AAMAS (OPTMAS 2013)*.

[Penya-Alba et al., 2014a] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2014a). Chainme code and experiments. `http://iiia.csic.es/~tonipenya/chainme_code_experiments.zip`.

[Penya-Alba et al., 2014b] Penya-Alba, T., Vinyals, M., Cerquides, J., and Rodriguez-Aguilar, J. A. (2014b). Exploiting Max-Sum for the Decentralized Assembly of High-Valued Supply Chains. In *13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*.

[Pujol-Gonzalez et al., 2013a] Pujol-Gonzalez, M., Cerquides, J., Escalada-Imaz, G., Meseguer, P., and Rodríguez-Aguilar, J. A. (2013a). On binary max-sum and tractable hops. In *European Workshop on Multi-agent Systems (EUMAS)*.

[Pujol-Gonzalez et al., 2013b] Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodríguez-Aguilar, J. A., and Tambe, M. (2013b). Engineering the decentralized coordination of uavs with limited communication range. In *Advances in Artificial Intelligence*, pages 199–208. Springer.

[Pujol-Gonzalez and Penya-Alba, 2014] Pujol-Gonzalez, M. and Penya-Alba, T. (2014). Binary max-sum. `https://binarymaxsum.github.io/`.

[Rogers et al., 2011] Rogers, A., Farinelli, A., Stranders, R., and Jennings, N. R. (2011). Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759.

[Sandholm, 2008] Sandholm, T. W. (2008). Expressiveness in Mechanisms and its Relation to Efficiency: Our Experience from $ 40 Billion of Combinatorial Multi-attribute Auctions, and Recent Theory.

[Stranders, 2009] Stranders, R. (2009). Coordinating teams of mobile sensors for monitoring environmental phenomena.

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

[Tarlow et al., 2011] Tarlow, D., Givoni, I. E., Zemel, R. S., and Frey, B. J. (2011). Graph cuts is a max-product algorithm. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*.

[Vinyals et al., 2010] Vinyals, M., Cerquides, J., Farinelli, A., and Rodríguez-Aguilar, J. A. (2010). Worst-case bounds on the quality of max-product fixed-points. In *NIPS*, pages 2325–2333.

[Vinyals et al., 2008] Vinyals, M., Giovannucci, A., Cerquides, J., Meseguer, P., and Rodríguez-Aguilar, J. A. (2008). A test suite for the evaluation of mixed multi-unit combinatorial auctions. *Journal of Algorithms*, 63(1-3):130–150.

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

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

[Walsh et al., 2000] Walsh, W. E., Wellman, M. P., and Ygge, F. (2000). Combinatorial auctions for supply chain formation. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 260–269. ACM.

[Weiss, 2000] Weiss, Y. (2000). Correctness of Local Probability Propagation in Graphical Models with Loops. *Neural Computation*, 12(1):1–41.

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

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

[Winsper, 2012] Winsper, M. (2012). *Using min-sum loopy belief propagation for decentralised supply chain formation.* PhD thesis, Aston University.

[Winsper and Chli, 2010] Winsper, M. and Chli, M. (2010). Decentralised supply chain formation: A belief propagation-based approach. *Agent-Mediated Electronic Commerce*, page 1.

[Winsper and Chli, 2012] Winsper, M. and Chli, M. (2012). Using the max-sum algorithm for supply chain formation in dynamic multi-unit environments. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1285–1286. International Foundation for Autonomous Agents and Multiagent Systems.

[Winsper and Chli, 2013] Winsper, M. and Chli, M. (2013). Decentralized supply chain formation using max-sum loopy belief propagation. *Computational Intelligence*, 29(2):281–309.

[Witzel and Endriss, 2012] Witzel, A. and Endriss, U. (2012). Time constraints in mixed multi-unit combinatorial auctions. In *Agent-Mediated Electronic Commerce: Designing Trading Strategies and Mechanisms for Electronic Markets*, volume 118 of *Lecture Notes in Business Information Processing*, pages 127–143. Springer-Verlag. Postproceedings of AMEC-2010.

[Wurman et al., 1998] Wurman, P. R., Walsh, W. E., and Wellman, M. P. (1998). Flexible double auctions for electronic commerce : theory and implementation. *Decision Support Systems*, (July):17–27.