Aston University
Birmingham

DOCTOR OF PHILOSOPHY

Using min-sum loopy belief propagation for decentralised supply chain formation

Michael Winsper

2012

Aston University

Using Min-Sum Loopy Belief Propagation for Decentralised Supply Chain Formation

Michael James Winsper

Doctor of Philosophy

Aston University

February 2012

Aston University

Using Min-Sum Loopy Belief Propagation for Decentralised Supply Chain Formation

Michael James Winsper

Doctor of Philosophy, 2012

Thesis Summary

Modern business trends such as agile manufacturing and virtual corporations require high levels of flexibility and responsiveness to consumer demand, and require the ability to quickly and efficiently select trading partners. Automated computational techniques for supply chain formation have the potential to provide significant advantages in terms of speed and efficiency over the traditional manual approach to partner selection. Automated supply chain formation is the process of determining the participants within a supply chain and the terms of the exchanges made between these participants.

In this thesis we present an automated technique for supply chain formation based upon the min-sum loopy belief propagation algorithm (LBP). LBP is a decentralised and distributed message-passing algorithm which allows participants to share their beliefs about the optimal structure of the supply chain based upon their costs, capabilities and requirements.

We propose a novel framework for the application of LBP to the existing state-of-the-art case of the decentralised supply chain formation problem, and extend this framework to allow for application to further novel and established problem cases. Specifically, the contributions made by this thesis are:

- A novel framework to allow for the application of LBP to the decentralised supply chain formation scenario investigated using the current state-of-the-art approach. Our experimental analysis indicates that LBP is able to match or outperform this approach for the vast majority of problem instances tested.

- A new solution goal for supply chain formation in which economically motivated producers aim to maximise their profits by intelligently altering their profit margins. We propose a rational pricing strategy that allows producers to earn significantly greater profits than a comparable LBP-based profit-making approach.
- An LBP-based framework which allows the algorithm to be used to solve supply chain formation problems in which goods are exchanged in multiple units, a first for a fully decentralised technique. As well as multiple-unit exchanges, we also model in this scenario realistic constraints such as factory capacities and input-to-output ratios. LBP continues to be able to match or outperform an extended version of the existing state-of-the-art approach in this scenario.
- Introduction of a dynamic supply chain formation scenario in which participants are able to alter their properties or to enter or leave the process at any time. Our results suggest that LBP is able to deal easily with individual occurences of these alterations and that performance degrades gracefully when they occur in larger numbers.

Keywords: Supply chain formation, min-sum algorithm, loopy belief propagation

Contents

List of Figures

Chapter 1

Introduction

1.1 Supply Chain Formation

A supply chain is a multi-layered network of entities involved in the production, sale and delivery of a product or service to an end consumer. A typical manufacturing supply chain network might be composed of a set of organisations involved in sourcing raw materials and supplying them to manufacturers, several layers of manufacturers who transform these raw materials step-by-step into a finished product, adding value along the way, and a series of distributors and retailers who are involved in delivering the finished product to the consumer. Services and intangible goods, such as software, often undergo a similar process, with features and functionalities being sourced, modified and maintained by a disparate range of providers.

Supply chain formation is "the process of determining the set of participants in a supply chain, the set of goods which are exchanged between the participants, and the terms of these exchanges" [Walsh and Wellman, 2003]. The use of computational techniques for supply chain formation is a relatively new area of research which has generated interest as a means to facilitate trends in business towards outsourcing, virtual corporations and agile manufacturing. Outsourcing involves the contracting out of business functions to external organisations, with the aim of reducing costs and allowing for an increased focus on core competencies. Virtual corporations are "groups of individuals or small businesses which come together temporarily to work together towards a common business goal" [Nachira et al., 2007]. Agile manufacturing is an operations concept characterised by high responsiveness to customer demands, flexibility in the face of rapidly changing market conditions, and an ability to quickly bring products to sale. Each of these three paradigms relies heavily on the ability to efficiently select business partners who are able join to form efficient and mutually beneficial trading relationships.

For businesses wishing to engage in practices such as these, or for organizations simply

wishing to cut costs and increase adaptability, the traditional approach to supply chain formation, with trade relationships manually established, built and strengthened over an extended period of time, is too slow and inflexible to be suitable. A manual approach also brings with it the risk that time constraints and human irrationality may lead to the establishment of inefficient, suboptimal supply chains. This is a problem that could be mitigated or avoided with the use of computational techniques. Indeed, such techniques have already proven their worth in a commercial setting, with combinatorial multi-attribute sourcing auctions having produced over $5 billion [Sandholm, 2008] of real-world savings to businesses.

Computational approaches to supply chain formation generally model potential supply chain participants - businesses capable of forming a link in the supply chain - as individual computational agents with limited information about the structure of the supply chain as a whole. These agents express their capabilities and costs through a mechanism, typically negotiations or auctions, through which the subset of agents capable of forming the most efficient supply chain is determined. At the conclusion of this process, which is typically completed in a fraction of the time required by the manual approach, the supply chain is formed.

Agent-based approaches to supply chain formation may either be centralised or decentralised. Centralised approaches typically make use of combinatorial auctions to determine allocations, with the NP-hard winner determination problem usually being solved with integer programming. The use of integer programming implies complete knowledge by some party about the bids of all agents; this is an assumption which might not always be practical or allowable in the real world. Centralised approaches also introduce a single point of failure into the process. Decentralised approaches to the supply chain formation problem make only minimal assumptions about the participating agents, giving them a wide range of applicability than centralised approaches. However, they are faced with a difficult problem in determining allocations, given that agents only possess local information about the structure of the network and the capabilities of other participants. The state of the art in decentralised techniques, presented in Walsh and Wellman [2003], uses a market-based approach consisting of a series of simultaneous double auctions. While these auctions were shown to frequently be able to produce good allocations over many network structures, results were consistently poor when the cost structures of participants did not allow for the existence of competitive equilibrium[1].

With this in mind, we argue there exists a need for a decentralised approach to the supply chain formation problem which is capable of producing consistently strong allocations over a large variety of network structures while not being unduly affected by the cost structures of par-

[1]Competitive equilibrium in Walsh and Wellman [2003] is a set of producer costs which permits a Pareto optimal state in which agents in the allocation receive non-negative surplus and agents not in the allocation would acquire non-positive surplus by participating in the supply chain.

ticipants. Rather than adopting a market-based approach, we use a technique for probabilistic inference in graphical models known as min-sum loopy belief propagation (LBP) [Pearl, 1988]. The LBP algorithm possesses a number of properties that make it well-suited as a technique for supply chain formation. LBP is decentralised, with participants acting only on the basis of local information. This presents an advantage over other techniques for probabilistic inference in graphical models such as graph cuts or the junction tree algorithm, which require significant modifications to and thus complete knowledge of the structure of the underlying network. By avoiding such modifications, LBP also allows for the preservation of the trading relationships originally present in the graph - an agent passes messages only to the buyers and sellers of its associated goods. Finally, LBP allows agents to assign reserve prices to their goods in a manner no different to the way such values would be held by participants in a one-shot market-based protocol, and to share these values only with relevant participants. These properties mean that the economic self-interest of participating agents is preserved.

In this thesis, we aim to develop a novel technique for the supply chain formation problem based upon the use of the LBP algorithm. Specifically, the objectives of our work are:

- Formulate a framework for the representation of supply chain formation problem instances which permits the use of a graphical inference algorithm such as LBP. This forms part of the contribution explained in detail in Section 1.2.1, and is built upon in the contribution detailed in Section 1.2.3.
- Produce a technique for decentralised supply chain formation using the LBP algorithm which outperforms the current state-of-the-art approach. This also forms part of the contribution laid out in Section 1.2.1.
- Formulate and apply LBP to problem scenarios more realistic than that which is investigated by the existing state-of-the-art in decentralised approaches to supply chain formation, and outpeform comparable techniques in each. These include a scenario in which economically motivated producers aim to maximise their profits, supply chain formation problems where goods are exchanged in multiple units, and a dynamic scenario in which producers are able to change their properties or to leave or enter the process at any time. These scenarios form the basis of the contributions detailed in Sections 1.2.2, 1.2.3 and 1.2.4, respectively.

1.2 Summary of Contributions

This thesis addresses the decentralised supply chain formation problem using min-sum loopy belief propagation. The main contributions of this thesis are the following:

1.2.1 Application of min-sum loopy belief propagation to decentralised supply chain formation

In Chapter 4, we explain our min-sum loopy belief propagation-based approach to the supply chain formation problem. This work was originally published in shorter form in Winsper and Chli [2010]; an extended version is currently in press [Winsper and Chli, 2012d]. The use of min-sum loopy belief propagation is a novel approach to the supply chain formation problem, and represents a significant departure from the established literature, which has focused exclusively on market-based approaches such as auctions and negotiations. Our experiments suggest that LBP is capable of producing more efficient allocations over many of networks we tested than the state-of-the-art decentralised auction based method.

1.2.2 Introduction of dynamic profit-seeking behaviour by supply chain participants

Surplus maximisation is a common goal of many approaches to supply chain formation. While this maximises the utility of the recipients of the goods which are ultimately produced by the supply chain, it also limits the utility of the participants involved in producing those goods. With this in mind, in Chapter 5 we propose a decentralised technique for surplus redistribution by introducing the ability for participants to dynamically adjust the prices of their outputs as the allocation is being determined. We also propose a pricing strategy for producers to use in conjunction with our technique, and show that the use of this strategy allows producers to obtain profits which greatly exceed those produced by a comparable LBP-based profit-making approach. This work forms the basis of the material in Winsper and Chli [2012a].

1.2.3 Application of LBP to a multi-unit case of the supply chain formation problem

The literature on decentralised supply chain formation is relatively limited, and has heretofore only examined scenarios which model simple, single-unit exchanges of goods. This is despite the fact that exchanges of multiple units have been modelled in centralised approaches to the supply chain formation problem for some time, most notably with mixed multi-unit combinatorial auctions [Cerquides et al., 2007]. In Chapter 6 we apply our algorithm to a multi-unit supply chain formation scenario, allowing us to demonstrate its effectiveness in a problem domain more close to that which is examined by state-of-the-art centralised approaches. We compare the performance of our protocol to extended versions of the two auction protocols from the state-of-the-art decentralised approach, and show that LBP generally produces results

equivalent to those of the auction protocols. This work, together with the work described in Section 1.2.4, was published in shorter form in Winsper and Chli [2012b], and also forms part of the basis of Winsper and Chli [2012c].

1.2.4 Investigation of the performance of multi-unit LBP in a supply chain reconfiguration scenario

One of the key advantages of automated computational techniques for supply chain formation is their ability to offer businesses greater responsiveness and adaptability to changing market conditions. Despite this, most computational approaches model supply chain formation as a closed system, with a static list of participants who keep their preferences and valuations constant for the duration of the process. In order to truly reflect the rapidly-changing conditions of real-world markets, in Chapter 7 we propose a scenario in which participants are able to enter and leave the process and to change their preferences and valuations at any point before the final allocation is determined. We investigate the ability of our multi-unit LBP model to deliver efficient allocations in this more realistic scenario. Our results indicate that LBP's performance is generally unaffected by small numbers of these alterations, and that performance degrades gracefully as the number of alterations increases. This work, together with the work described in Section 1.2.3, was published in shorter form in Winsper and Chli [2012b], and also forms part of the basis of Winsper and Chli [2012c].

1.3 Contents of Thesis

The contents of this thesis are as follows:

- In Chapter 2 we introduce supply chain formation and explain why agent-based approaches present a promising approach to this problem. We present details of previous related work in the fields of agent-based supply chain management and supply chain formation, and also include a summary of other work which served to inspire the loopy belief propagation-based approach to supply chain formation presented in this thesis.
- Chapter 3 formalises the fundamental properties underpinning the specific version of the supply chain formation problem we tackle in this thesis, explains the properties of competitive equilibrium and input complemetarities, and explains how we use mixed integer programming to produce optimal benchmark values against which we compare the results produced by our loopy belief propagation-based approach.

- In Chapter 4 we explain the details of how we apply min-sum loopy belief propagation to an instance of the supply chain formation problem identical to that which is investigated using the current state-of-the-art decentralised approach, and compare the results produced by these approaches with those produced using our LBP-based technique.
- In Chapter 5 we present a new version of the supply chain formation problem in which producers aim to maximise their profits. We present a variation of the min-sum loopy belief propagation algorithm, which we refer to as LBP_μ, which attempts to find an approximate solution to this problem by allowing producers to change their profit margins as the algorithm is being run.
- Chapter 6 tests the performance of LBP in a more complex scenario where goods are traded in multiple units. We compare the results of LBP in this scenario to a multi-unit version of the auction protocol we tested against in the previous two chapters.
- Chapter 7 introduces a dynamic supply chain formation scenario in which participants are able to enter and leave the process and to change several of their properties as the algorithm is run. This allows us to test the ability of LBP to respond to unexpected changes to the parameters of the potential supply chain.
- A summary of the thesis and potential directions for future work are presented in Chapter 8.

Chapter 2

Background

Supply chain formation is the process of determining the supply chain participants - producers able to process inputs and manufacture outputs - capable of forming a supply chain which ultimately leads to the production of goods which are sold to one or more consumers according to some predefined social goal.

A software agent is a computer program that acts on behalf of a user. A multi-agent system is a collection of multiple intelligent software agents which interact within an environment. Interactions between agents are usually conducted under the constraints of limited information about the environment and system as a whole. Multi-agent systems allow large global problems to be decomposed into multiple smaller problems which are solved by the agents [Wooldridge, 2009].

Multi-agent systems enable us to model a number of properties characteristic of supply chains, including *uncertainty*, *decentralised decision making* by self-interested agents and the process of *self-organisation* by participants. It is no surprise, then, that application of the agent-based paradigm to several aspects of supply chains has been an ongoing focus of multi-agent systems research for several years, particularly in supply chain management [Pardoe and Stone, 2006; Wellman et al., 2003], most notably the TAC SCM game [Collins et al., 2006], and in the area of supply chain formation [Walsh and Wellman, 1999]. The majority of the literature on supply chain formation uses market-based approaches to elicit costs and capabilities from participants. These market-based approaches can be classified into two broad areas: research which focuses on the use of negotiation-based methods as a means for determining allocations, and studies which model the supply chain as a series of auctions. We explore background work in these two areas in Sections 2.1 and 2.2 respectively. We introduce related work involving the

use of LBP for similar problems in Section 2.4, and assess the feasibility of alternative non-market-based approaches in Section 2.5.

2.1 Negotiations

Distributed negotiation is an approach which is well-suited to the modeling of supply chain formation: each individual procurement and sale decision by each participant in the supply chain can be modelled as a multi-party negotiation, with bids or offers allowing participants to express their capabilities and preferences to potential exchange partners. The Contract Net protocol [Davis and Smith, 1983], a technique for distributed problem-solving based on task decomposition and negotiation, formed the basis of many agent-based models of distributed negotiation. The main focus of the Contract Net is problem distribution, involving discussions between a decentralised network of manager nodes which require tasks to be executed and contractor nodes capable of completing these tasks. These nodes are referred to collectively as the Contract Net. Nodes are not designated with roles, and may act as both managers and contractors simultaneously. Selection of bids by manager nodes is performed in a greedy manner. This myopic approach limits the usefulness of the standard Contract Net protocol for supply chain formation due to a subsequent inability to deal with resource contention.

In more recent literature, a number of other negotiation-based approaches have been proposed and applied to the supply chain formation problem. Wang et al. [2006] use argumentation-based negotiation for decision making in supply chain formation, while Kim and Cho [2010] propose a heuristic-based agent negotiation method for supply chain formation. The results of Kim and Cho [2010] suggest that their negotiation method is capable of producing reliably near-optimal allocations in their scenario, which involves scheduling and manufacturing cost minimisation within a three-tiered supply chain. One common limiting factor present in negotiation-based approaches to supply chain formation, shared by both Wang et al. [2006] and Kim and Cho [2010], is a reliance upon dedicated “mediator” agents in order to facilitate allocations through preference and capability elicitation and aggregation. The use of these mediator agents implies an assumption of centralisation which precludes the application of these methods in areas where this assumption is not valid. Our approach is based upon the sharing of limited information directly between agents, avoiding the need for mediators and ensuring decentralisation.

2.2 Auctions

The other main approach to supply chain formation involves modeling the supply chain as one or more auctions, with first and second-price sealed bid auctions, double auctions and combinatorial auctions among the most frequently-used methods. Supply chain formation through auctions is a popular approach for a number of reasons. Auctions are frequently used in real-world tendering and sales situations [He et al., 2003], they are a very widely-studied approach for agent-mediated e-commerce in general [He et al., 2003; Parsons et al., 2011], they are often able to form adequate solutions to the supply chain formation problem, and some auctions are able to guarantee various desirable game-theoretic properties. Such properties include *incentive compatibility*, which means that a participant's dominant strategy is to truthfully reveal its private information; *individual rationality*, where participants are guaranteed to receive non-negative utility by participating, *budget balance*, where the net flow of money into and out of the mechanism is equal to zero, and *allocative efficiency*, where the utility of participants is maximised. It is important to note, however, that it is impossible [Myerson and Satterthwaite, 1983] for a two-sided mechanism to satisfy each of these four properties simultaneously. Although auctions are a common area in which these properties are studied, they may be present in any mechanism. The game-theoretic properties of our approach are evaluated for each respective scenario it is applied to in Chapters 5, 6, 7 and 8. We introduce these concepts in this section because game-theoretic analysis tends to be more prominent in supply chain formation models which use auctions.

Perhaps the most comprehensive series of studies on supply chain formation using auctions comes from Walsh, Wellman and Babaioff, who examine the efficiency of supply chains formed using simultaneous double auctions [Walsh and Wellman, 2003], one-shot double auctions [Babaioff and Walsh, 2003] and combinatorial auctions [Walsh et al., 2000]. These approaches are evaluated in more detail in Sections 2.2.1 and 2.2.2 for double auctions and combinatorial auctions respectively.

2.2.1 Double Auctions

In Walsh and Wellman [2003], the authors characterise their supply chain formation model in terms of several defining features. First and foremost is the feature of *hierarchical task decomposition*. In their model, supply chain participants are specialised and are only capable of completing specific tasks (i.e. producing a certain type of good). In order to complete their task, they are often reliant on the completion of subtasks (the

production of their input goods) by producers upstream in the supply chain. These producers, in turn, rely on the production of their inputs from producers further upstream, and so on, forming a hierarchy of subtasks.

The authors introduce *resource contention* as another key feature of their model. Multiple participants may rely on common resources, such as goods produced upstream in the supply chain. If these resources are scarce, then these participants may be unable to participate simultaneously in the supply chain. This serves to constrain the number of possible solutions for a given set of participants. Resource scarcity also leads to the *exposure problem*, a situation in which participants acquire an incomplete set of their input goods and are thus unable to produce their output good.

Walsh and Wellman [2003] proposes a market protocol with bidding restrictions referred to as SAMP-SB. The SAMP-SB mechanism comprises a set of auctions, one for each good. These auctions are run simultaneously, asynchronously and independently, without direct communication. SAMP-SB remains the current state of the art in decentralised approaches, and so forms the main basis of comparison for the work presented in this thesis. SAMP-SB was shown to be capable of producing highly-valued allocations - solutions which maximise the difference between the costs of participating producers and the values received by participating consumers - over several network structures, although it frequently struggled on networks where competitive equilibria did not exist. The authors also propose a similar protocol, SAMP-SB-D, with the provision for decommitment in order to remedy the inefficiencies caused by "dead ends", solutions in which one or more producers acquire an incomplete set of complementary input goods and are unable to produce their output good, leading to negative utility. This use of a post-allocation decommitment stage was recognised as an imperfect approach, however, due to the possible problems created by rendering the results of auctions as non-binding. Despite their age, SAMP-SB and SAMP-SB-D represent the current state of the art in decentralised approaches to supply chain formation, and form the main basis of comparison for our work.

Babaioff and Walsh [2003] propose a one-shot double auction mechanism, referred to as Trade Reduction auctions, based upon existing work in Babaioff and Nisan [2001], that sacrifices perfect allocative efficiency, in order to guarantee incentive compatibility, individual rationality and budget balance. The authors propose both a centralised and a distributed algorithm for determining allocations; however, their distributed algorithm relies on the use of mediators for each good, communication between these mediators, and a central coordinator agent. These factors combine to indicate an assumption of

centralisation which, as mentioned earlier, may not always be valid.

2.2.2 Combinatorial Auctions

In Walsh et al. [2000], the authors apply a combinatorial auction protocol to a subset of the networks in Walsh and Wellman [2003] to attempt to find allocations under strategic bidding behaviour by agents. Combinatorial approaches to supply chain formation hold the advantage of being able to avoid the problem of dead ends in the presence of input complementarities by allowing agents to bid for bundles of goods. Due to the strategic bidding behaviours adopted by the agents in Walsh et al. [2000], the results of the combinatorial protocol did not represent a signficant improvement on the double auction protocol, with the quality of the solutions found to be influenced in large part by the amount of available surplus in the networks.

2.2.2.1 Multi-Unit Combinatorial Auctions

Multi-unit combinatorial auctions are a relatively recent [Leyton-Brown et al., 2000] extension to standard combinatorial auctions in which goods may be exchanged in multiple units. A generalisation of this class of auction was applied to the supply chain formation problem in the form of mixed multi-unit combinatorial auctions (MMUCAs) [Cerquides et al., 2007], with the standard combinatorial model of bids being placed for bundles of goods replaced by negotiations over "transformations", essentially commitments by bidders to produce a set of output goods given a set of input goods. There exist several approaches to solving the NP-hard winner determination problem associated with MMUCAs, and the quality of the solutions produced by these techniques tends to depend on the characteristics of the network being tested [Ottens and Endriss, 2008]. Although all existing MMUCA solvers rely on integer programming and thus may eventually face difficulties with scalability, work by Giovannucci et al. [2008] has improved the applicability of MMUCAs to larger supply chain formation problems by proposing an integer program mapping which improves the computational efficiency of the winner determination problem (WDP) calculation by taking advantage of the structural properties of the network. Finding a local, decentralised solver for MMUCAs remains an ongoing area of research.

2.3 Multi-Agent Systems for Supply Chain Management

Although the area of supply chain formation is the focus of this thesis, for completeness we now present a summary of the most well-known applications of multi-agent systems to the related problem of supply chain management. Supply chain management is a distinct problem to supply chain formation because it involves coordination of a supply chain that has already been formed.

The seminal work in this area is Fox et al. [2000], which proposes a series of agent-based coordination strategies in an existing multi-tiered supply chain in which unexpected events occur. The paper provides an illustrative example of a vertically integrated computer manufacturer, with planning, materials, production and dispatching processes at each hypothetical factory being controlled by separate autonomous agents. The authors present a series of simulations demonstrating the effectiveness of a notification system in minimising inventory levels due to disruptions at different tiers within the supply chain. Significant inventory reductions compared to a non-notification approach were found when disruption occured at the final tier of the supply chain, while reductions were less significant with disruptions occured at tiers upstream of the final tier.

The most well-known current and ongoing use of agent-based techniques for supply chain management comes in the form of the Trading Agent Competition Supply Chain Management (TAC SCM) game [Collins et al., 2006]. In TAC SCM, agents represent competing computer manufacturers, who vie to sell computers to customers and maximise their profits whilst operating under limited information in constantly changing market conditions. This involves purchasing components from suppliers as cheaply as possible while still ensuring that commitments are fulfilled, optimising production scheduling and minimising inventory costs.

The most successful agents in the TAC SCM competition to date are TacTex [Pardoe and Stone, 2006] and Deep Maize [Wellman et al., 2005]. TacTex uses a modular approach, with relatively little communication between separate supply manager and demand manager modules. These modules make use of three predictive models - a supplier model, a demand model and an offer acceptance predictor - in order to predict the results of their actions and the state of the market in future steps. Deep Maize focuses on the accurate estimation of marginal values for inputs and outputs in order to break down the larger coordination problem present in TAC SCM into more manageable sub-problems.

2.4 Loopy Belief Propagation

Although auctions and negotiations are by far the most commonly-employed techniques in agent-based approaches to the supply chain formation problem, LBP has been used as a method for task allocation for several years in the related area of agent-based decentralised coordination, particularly for the coordination of sensor networks [Crick and Pfeffer, 2003; Voice et al., 2010].

LBP is a decentralised and distributed approximate inference scheme involving the application of Pearl's belief propagation algorithm [Pearl, 1988] to graphical models containing cycles. We explain the concept of graphical models in Section 2.4.1. LBP uses iterative stages of message passing as a means for estimating the marginal probabilities of nodes being in given states: at each iteration, each node in the graph sends a message to each of its neighbors giving an estimation of the sender's beliefs about the likelihoods of the recipient being in each of its possible states. Nodes then update their beliefs about their own states based upon the content of these messages, and the cycle of message passing and belief update continues until the beliefs of each node become stable.

The most commonly used version of LBP, the sum-product algorithm, is used to estimate marginal probabilities at individual nodes. This is not suitable as an approach for the supply chain formation problem because we are interested in finding the global set of states that produces the most efficient supply chain network, rather than estimating the probability of each node being in the globally optimal solution.

Because the sum-product algorithm is not suitable for solving supply chain formation problem, we use a well-known variant of LBP, the min-sum algorithm, which is used to find the most likely assignment - the maximum a posteriori (MAP) assignment - of the graph as a whole given the probabilities encoded at each node. We provide further details of states in Section 4.3 and our application of the min-sum algorithm to supply chain formation in Section 4.6.

Min-sum LBP presents a promising solution technique for supply chain formation for a number of reasons. Replacing the process of bidding in auctions with message passing between agents allows participants to share their beliefs about the optimal structure of the supply chain without revealing any more private cost information than they would in an open auction. The LBP algorithm is decentralised and distributed, properties important for the realistic representation of individual self-interested business entities. LBP also guarantees optimality for problem instances possessing certain structural properties, while still being able to produce good results for instances without these properties. We

provide more detail about the properties and performance guarantees of LBP in Section 2.4.2.

2.4.1 Graphical Models

Both standard and min-sum LBP require that problems are formulated as graphical models. Graphical models are a means for encoding probability distributions over a set of variables using graphs [MacKay, 2003]. Graphical models may be directed or undirected. Directed graphical models are also known as Bayesian Networks (BNs), while undirected graphical models are also known as Markov random fields (MRFs).

In a directed graphical models, an edge from node i to node j indicates that i causes j. Specifically, the strength of this causal relationship is specified in the conditional probability table at each node - $p(x_j|x_i)$ represents the probability of j being in state x_j given that its parent i is in state x_i.

Undirected graphical models - Markov Random Fields (MRFs) - are useful for representing symmetric dependencies between variables. In MRFs, as in BNs, adjacency (through an undirected edge) of nodes indicates dependence, while nodes which are not directly connected are strictly independent.

Graphical models are suitable as a means for encoding instances of the supply chain formation problem because they allow for the representation of the characteristic features of supply chain formation proposed in Walsh and Wellman [2003] - hierarchical subtask decomposition can be represented explicitly in BNs (through the direction of edges between nodes) or implicitly in MRFs (if we assume that nodes consume goods from linked nodes to their left, and supply goods to linked nodes on their right), while resource contention can be represented implicitly through edges.

2.4.2 Properties

While LBP is known to converge to exact results in a finite number of iterations on tree-structured graphs, there is no such guarantee for more loopy graphs, and if convergence is reached, the solution will be an approximation, unless the graph contains only a single loop [Weiss, 2000]. Recent work [Vinyals et al., 2010] has established worst-case bounds on the quality of solutions produced by min-sum LBP, although these guarantees hold only when all unary and pairwise potentials are non-negative, which is not the case in our model. Despite these limitations, LBP has seen great success in a number of areas,

including Turbo Codes [McEliece et al., 1998] and Low Density Parity Check codes [Frey and MacKay, 1998], stereo vision [Felzenszwalb and Huttenlocher, 2004], as well as in the related field of communication in sensor networks [Crick and Pfeffer, 2003; Farinelli et al., 2008].

2.5 Feasibility of Alternative Non-Market-Based Approaches

There are, of course, other potential approaches to the supply chain formation problem, but, for various reasons, many of these are not practical.

Fully centralised techniques such as mixed integer programming and global search are capable of finding optimal allocations in fractions of a second; however, the complete global knowledge required by these methods limits their practicality for most supply chain formation applications.

Myopic techniques such as greedy search may generate optimal solutions on networks without resource contention, such as the Simple network, an example of which is shown in Figure 3.1. However, their inability to look ahead means they are unsuitable for networks where a decision to allocate a scarce good to a certain producer, such as good 4 being allocated to producer P6 in the Greedy-Bad network (see Figure 2.1), may lead to infeasible solutions.

Figure 2.1: Greedy-Bad network, from Walsh and Wellman [2003]

As mentioned in Chapter 1, exact graphical inference techniques such as graph cuts and junction trees are also possible alternatives; however, the need for complete knowledge of the underlying network obviates the usefulness of these techniques to decentralised applications, and such modifications, whether they involve clustering nodes or cutting

edges, serve to destroy the trading relationships (and thus the logical paths for information flows) originally present in the network. LBP is able to deal with resource contention whilst preserving the original structure of the network, and operates in a distributed and decentralised manner.

2.6 Summary

Research into supply chain formation has been an ongoing focus of multi-agent systems research for some time. Market-based approaches, such as negotiations or auctions, are the most common techniques used in the literature, but these approaches have their drawbacks. Centralised methods, such as most negotiation-based methods, or combinatorial auctions, face problems with scalability, while decentralised techniques such as double auctions are prone to the exposure problem and may face problems in producing good solutions. Min-sum loopy belief propagation, a non-market-based approach, has been used to solve similar problems, and is used as the basis for our work because it represents a promising potential technique for decentralised supply chain formation for several reasons - it is a distributed and decentralised algorithm, it requires only minimal information-sharing, and it possesses performance guarantees for certain problem instances.

The following chapter presents the core properties common to each of the specific instances of the supply chain formation problem examined in this thesis. We also provide details of how we determine optimal benchmarks, through the use of mixed integer linear programming, to provide a point of comparison for the performance of LBP.

Chapter 3

Problem Formulation

Supply chain formation is the process of determining the set of participants within a supply chain and the terms of exchanges between these participants. The ultimate goal of any supply chain is the production and sale of goods to end consumers. However, the production of these final goods depends upon the completion of a series of subtasks by participants further upstream in the supply chain. In most cases, before a producer can produce any goods, it must acquire a set of input goods from its suppliers. In turn, these suppliers must have acquired their sets of input goods from their suppliers, and so on. Because of this characteristic feature, supply chains lend themselves well to representation in the form of task dependency networks. This representation, first proposed in Walsh and Wellman [2003], forms the basis of our supply chain network representation.

3.1 Task Dependency Networks

Our task dependency networks take the form of bipartite directed acyclic graphs. An example of this representation is given in Figure 3.1. There are two types of node: individual producers and consumers, which are represented by rectangles in our network diagrams, and goods represented by circles. Directed edges indicate potential flows of goods. An edge leading from a producer to a good indicates that the producer is capable of producing the good, while an edge leading from a good to a producer or consumer means that the producer or consumer is able to consume the good. Consumers, as their name suggests, cannot produce goods.

This representation allows for the clear statement of network structures while retaining fidelity to the structure of real-world supply chains. For example, in Figure **??**, we see

Figure 3.1: A sample supply chain network - Simple - from Walsh and Wellman [2003]. Producers (P1, P2, P3, P4) and consumers (C1) are represented by rectangles, while goods are represented by circles. Edges between vertices indicate potential flows of goods. Numbers below producers represent production costs, while numbers below consumers indicate consumption values.

that producer P1 is able to produce good 1 at a price of 0.36, which producer P3 needs to consume in order to produce good 3, at a price of 0.53 plus the cost of acquiring good 1, for consumer C1. Similarly, producer P2 is able to produce good 2, for possible consumption by producer P4, which is also able to supply consumer C1 with good 3. If both producers P3 and P4 are able to acquire their single input good, C1 must make a choice about which producer to purchase from. Ideally it will choose the producer able to supply the good at the lowest total price, in this example P3, leaving C1 with a final positive consumption value of $1.216 - (0.362 + 0.535) = 0.319$.

In Chapters 4 and 5, we assume that each good represents a single unit of a commodity which is non-divisible, and equivalent in all aspects other than price. In Chapter 6 we explain how we extend this representation to the multi-unit case of supply chain formation by allowing goods to be traded in multiple units, and introduce additional factors including production capacities, input-to-output good ratios and consumer desired goods. Chapter 7 continues to explore the multi-unit case in a supply chain reconfiguration setting.

3.1.1 Agents

Our supply chain networks are made up of multiple interlinked producers aiming to supply goods to one or more consumers. Each producer and consumer is represented by a software agent.

3.1.1.1 Producers

Producers are capable of producing one or more types of output good, and to do so are required to have obtained the necessary quantities of each good in their set of input goods. In this thesis, we assume that producers produce a single type of output good, although our model is readily generalisable to scenarios where producers are able to produce more types of goods. Some producers do not require any input goods; these producers form the initial echelon of the supply chain. In the case of a producer requiring multiple inputs, we refer to the goods as complementary - a producer is unable to produce its output good if it is only able to acquire a subset of its required input goods. Complementarities are explained in greater detail in Section 3.2. Producers assign a reserve price R_p[1] to each unit of their output good, which is a producer-specific constant encoding the cost of producing each unit of the good plus an additional fixed profit margin. For example, P1's reserve price in Figure 3.1 is 0.36. In Chapters 4 and 5, goods are produced, exchanged and consumed in single units. This allows us to present a clear comparison of the performance of our approach to the results produced by the auction protocols originally presented in Walsh and Wellman [2003]. In Chapters 6 and 7, we examine scenarios where goods are produced, exchanged and consumed in multiple units.

3.1.1.2 Consumers

Consumers aim to obtain goods from their set of consumable goods. In Chapters 4 and 5, where we assume that goods are exchanged in single units, consumers aim to obtain a single unit of each of their consumable goods. In Chapters 6 and 7 we allow consumers to obtain multiple units of their consumable goods. In each network, each consumer is assigned a static consumption value V_c: this is the personal valuation the consumer holds for obtaining one unit of one of its consumable goods. Consumer C1's V_c in Figure 3.1 is 1.216.

3.2 Complementary Goods and Exposure

Complementary goods and the exposure problem are characteristic features of the supply chain formation problem. *Input complementarity* is a situation in which a producer

[1]The reserve prices of our producers are equivalent to the production costs of producers in Walsh and Wellman [2003]. We assume that reported costs include a profit margin in order to provide producers with an incentive to participate in the mechanism.

requires multiple separate goods in order to produce its output good [Walsh and Wellman, 2003]. Input complemetarities, in the presence of resource contention, constrain the number of possible solutions to a supply chain network. An example of input complementarities is for producer P4 in the Two-Cons network, as shown in Figure 3.4, which requires units of both goods 1 and 2 in order to produce its output good. Input complementarities also introduce problem of *exposure* in non-combinatorial approaches to supply chain formation [Walsh and Wellman, 2003]. The exposure problem occurs due to the fact that producers must acquire complementary goods individually. This leads to the risk of the producer acquiring some but not all of their required input goods. Combinatorial approaches avoid this problem by allowing producers to bid on bundles of goods. Our LBP approach limits the exposure problem somewhat by encoding all of each producer's required inputs in each of their active states. The concept of states is explained in greater detail in the following chapter.

3.3 Competitive Equilibrium

Competitive equilibrium, as defined in Walsh and Wellman [2003], is a set of prices assigned to goods in which producers in the optimal allocation obtain non-negative surplus by being active, and producers not in the allocation would acquire non-positive surplus by being active. Additionally, all consumers in the optimal allocation are required to obtain the consumable good which gives them the maximum non-negative surplus, and consumers not in the allocation would receive non-positive surplus by obtaining any good. Figure 3.2 shows an optimal allocation to the Greedy-Bad network with a set of reserve prices that do not permit competitive equilibrium. Active producers and their associated goods are in grey, while inactive producers and goods which are not produced in the allocation are in white. Edges associated with unproduced goods are dashed. For example, in order for P5 to receive its reserve price, the price of good 5 must be greater than the sum of the prices of producer P5's inputs plus P5's reserve price. However, because inactive producer P6 must not be able to make a profit if it was to participate, the price of good 5 must also be lower than the sum of the prices of producer P6's inputs plus its reserve price. In much the same way, good 6 must be exchanged at a price below consumer C1's consumption value, but above P7's reserve price plus the sum of the prices of its inputs. Because the goods cannot be exchanged at prices which fulfil all of the inequalities given the set of agent reserve prices, competitive equilibrium does not exist in this instance of the network.

Figure 3.2: An instance of the Greedy-Bad network with sample prices that do not permit competitive equlibrium

3.4 Networks

Following Walsh and Wellman [2003], we define a set of supply chain network structures exhibiting a variety of structural properties, and use these networks to test the performance of LBP and all comparison techniques. Networks Simple, Two-Cons, Greedy-Bad, Unbalanced, Many Cons and Bigger were originally defined in Walsh and Wellman [2003]. Network Harder was originally defined in Walsh and Wellman [1999], while the Huge network is of our own creation.

The Simple network, shown in Figure 3.3, is a small three-tier network with two possible sources for the supply of C1's consumable good, good 3. Two-Cons, shown in Figure 3.4, introduces the issue of complementary goods - P4 needs both goods 1 and 2 to produce its output. Because of this, only one of the consumers in this network can be satisfied at one time.

The Greedy-Bad network, shown in Figure 3.5, introduces further complementarity issues. Producer P6 is a possible seller of one of Producer P7's input goods, good 5. However, in order to produce good 5, P6 requires good 4, which is also one of P7's inputs. Because P7 is necessarily present in the single optimal solution to this network, it must buy good 5 from P5, even if the price is more expensive than when bought from P6. This network serves to show the weakness of greedy search-based techniques for supply chain formation - although P7 may be able to acquire good 5 more cheaply from P6, in doing so it renders the rest of the supply chain infeasible.

Figure 3.6 shows Unbalanced, a larger network with several instances of complementarity. The Many-Cons network, shown in Figure 3.7 is a larger tree-structured network in which multiple consumers may be satisfied simultaneously. The Bigger network, in Figure 3.8 is a large-scale network with many feasible solutions. Harder, shown in Figure 3.9 can be seen as a much larger version of Greedy-Bad: despite the scale of this network, there exists only one possible solution due to the presence of a number of complementarities. Finally, the Huge network, shown in Figure 3.10, models a very large-scale supply chain, with six tiers of production and three consumers.

Figure 3.4: Two-Cons network, from Walsh and Wellman [2003]

Figure 3.5: Greedy-Bad network, from Walsh and Wellman [2003]

Figure 3.6: Unbalanced network, from Walsh and Wellman [2003]

Figure 3.7: Many Cons network, from Walsh and Wellman [2003]

Figure 3.8: Bigger network, from Walsh and Wellman [2003]

Figure 3.9: Harder network, from Walsh and Wellman [1999]

Figure 3.10: Huge network

3.5 Conversion to MRF form

As mentioned in Section 2.4.1, in order to use LBP we must formulate our problem as a graphical model. To convert the task dependency networks proposed in Walsh and Wellman [2003] into pairwise MRF form, two simple modifications must be made: First, the explicit representation of goods is removed from the network. Where edges previously linked an agent to a good or a good to an agent, edges now link agents directly, though they preserve the notion of an edge between agents meaning a potential route of exchange. Second, we remove direction from the edges in the graph. With the graph converted into pairwise MRF form, we are now in a position to define the states and costs required for the running of LBP.

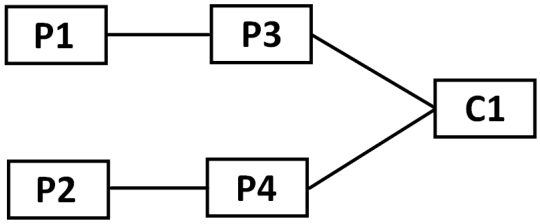

Figure 3.11: The Simple supply chain network converted into MRF form. Edges now link agents directly, and are undirected.

3.6 Creating Benchmark Solutions

In order for us to be able to evaluate the quality of the solutions produced by LBP, we require an optimal benchmark value as a means for comparison. To this end, we use lp_solve [1]to calculate the values of the optimal allocations within our network instances. Each network is formulated as a general MILP model, with production costs, capacities and ratios and consumption values and consumer desired good quantities supplied to the model on a per-run basis. While our MILP models are necessarily network-specific, they are equally applicable to both the single and multi-unit case - if a solution for a single-unit scenario is required, all capacities, ratios and desired good quantities are set to a value of 1.

[1]lp_solve is a free mixed integer linear programming (MILP) solver, available at http://lpsolve.sourceforge.net

3.6.1 Optimal Solutions

We formulate each network as a general mixed integer linear model which encodes the structural properties of the network as a series of linear constraints. Integer and non-integer values representing production costs, producer capacities, producer ratios and consumer desired good quantities and consumption values are supplied to the model on a per-run basis by our Java code. In Appendix 8.1.1 we use the Two Cons network model as an example of how we formulate supply chain networks as linear models.

3.7 Summary

In this Chapter, we presented details of how we formulate the supply chain formation problems studied in this thesis. Following Walsh and Wellman [2003], we use a task dependency network formulation, composed of multiple producer and consumer agents linked within a graph. In the following four Chapters, we demonstrate how we applied LBP to this problem formulation and to three variations of it. Specifically, in Chapter 4 we present details of how we applied LBP to the basic supply chain formation formulation described in this chapter, a scenario also studied in Walsh and Wellman [2003], the current state-of-the-art decentralised approach. In Chapter 5 we propose a novel solution goal for this formulation of the supply chain formation problem, where the aim is to allow self-interested producers to maximise their own profits. Chapter 6 introduces a scenario in which agents exchange goods in multiple units, to our knowledge a first for a fully decentralised approach. Finally, in Chapter 7 we investigate a dynamic supply chain formation scenario in which participants are able to change their properties or to enter or leave the supply chain formation process at any time.

Chapter 4

Supply Chain Formation with Min-Sum Loopy Belief Propagation

In this chapter, we present details of our first contribution: application of the min-sum loopy belief propagation algorithm to an established case of the supply chain formation problem, and experimental analysis that demonstrates that this technique is able to consistently outperform the existing state-of-the-art. This work was originally published in shorter form in Winsper and Chli [2010], and an extended version is currently in press [Winsper and Chli, 2012d]. The specific instance of the supply chain formation problem we consider in this chapter was originally investigated in Walsh and Wellman [2003], and involves a relatively abstract scenario in which goods are traded in single units and producers aim to break even.

We consider this scenario an ideal starting point for demonstrating LBP's performance as a technique for supply chain formation because, despite advances in the complexity of scenarios investigated by centralised approaches such as combinatorial auctions, this instance of the supply chain formation problem represents the most difficult case currently investigated by a decentralised approach. We compare the results produced by min-sum LBP to those produced by our reimplementations of the two decentralised double auction protocols detailed in Walsh and Wellman [2003].

4.1 Background

As discussed in Chapter 2, existing approaches to supply chain formation typically produce solutions through the use of market-based techniques such as negotiations or auc-

tions. Decentralised market-based approaches from the literature have tended to be unable to consistently produce optimal or near-optimal solutions, while centralised market-based approaches require agents to share information with a central mediator, which may not always be possible or preferable. Many potential non-market-based approaches require impractical levels of global knowledge, either in terms of the properties of participants, the structure of the network, or, in many cases, both. We suggest that min-sum loopy belief propagation presents a promising approach to the supply chain formation problem for several reasons - it has been demonstrated to produce good results in application areas similar to the supply chain formation problem. Most notable of these is in the related area of agent-based decentralised coordination, particularly for the coordination of sensor networks [Crick and Pfeffer, 2003; Voice et al., 2010]. Unlike other non-market-based approaches to supply chain formation, LBP operates in a decentralised and distributed manner, and does not require global knowledge of participant properties or network structure.

Min-sum LBP uses iterative stages of message passing as a means for estimating the marginal probabilities, which in our case correspond to the cost to the value of the solution, of nodes being in given states. At each iteration, each node in the graph sends a message to each of its neighbors giving an estimation of the sender's beliefs about the likelihoods of the recipient being in each of its possible states. Nodes then update their beliefs about their own states based upon the content of these messages, and the cycle of message passing and belief update continues until the beliefs of each node become stable. We explain the concept of states in Section 4.3, costs in Section 4.4, and message passing and belief update in Section 4.6. In the following Section, we explain the different levels of knowledge which may be available to the collaborators of participants when using a decentralised approach such as ours.

4.2 Collaborator Knowledge

Decentralised supply chain requires participants to provide some level of information to outside parties, either to independent auctioneers or directly to collaborators, such as buyers of their outputs and sellers of their inputs.

There exist three levels of knowledge a participant may have of its collaborators. The first and most private level is for participants to have no knowledge of the states of their collaborators. This is the level of information available in the SAMP-SB protocol proposed by Walsh and Wellman [2003]. Second, a participant may know that its collaborator has

k states and that states in the set S are compatible with the states in which it collaborates and are incompatible with those in which it does not. Finally, the greatest level of knowledge is for participants to have full knowledge of the states of their collaborators. This is the level of knowledge available using our technique.

4.3 States

Due to the fixed structure of the networks, for each agent there exists a finite number of purchases and sales (if the agent is a producer) in which the agent is viable, i.e. it acquires all its input goods and sells its output good. We encode each of these tuples of exchange relationships as states, with each state defining a list of suppliers and a buyer if the agent is a producer, and a single supplier for consumers. For example, a possible state for producer P3 in Figure 3.1 is "Buy from P1 and sell to C1". The number of states an agent possesses increases with the number of producers able to supply its input good(s), and the number of producers or consumers able to consume its output good. As well as a list of active states, we also allow for the inactive state, where the agent does not acquire or produce any goods.

Because the LBP algorithm requires that agents are aware of the states of their neighbours, we assume that when setting up their states, agents also consider their neighbours states when considering the viability of a particular state. Specifically, an active state is viable if it encodes a complete set of purchases and sales as well as being compatible with at least one of the states of each of the buyers and sellers listed.

For example, if agents do not take into account the states of their neighbours then producer P6 in the Greedy-Bad network, shown in Figure 3.5, has a single active state where it buys goods 2, 3 and 4 from P2, P3 and P4 respectively and sells good 5 to P7. Producer P7 has two active states: one where it buys good 4 from P4, good 5 from P5 and sells good 6 to C1, and another where it buys good 4 from P4 good 5 from P6 and sells good 6 to C1. P6's sole active state is not compatible with either of its listed seller P7's active states.

The first of P7's states is incompatible because P6 is not listed as a seller. P7's second state does list P6 as a seller, but it also lists P4 as a buyer, as does P6's state. Due to the single-unit limitation in this scenario, producer P4 cannot satisfy both P6 and P7. This means that P7's second active state and, therefore, P6's sole active state are non-viable for this network. Agents prune these non-viable states before the algorithm is run.

4.4 Costs

4.4.1 Unary Cost

Each agent associates each of its states with a cost. These values represent the cost to the value of Equation 4.1 were the agent to be assigned with that state in the allocation. For all agents, the cost of being in the inactive state is zero. For producers, all active states incur a positive cost, equal to the reserve price of the producer in question. Consumers assign a negative cost $0 - V_c$ to all states in which they acquire a good, where V_c represents the consumer's consumption value, the value the consumer assigns to the acquisition of its consumable good.

4.4.2 Pairwise Cost

Pairwise costs encode the compatibility of two of the states of a pair of neighboring agents. Two states are compatible if agent i's state lists agent j as a buyer and the list of sellers in j's state includes i and vice versa, or if agent i's state does not list agent j as a buyer and j's state does not list agent i as a seller and vice versa, or if both states are inactive states. If the states are compatible, the pairwise cost is equal to zero. If the two states do not meet any of these conditions, they are incompatible, and the pairwise cost of this combination of states is equal to positive infinity.

4.4.3 Example of States and Costs

To provide an example of our system of costs in practice, we now show the set of states and the unary and pairwise cost values (in Table 4.1) in the Simple network, as shown in Figure 3.1. The Simple network is made up of a set of four producers and a single consumer, as well as three potential goods for production. The possible states of our agents are:

- $P1$: t_1, t_2.
  - $t_1 =$ "Inactive". $t_2 =$ "Sell to P3".
- $P2$: u_1, u_2.
  - $u_1 =$ "Inactive". $u_2 =$ "Sell to P4".
- $P3$: v_1, v_2.

* $v_1 =$ "Inactive". $v_2 =$ "Buy from P1 and sell to C1".

– $P4$: w_1, w_2.

* $w_1 =$ "Inactive". $w_2 =$ "Buy from P2 and sell to C1".

– $C1$: x_1, x_2, x_3.

* $x_1 =$ "Inactive". $x_2 =$ "Buy from P3". $x_3 =$ "Buy from P4".

Producer $P1$ does not require any inputs, and is only capable of selling to one agent - producer $P3$ - meaning its sole active state is t_2, representing the state of not buying any inputs, and selling to $P3$. Consumer $C1$ has two valid active states: buying from $P3$ and selling to no-one - x_2 - and buying from $P4$ and selling to no-one, x_3.

With our list of states complete, we now show the unary costs of the states. Inactive states incur a unary cost of 0, while active states depend upon the type of agent in question. For producers, the unary cost is equal to the reserve price of the producer in question. Consumers incur a unary cost of $0 - V_c$, where V_c is the consumption value of the consumer in question. Thus, our unary costs are as follows:

– P1: $f_{P1}(t_1) = 0$, $f_{P1}(t_2) = 0.362$.

– P2: $f_{P2}(u_1) = 0$, $f_{P2}(u_2) = 0.619$.

– P3: $f_{P3}(v_1) = 0$, $f_{P3}(v_2) = 0.535$.

– P4: $f_{P4}(w_1) = 0$, $f_{P4}(w_2) = 0.854$.

– C1: $f_{C1}(x_1) = 0$, $f_{C1}(x_2) = -1.216$. $f_{C1}(x_3) = -1.216$.

Finally, we show in Table 1 the pairwise costs associated with $P3$ in the Simple network:

Table 4.1: Pairwise costs between $P1$ and $P3$, and $P3$ and $C1$, in the Simple network.

Pairwise Costs

$P1 \leftrightarrow P3$	$P3 \leftrightarrow C1$
$g_{P1P3}(t_1, v_1) = g_{P3P1}(v_1, t_1) = 0$ $g_{P1P3}(t_1, v_2) = g_{P3P1}(v_2, t_1) = \infty$ $g_{P1P3}(t_2, v_2) = g_{P3P1}(v_2, t_2) = 0$	$g_{P3C1}(v_1, x_1) = g_{C1P3}(x_1, v_1) = 0$ $g_{P3C1}(v_1, x_2) = g_{C1P3}(x_2, v_1) = \infty$ $g_{P3C1}(v_1, x_3) = g_{C1P3}(x_3, v_1) = 0$ $g_{P3C1}(v_2, x_2) = g_{C1P3}(x_2, v_2) = 0$ $g_{P3C1}(v_2, x_3) = g_{C1P3}(x_3, v_2) = \infty$

4.5 Cost Function

We allow for two distinct types of cost, denoted as $f_v(x_v)$, the unary cost for agent v of being in state x_v, and $g_{uv}(x_u, x_v)$, the pairwise cost of connected agents u and v being in states x_u and x_v. Our method aims to minimise these costs and thus the cost function given below:

$$\epsilon(x_1, \ldots, x_N) = \sum_{v \in V} f_v(x_v) + \sum_{(u,v) \in E} g_{uv}(x_u, x_v) \tag{4.1}$$

Where $x_1, \ldots, x_N$ is the set of agents, $f_v(x_v)$ is the unary cost of agent v being in state x_v, and $g_{uv}(x_u, x_v)$ is the pairwise cost of linked agents u and v, being labeled with states x_u and x_v. With all else equal, the lower the result of our cost function, the more efficient the allocation. We use the efficiency of the allocation as a measure of the quality of a solution.

The next section introduces the details of min-sum LBP, the technique we employ to minimise our cost function.

4.6 Supply Chain Formation using Min-Sum LBP

Min-sum LBP uses iterative stages of message passing as a means for estimating the marginal probabilities of nodes being in given states. At each iteration, each node in the graph sends a message to each of its neighbors giving an estimation of the sender's beliefs about the likelihoods of the recipient being in each of its possible states. Nodes then update their beliefs about their own states based upon the content of these messages, and the cycle of message passing and belief update continues until the beliefs of each node become stable. To use LBP in a supply chain formation scenario, we convert the task dependency networks described in Chapter 3 into a Markov random field representation suitable for use with LBP, and allow producers and consumers to pass messages between one another encoding their beliefs about the cost of their neighbours' states to the overall efficiency of the supply chain.

The value of an agent's belief about one of its states represents the agent's belief about the cost to the efficiency of the network as a whole were it to be assigned that state, given the content of the messages it has received. Accordingly, LBP begins by initialising the beliefs of each agent about each of their possible states to zero. Each agent then passes a

message containing a vector of belief values to each of its neighbors in the network. Once all agents have passed a message to each of their neighbors, each agent updates its beliefs based upon the content of the messages it received. This process of message passing and belief update continues until the beliefs of our agents about the MAP assignment of the network become stable, at which point we determine the final state of each agent and perform the allocation.

4.6.1 Belief Update

For each of agent u's possible states, we use Equation 4.2 to calculate u's belief in that state. At initialisation, each agent holds a belief of zero about each of its states.

$$bel_u(x_u) = f_u(x_u) + \sum_{w \in N_u} m_{w \to u}(x_u) \tag{4.2}$$

$bel_u(x_u)$ denotes agent u's belief in its state x_u. This belief is made up of two parts: first is the unary cost $f_u(x_u)$ to u incurred by being in state x_u. This is added to the sum of the beliefs about state x_u contained within the messages $m_{w \to u}(x_u)$ received from u's set of neighbors $w \in N_u$.

4.6.2 Messages

At each step of LBP, each agent in the network passes a message to each of its neighbors, consisting of a vector of values representing the sender's beliefs about each of the recipient's states. This involves sender u comparing the compatibility of each individual state x_u from its own set of states with each individual state x_v from recipient v's set of states, taking into account u's belief about its own state x_u, as well as the belief value about state x_u contained within the message passed from v to u in the previous step. Messages can therefore be interpreted as encoding both a compatibility component (through the pairwise cost) and a cost component (through the encoding of cost data in one's current beliefs, if the states are compatible).

$$m_{u \to v}(x_v) = min_{x_u}\Big(bel_u(x_u) - m_{v \to u}(x_u) + g_{uv}(x_u, x_v)\Big) \tag{4.3}$$

Equation 4.3 shows the process of calculating a message to be passed from agent u to agent v. $bel_u(x_u)$ corresponds to agent u's belief in its own state x_u. We subtract from this the belief passed from v to u about state x_u in the previous round of messages,

represented as $m_{v \to u}(x_u)$. Finally, we add the pairwise cost incurred by agents u and v being in states x_u and x_v. We repeat this process for each of agent u's possible states, comparing them in turn to agent v's state x_v. Once the set of possible costs for state x_v dependent on u's set of states have been determined, we take the minimum of these values and add it to the vector of beliefs to be passed from agent u to agent v. This process is repeated for each of v's possible states, resulting in a final vector of values to be passed from u to v.

4.6.3 Example of Min-Sum LBP for Supply Chain Formation

To provide an example of how LBP works in a supply chain formation scenario, we now present a snapshot of the message passing and belief update behaviours in the Simple network example shown in 3.1. Again, we focus on agents P1, P3 and C1. The following subsections show the beliefs of agents P1, P3 and C1 at the step immediately prior to convergence, the ensuing messages that are passed, and the final belief update that leads to a stabilisation of beliefs.

4.6.3.1 Beliefs of P1, P3 and C1 Prior to Convergence

- P1: $bel_{P1}(t_1) = -0.362$, $bel_{P1}(t_2) = -0.319$.
- P3: $bel_{P3}(v_1) = 0.000$, $bel_{P3}(v_2) = -0.319$.
- C1: $bel_{C1}(x_1) = 0.000$, $bel_{C1}(x_2) = 0.257$, $bel_{C1}(x_3) = -0.319$.

At this point, the beliefs of agents P3 and C1 are completely correct, but P1 has an erroneous belief that its inactive state t_1 has less cost to the efficiency of the network than its active state (t_2). This is because the number of iterations that have occurred have not yet been enough for the beliefs of each agent to propagate fully around the network.

4.6.3.2 Messages Passed Between P1, P3 and C1 Prior to Convergence

- P1 to P3: $m_{P1 \to P3}(v_1) = 0.000$, $m_{P1 \to P3}(v_2) = 0.362$.
- P3 to P1: $m_{P3 \to P1}(t_1) = 0.000$, $m_{P3 \to P1}(t_2) = -0.681$.
- P3 to C1: $m_{P3 \to C1}(x_1) = 0.000$, $m_{P3 \to C1}(x_2) = 0.000$, $m_{P3 \to C1}(x_3) = 0.897$.
- C1 to P3: $m_{P3 \to P1}(v_1) = 0.000$, $m_{P3 \to P1}(v_2) = -1.216$.

4.6.3.3 Beliefs of P1, P3 and C1 at Convergence

- P1: $bel_{P1}(t_1) = 0.000$, $bel_{P1}(t_2) = -0.319$.
- P3: $bel_{P3}(v_1) = 0.000$, $bel_{P3}(v_2) = -0.319$.
- C1: $bel_{C1}(x_1) = 0.000$, $bel_{C1}(x_2) = 0.257$, $bel_{C1}(x_3) = -0.319$.

At this point, all agents have correct beliefs - P1 has correctly updated its belief in its inactive state, based upon the content of the message from P3. The beliefs of the agents are that P1 and P3's active states and C2's second active state bring a value to the network of 0.319, and that their inactive states bring a value of 0.000. From this point, another round of message passing and belief update occurs to ensure that beliefs have stabilised, and convergence is then called. The process of convergence is explained in the following Section.

4.7 Convergence

We make use of a convergence detector agent, as recommended in Walsh and Wellman [2003] for scenarios with multiple agents in initially non-quiescent states. In the Walsh and Wellman scenario, non-quiescence is a state in which all auctions have not yet been finalised; in our model, this is essentially equivalent to a state in which beliefs have not yet fully propagated around the network. The convergence detector agent controls termination but is otherwise uninvolved in the workings of the algorithm. The use of such an agent is not unrealistic for supply chain formation settings - in the real world, it is highly likely that supply chain formation facilitation would be provided by a coordinating third party, who would be represented by such an agent, rather than being instituted directly by the businesses themselves. This means that limited communication between participating agents and the third party is a reasonable assumption. Indeed, this assumption of communication with third parties is made by all market-based approaches, whether centralised or decentralised, in the form of bids placed in auctions or negotiation via dedicated mediator agents.

Even in decentralised approaches such as Walsh and Wellman [2003], the market for a single good can ultimately be viewed as centralised, with a single auction responsible for aggregating the bids of multiple participants and deciding the winning bidders. The difference between this approach and a centralised approach is that the solution produced relies upon the result of multiple local auctions, rather than one single global

auction. Thus, the goal in decentralised supply chain formation is not to produce a technique for the niche situation which requires all aspects of the process to be completely decentralised, but to decentralise the most important aspect of the process, which is to compute *global solutions* in a *local manner* on the basis of limited information. Our convergence detector agent allows us to detect convergence more quickly and with less overhead than a distributed method such as the Dijkstra-Scholten algorithm [Dijkstra and Scholten, 1980] for directed cyclic graphs, which involves a similar stage of spanning tree construction. It also plays no part in the computation of solutions and thus does not centralise our approach.

It is also important to note that while the use of a convergence detector agent serves to shorten the running time of the algorithm, the fact that it is not required for the algorithm to produce solutions means that it does not represent a single point of failure. This in contrast with the auctions or mediators present in market based approaches, the failure of which would prevent the technique from producing solutions in most cases.

In the following paragraphs, we explain the operation of our convergence detector agent. We present at the end of this section two potential alternatives to the use of a dedicated convergence detector agent for situations requiring full decentralisation.

Once the LBP algorithm has begun, each agent reports to the convergence detector agent at each iteration specifying whether the state in which they believe holds the lowest cost has changed since the previous iteration. This is the only information shared by agents with the convergence detector - no data about reserve prices, valuations or capabilities is exchanged. If the current number of iterations is greater than the size of the spanning tree (please see Appendix B for an explanation of how we find the spanning tree) and all agents reported that their lowest-cost state has remained the same as the previous iteration, then this indicates that the beliefs of the agents have stabilised, or at least reached the first stage of a stable oscillation. In this case, the convergence detector agent halts the algorithm, and allocation is performed. If such a state is never reached, the algorithm is halted after a pre-defined number of steps and allocation is performed. The process of allocation is outlined in Section 4.8.

As mentioned in Section 2.4.2, LBP is known to converge on tree-structured graphs in a number of iterations equal to the diameter of the graph [Murphy et al., 1997]. Although not all of our networks are trees, we take this value as the earliest number of iterations at which it can be said that LBP has converged. In the absence of an efficient, distributed and fully general technique for finding an exact value for the graph diameter [Magnien et al., 2009], we use distributed depth-first search to find a spanning tree of the graph,

and determine the diameter of the spanning tree using distributed breadth first search to provide an upper bound for the value of the actual diameter of the graph [Magnien et al., 2009]. The full process we follow for this is given in Appendix B.

A random agent present in the original network can perform the function of the convergence detector agent if the use of such an agent is not permissible. The designated agent is not aware of the beliefs of its neighbours about their own states, and thus has no incentive to manipulate the calling of convergence. Once LBP has reached a number of iterations equal to the diameter of the spanning tree, the agent initiates a distributed breadth-first search similar to that used to find the diameter of the spanning tree. This time, agents send messages to their neighbors indicating whether they have reached convergence or not. These messages are propagated back to the agent performing the function of the convergence detector. Once the agent has received a message indicating the convergence status of each node in the network - it is aware of the identities of each agent, though not their costs or capabilities, through the construction of the spanning tree - then it either terminates the algorithm if all agents have converged, or restarts it for a number of steps equal to the diameter of the spanning tree. These additional rounds of message passing increase the bandwidth required by the algorithm but present a better solution in terms of bandwidth and running time in most cases than the final proposed solution, given below.

In situations where neither of the above two techniques are practical, the algorithm can instead be run for a pre-designated number of iterations. This requires that the algorithm is run for a longer period of time than if convergence detection were used, but does not significantly affect the quality of solutions produced.

4.8 Allocation

Before allocation is performed, each agent determines their *final state* - the state, at convergence, in which the agent believes holds the lowest cost. Once the final states of each of the agents have been determined, the process of allocation can begin. For each of the agents in the network, we remove edges leading to other agents which are not listed in their final state if there are no other producers/consumers of that good; in the case of agents being in the inactive state, we remove all of their edges. We then iterate through the agents once more, this time checking to see if, given the results of the previous stage of allocation, each producer was able to acquire all the goods in its set of input goods. If a producer is determined to have acquired an incomplete set, we remove the edges

leading to the buyers of their output good, but any edges leading to sellers of their input goods are untouched. This allows for the presence of "dead ends", a situation where a producer acquires a subset of its required input goods and is unable to sell its output good.

At the conclusion of the allocation process, producers with an edge leading to the buyers of their output good are regarded as having produced a good in the allocation, while consumers with an edge connected to one of the sellers of their consumable good are regarded as having acquired their consumable good in the allocation.

4.8.1 Allocation Value

We determine the value of our allocations using Equation 4.4, given below, where C is the set of consumers to acquire a good, V_c is the consumption value obtained by each of those consumers, P is the set of producers in the allocation who produce a good, and R_p is the production cost of each producer p. This is corresponds to Equation 4.1.

$$Value = \sum_{c \in C} V_c - \sum_{p \in P} R_p \tag{4.4}$$

4.9 Payments

Once the allocation has been performed, each active producer receives a payment equal to their reserve price plus the accumulated cost of their inputs from the buyer of their output good. All active producers therefore recover their total costs of production plus the additional fixed margin encoded in the reserve price. This allows producers to make a profit, motivating participation by economically self-interested producers. Consumers receive a value equal to their consumption value minus the cost of the payments they make to the supplier of their consumable good. In an allocation with no "dead ends", i.e. all producers in the allocation produce their output good, the sum of the differences between the payments made by active consumers and their consumption values is equal to the allocation value.

4.10 Experiments

We test our LBP-based method over the full set of network structures from Walsh and Wellman [2003], one network from Wellman and Walsh [2000], and one additional larger network of our own creation. These networks, which are presented in Section 3.4, exhibit a variety of structural properties which allow us to show the performance of LBP under varying conditions. Further details about the properties of each network are also presented in Section 3.4. Upon initialisation of each of the networks, the reserve price of each producer is set to a decimal value drawn from the interval $U(0, 1)$. These values are re-computed and changed after each run. Consumption values, taken from Walsh and Wellman [2003], are fixed at the values given underneath each consumer (C1, C2 and so on) in each of the networks presented in Section 3.4, over every run.

4.10.1 Performance Evaluation

To evaluate the performance of our method, we perform LBP on each network until a convergent state is reached, using the final state of each agent as the basis for our allocations. If convergence is not reached, i.e. the state which each believes holds the lowest cost continues to oscillate, LBP continues to run for a maximum of 250 steps [1]. The procedure by which we determine convergence is explained in Section 4.7. We record the result at the end of these 250 steps as normal. We compare the value of our allocations to the optimally efficient value, determined using mixed integer programming, and to the results of our re-implementation of the auction protocols given in Walsh and Wellman [2003]: SAMP-SB, and SAMP-SB-D.

SAMP-SB uses a series of double auctions, one per good, which run simultaneously and independently. Winner determination is performed according to the (M+1)st price rule, where buyers win with a bid at or above the (M+1)st price and a sellers win with a bid at or below this price. Buyers and sellers place ascending bids according a simple set of rules, with producers seeking to pay no more for their combined set of input goods than they expect to receive from the sale of their output good. Consumers aim to acquire their single consumable good as cheaply as possible. Allocation is performed as for LBP, as described in Section 4.8, with production costs of active producers taking the place of reserve prices in the allocation value calculation.

[1] We use this value as a reasonable upper limit to the number of iterations to run LBP for on the networks we tested.

SAMP-SB-D is a modification of SAMP-SB which allows inactive producers - those producers who do not produce an output good in the allocation - to decommit from contracts to buy inputs for which they would pay a positive price, a situation referred to as a "dead end". In such a situation, decommitment means that the incoming edges of producers in a dead end are removed, and the production cost of decommiting producer is not counted in the value of the allocation. Producers further down the chain who are affected by this decommitment are, in turn, also allowed to decommit. Once the decommitment stage is completed, the value of the allocation is calculated as outlined in Section 4.8.1. Decommitment allows for the avoidance of this potential source of inefficiency, though at the cost of rendering contracts between bidders non-binding.

As with Walsh and Wellman [2003], we gather 100 results for each network for LBP, SAMP-SB and SAMP-SB-D, discarding runs in which the optimally efficient value is non-positive. Due to this fairly small sample size the results produced by our reimplementations of SAMP-SB and SAMP-SB-D differ slightly to those given in Walsh and Wellman [2003], but in all cases they follow similar trends and thus give a fair representation of the performance of these auction protocols.

4.10.2 Competitive Equilibrium

For fair comparison with SAMP-SB and SAMP-SB-D, we divide our results for networks Unbalanced, Two-Cons, Greedy-Bad and Harder into instances where the sets of reserve prices (for LBP) or production costs (for SAMP-SB) admit competitive equilibrium, and instances where they do not. We generated 100 instances each of competitive equilibrium and non-competitive equilibrium for these networks, determining the presence (or otherwise) of competitive equilibrium using mixed integer programming.

Input complementarities - a situation where a producer has to make a choice between two or more identical goods from two or more different producers - are required for the non-existence of competitive equilibrium; because networks Simple and Many-Cons are polytrees, competitive equilibria always exist for these networks. Although the Bigger and Huge networks do contain input complementarities, we, as with Walsh and Wellman [2003], in the case of Bigger, were unable to generate no-equilibrium instances of these networks. We explain competitive equilibrium in greater detail in Section 3.3.

4.10.3 Efficiency

We divide our results into one of four efficiency classes: negative, zero, suboptimal and optimal. Recall Equation 4.4, which allows us to determine the value of an allocation. The optimally efficient allocation within a network, given a set of producer costs, is the one which maximises this value. We use the optimally efficient allocation as a benchmark for the results we obtain using our LBP method. We determine the optimally efficient allocation for each run using mixed integer programming. We classify our results using the LBP method as follows:

4.10.3.1 Negative

A negative-valued allocation is an allocation in which the reserve prices (for LBP) or production costs (for SAMP-SB) of active producers exceeds the consumption values of active consumer(s). This is caused by dead ends: inactive producers who acquire one or more input goods but do not produce an output, either due to no buyer being found for their potential output good, or due to the producer acquiring an incomplete set of input goods. In LBP, dead ends may be produced by the double-counting of belief values caused by loopy networks, or by non-convergence. SAMP-SB-D avoids the problem of dead ends by allowing producers in such situations to decommit from contracts to buy their inputs. While a similar post-allocation decommitment stage is possible with LBP, we omit this functionality to avoid problems with rendering post-allocation contracts non-binding.

4.10.3.2 Zero

A zero-valued allocation is one in which all producers are assigned to an inactive state, meaning that no goods are bought or sold. Zero-valued allocations are more desirable than negative-valued allocations, but less desirable than suboptimal or optimal allocations.

4.10.3.3 Suboptimal

Suboptimal allocations are allocations in which a positive non-optimal solution was found. This can be caused by the presence of dead ends, or by finding an allocation without dead ends when an allocation of higher value existed.

4.10.3.4 Optimal

An optimal allocation means that our algorithm was able find the allocation which produced the maximum efficiency available, meaning we achieved the same value as the centralised benchmark, determined by local search.

4.11 Results

4.11.1 Efficiency Classes

In keeping with our desire for as fair a comparison between the methods as possible, the efficiency classes of the results produced by SAMP-SB and SAMP-SB-D are identical to those we use for our LBP-based method. For SAMP-SB and SAMP-SB-D, a zero result means that no solution was found, and no dead ends were created. The definitions of negative, suboptimal and optimal allocations given in [Walsh and Wellman, 2003] are identical to ours. The ability for inactive producers to decommit from contracts and thus eliminate the problem of dead ends under the SAMP-SB-D protocol means that there is no negative efficiency category for SAMP-SB-D.

We see from Table 4.2 that LBP is able to match SAMP-SB's performance for networks Simple and Bigger, while producing less efficiency on Huge. The inability for LBP to converge to the optimal on the Huge network is attributable to the large number of undirected cycles present in the network, rather than its size - as we note in Section 2.4.2, LBP is guaranteed to converge to the optimal on trees, regardless of their size. The presence of undirected cycles leads to the double counting of beliefs by nodes within the cycles, which in turn leads these agents to pass incorrect values to the other nodes in the network. This may lead to agents being assigned incorrect states in the final allocation. For all other networks, LBP strongly outperforms SAMP-SB. It is also clear that in most cases, the existence of competitive equilibrium has little effect on the results produced by LBP; as would be expected given our non-market-based approach, the distribution of reserve prices/production costs for producers does not appear to matter as much for LBP as it does for SAMP-SB. Even if we compare our results with the best case for SAMP-SB, using only those results in which competitive equilibria exist, we are still able to show a clear advantage in the proportions of our runs showing optimal efficiency, with marked reductions in negative, zero and suboptimal solutions in almost all cases.

Our results are also comparable to those produced by SAMP-SB-D, with similar efficiency class proportions between the two methods for most networks if only the results

where competitive equilibria exist for SAMP-SB-D are compared. In this case, SAMP-SB-D generates optimal allocations with equal frequency to LBP for networks Unbalanced, Bigger and Greedy-Bad, though like SAMP-SB it struggles when competitive equilibria are not present, with the allocations produced by LBP in the absence of these conditions once again being vastly more efficient.

Table 4.2: Distribution of efficiency classes from LBP, and the SAMP-SB and SAMP-SB-D protocols from Walsh and Wellman [2003]. Classes are Negative, Zero, Suboptimal and Optimal.

	LBP % of instances				SAMP-SB % of instances				SAMP-SB-D % of instances		
Network	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Zero	Sub	Opt
Simple	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	100.0
Unbalanced											
CE	0.0	1.0	0.0	99.0	0.0	0.0	17.0	83.0	0.0	6.0	94.0
No CE	0.0	3.0	0.0	97.0	95.0	0.0	1.0	4.0	92.0	1.0	7.0
Two-Cons											
CE	0.0	0.0	2.0	98.0	18.0	0.0	11.0	71.0	0.0	6.0	94.0
No CE	0.0	0.0	0.0	100.0	25.0	1.0	74.0	0.0	0.0	100.0	0.0
Bigger	2.0	1.0	0.0	97.0	0.0	0.0	2.0	98.0	0.0	2.0	98.0
Many-Cons	0.0	0.0	0.0	100.0	36.0	0.0	54.0	10.0	0.0	0.0	100.0
Greedy-Bad											
CE	0.0	0.0	0.0	100.0	2.0	0.0	19.0	79.0	0.0	0.0	100.0
No CE	0.0	0.0	0.0	100.0	99.0	0.0	1.0	0.0	96.0	0.0	4.0
Harder											
CE	0.0	37.0	0.0	63.0	53.0	0.0	4.0	44.0	34.0	0.0	66.0
No CE	1.0	87.0	0.0	12.0	93.0	0.0	7.0	0.0	69.0	0.0	31.0
Huge	0.0	3.0	96.0	1.0	0.0	0.0	0.0	100.0	0.0	0.0	100.0

While the use of a post-allocation decommitment protocol similar to SAMP-SB-D would have slightly improved the performance of LBP on the Bigger network by converting the two negative-valued allocations into zero-valued allocations, the consistent optimality of our results suggests that such an addition would largely be unnecessary. The performance of our method is aided by the fact that, when viewed as undirected graphs, networks Simple, Greedy-Bad and Many-Cons are all acyclic - LBP is guaranteed to converge to

the correct solution on networks with this structure. The strong performance of LBP on the other networks, however, shows that this network structure is not a prerequisite for allocative efficiency, and that LBP is still able to produce optimal results on more loopy networks.

4.11.2 Average Efficiency

Table 4.3 shows the average efficiency achieved over 100 runs for each network by LBP, SAMP-SB and SAMP-SB-D as a fraction of the available efficiency. An average efficiency of 1.000 indicates that 100% of the available efficiency was captured on each of the hundred runs for that particular network instance, and represents the best possible result. Negative values indicate that over 100 runs the method recorded negative average efficiency in that network. For example, a result showing -1.000 average efficiency means the method achieved, on average, -100% of the maximum available efficiency value. Because we are measuring results as a fraction of the efficient value, strongly negatively efficient results can lead to average efficiency values below -1.000.

Table 4.3: Average efficiency in each network produced by the proposed LBP-based technique, and the SAMP-SB and SAMP-SB-D protocols from Walsh and Wellman [2003]. A result of 1.000 is equal to the capture of an average of 100% of available efficiency, while a result of -1.000 is equal to an average capture of -100% of available efficiency. Note that while 1.000 is the maximum achievable positive value, it is possible to produce negative overall efficiencies below -1.000.

Network	LBP average efficiency	SAMP-SB average efficiency	SAMP-SB-D average efficiency
Simple	1.000	1.000	1.000
Unbalanced			
CE	0.988	0.951	0.996
No CE	0.944	-4.224	0.066
Two-Cons			
CE	0.998	0.719	0.963
No CE	1.000	0.215	0.543
Bigger	0.941	0.998	0.998
Many-Cons	1.000	0.174	1.000
Greedy-Bad			
CE	1.000	0.941	1.000
No CE	1.000	-3.316	0.047
Harder			
CE	0.734	-0.655	0.843
No CE	0.192	-3.260	0.431
Huge	0.646	1.000	1.000

We see from Table 4.3 that once again, LBP essentially equals or significantly outperforms SAMP-SB for the majority of networks, capturing, with the exception of the Bigger and Huge networks, a higher proportion of the efficient value than SAMP-SB is able to. As with the previous set of experiments, if our results are compared to only those where competitive equilibria are present for SAMP-SB-D, then we see that SAMP-SB-D is able to capture 30% more for the Huge network, around 6% more of the average efficiency than LBP for the Bigger network, and around 1% more for Unbalanced, with essentially equally near-optimal or optimal results for the other networks. However, with the exception of the Harder network, LBP tends to strongly outperform SAMP-SB-D in the absence of competitive equilibria, performing near-perfectly on several networks,

such as Greedy-Bad and Unbalanced, where SAMP-SB-D captures less than 10% of available efficiency. Table 4.4 shows that for most networks there exists very little spread within our results.

Table 4.4: Median and interquartile ranges measures for the results of LBP presented in Table 4.3

Network	LBP median	LBP interquartile range
Simple	1.000	0.000
Unbalanced		
CE	1.000	0.000
No CE	1.000	0.000
Two-Cons		
CE	1.000	0.000
No CE	1.000	0.000
Bigger	1.000	0.000
Many-Cons	1.000	0.000
Greedy-Bad		
CE	1.000	0.000
No CE	1.000	0.000
Harder		
CE	1.000	1.000
No CE	0.000	2.211
Huge	0.642	0.274

4.11.3 Messages and Bids Before Convergence

Table 4.5 shows a comparison between the mean averages over 100 runs for each network of the total number of messages passed and the total bandwidth required before convergence in LBP versus the mean average total number of bids placed, and bids placed plus price quotes sent, before quiescence - a state in which no agent wishes to change its bid for any good - in SAMP-SB. We measure the total bandwidth required by LBP by recording the total number of belief values sent between agents in each run. Recording this value allows us to perform a like-for-like comparison with the total bandwidth required by SAMP-SB, as measured by adding the total number of price quotes sent to the

Table 4.5: Average numbers of messages passed before convergence in each network using the proposed LBP-based compared with the average numbers of bids placed in each network before quiescence in the SAMP-SB protocol from Walsh and Wellman [2003].

Network	LBP average number of messages passed	LBP average total bandwidth required	SAMP-SB average number of bids placed	SAMP-SB average number of bids placed and price quotes sent
Simple	46.4	120.78	107.96	411.46
Unbalanced				
CE	367.54	1626.0	615.51	3045.83
No CE	368.0	1621.5	871.93	4649.17
Two-Cons				
CE	90.86	270.3	534.01	2070.68
No CE	84.0	305.32	661.14	2678.82
Bigger	1440.0	17945.28	888.91	6956.41
Many-Cons	399.36	1166.0	2620.12	9153.11
Greedy-Bad				
CE	90.0	284.24	543.32	1934.6
No CE	90.0	292.16	801.15	2995.02
Harder				
CE	11626.48	79547.2	1769.03	16190.65
No CE	12260.32	175237.9	731.06	8091.07
Huge	4548.36	14429.28	3164.4	15412.44

number of bids placed.

We see that, in most cases, LBP requires the passing of far fewer messages to reach convergence than the number of bids needed for SAMP-SB to reach quiescence. One exception to this is the Bigger network - 1440 is the minimum total number of messages that can be passed before LBP can be said to have converged for this network, and is equal to the diameter of the network plus one (an additional iteration is necessary to determine that the states have not changed) multiplied by the number of messages passed in a single step.

We see similar outcomes when comparing the total bandwidth required by LBP with the total number of bids placed plus price quotes sent in SAMP-SB. LBP reliably requires less bandwidth than SAMP-SB on both small networks and large networks with low interconnectedness, such as Many-Cons and Bigger, but tends to require a great deal more bandwidth on large, highly interconnected networks like Harder and the two Many-

Alts networks.

4.11.4 Scaling

In this section, we examine how the efficiency of the allocations produced by LBP, SAMP-SB and SAMP-SB-D are influenced by three network properties: the number of agents in the network, the average degree of connectivity between agents, and the number of tiers of agents in the network.

We see from Figures 4.1, 4.2 and 4.3 that, over the networks we tested, average efficiency in LBP appears to be weakly negatively correlated to the number of agents and the number of tiers, while there is little or no correlation between the average efficiency of LBP and average interconnectedness. SAMP-SB and SAMP-SB-D show no correlation between average efficiency and network structure. It is clear from our results that, as would be expected, LBP performs flawlessly on tree-structured networks, such as Simple or Many-Cons, achieving perfect allocative efficiency. This is a guarantee which holds regardless of the network's size. The performance of LBP on networks with loops cannot, unfortunately, be guaranteed, and a full set of convergence conditions for LBP has yet to be found. This, again, is true regardless of the size of the network, meaning an analysis of the efficiency produced on a set of even larger networks would not be instructive. The absence of a convergence guarantee is the one unavoidable weakness of the algorithm: in much the same as SAMP-SB and SAMP-SB-D are generally unable to deal with the non-existence of competitive equilibrium, LBP may sometimes find difficulties when applied to loopy networks.

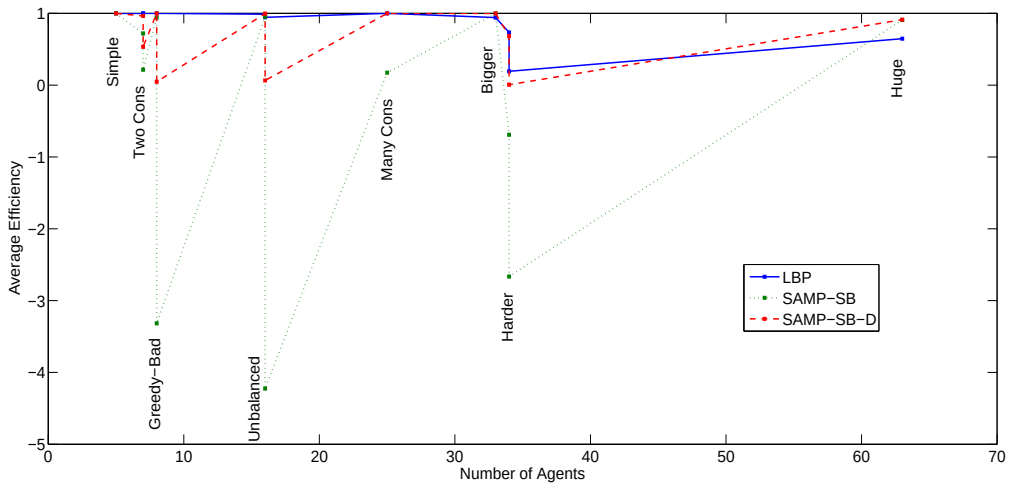

Figure 4.1: A graph showing how efficiency in each of the protocols varies with the number of agents in a network.

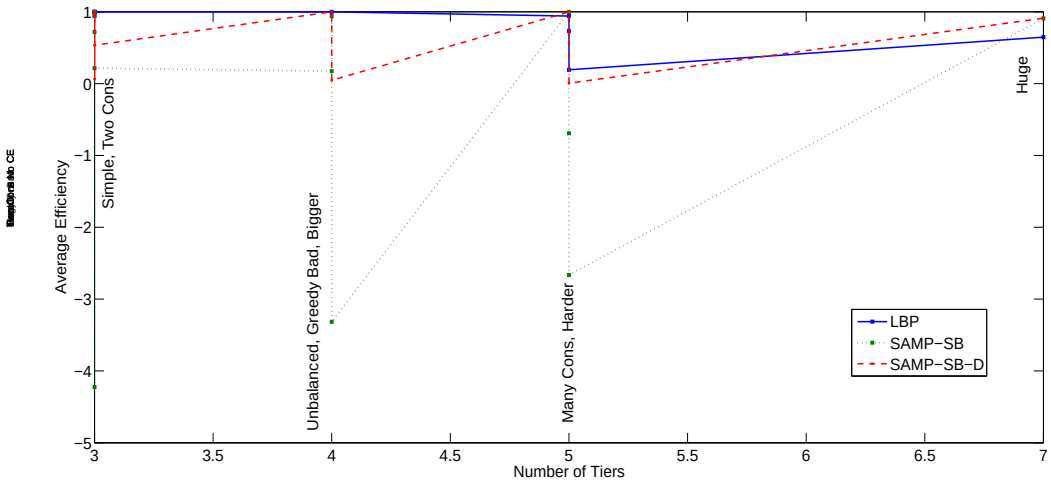

Figure 4.2: A graph showing how efficiency in each of the protocols varies with the number of tiers in a network.

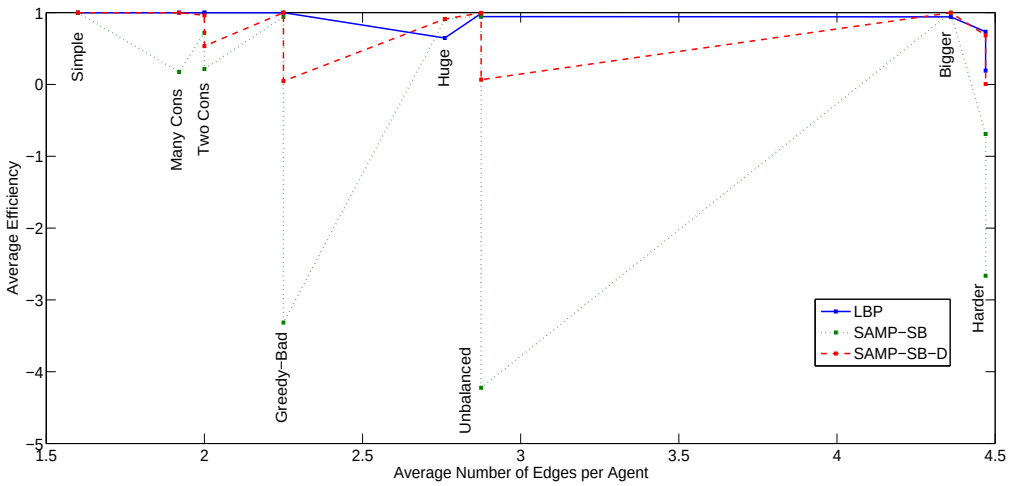

Figure 4.3: A graph showing how efficiency in each of the protocols varies with the average interconnectedness of the agents in a network. We measure average interconnectedness as the total number of edges of each agent in the network divided by the number of agents.

4.11.5 Game-theoretic Properties

As previously mentioned, auction-based approaches are often favored for supply chain formation due to their possession of various game-theoretic properties. In this section, we analyse the game theoretic properties of our LBP-based approach, and compare them to those of SAMP-SB and SAMP-SB-D. Although LBP is not incentive compatible, it is strongly budget balanced and guarantees perfect allocative efficiency when networks are acyclic; individual rationality could also be guaranteed with a post-allocation decommitment stage similar to that used by SAMP-SB-D.

4.11.5.1 Individual Rationality

A mechanism is classified as individually rational if a participant cannot receive negative utility by participating. As with SAMP-SB, we cannot guarantee the individual rationality of our approach given that there exists the possibility of dead ends being present in our allocations. Producers involved in dead ends purchase inputs but are unable to sell their outputs, and so receive negative utility. SAMP-SB-D guarantees individual rationality through its post-allocation decommitment stage, which allows producers involved

in dead ends to decommit from their contracts to buy goods which are no longer needed, and thus avoid negative utility. Individual rationality could be guaranteed in our approach if we were to use a similar process of post-allocation decommitment which is not shown here to avoid the aforementioned problem of rendering post-allocation contracts non-binding.

4.11.5.2 Incentive Compatibility

A mechanism is incentive compatible if the dominant strategy for participants is to truthfully reveal their private valuations. At present, our mechanism is not incentive compatible for either buyers or sellers due to the fact that participants may potentially increase their utility by inflating their reserve prices. However, there is an uncertain upper limit to this potential increase in utility - if a producer reports a reserve price which is too high, there may be, depending on the network structure and the reported production costs of other producers, an alternative, cheaper allocation in which the misreporting agent does not participate. The upper limit is uncertain due to the fact that producers have no information about the structure of the network as a whole, nor do they know the reported costs of any agents other than those which they are directly linked to. This is an issue for sellers in any real-life market-based scenario.

4.11.5.3 Budget Balance

Our approach involves no payments either to or from the mechanism, and is therefore strongly budget balanced. This property is also present in both SAMP-SB and SAMP-SB-D, where no payments are made to or by the auctions.

4.11.5.4 Allocative Efficiency

The results presented in Tables 2 and 3 suggest that our approach is capable of reliably producing more efficient allocations than SAMP-SB and SAMP-SB-D on the network instances tested. LBP guarantees perfect allocative efficiency on acyclic networks, due to its ability to reliably converge to the optimal MAP assignment on graphs which do not contain loops. If LBP converges on a network with a single loop, the resulting allocation is also guaranteed to have perfect allocative efficiency. Because there is no guarantee of the quality of solutions produced by LBP on networks with more than a single loop, allocative efficiency for these networks is also impossible to guarantee. Of the networks

we tested, two - Unbalanced and Bigger - contain multiple loops, and LBP showed strong allocative efficiency for both.

4.12 Conclusions

In this chapter, we presented a new method for decentralised supply chain formation, using work by Walsh and Wellman [2003] as both a foundation for the structure of our networks, and as a basis for comparison to our results. Our LBP-based method, involving decentralised message passing to propagate beliefs held by our agents, is able to perform significantly better at finding efficient allocations for most network than the established approach utilising simultaneous ascending double auctions we compare it to, whilst making no assumptions of centralisation.

For the majority of networks tested, we were able to show that min-sum LBP is able to match or outperform the results obtained by the auction-based method, producing consistently optimal or near-optimal average efficiency results regardless of cost structures. With one exception, our method is able to avoid the problem of consistent suboptimality of allocative efficiency encountered by the auction-based approach when competitive equilibrium is not present, continuing to produce optimal or near-optimal allocations over most networks.

In the following chapter, we introduce a min-sum LBP-based method which allows producers to attempt to make a profit in the LBP mechanism. This technique, which we refer to as LBP_μ, also allows for the property of incentive compatibility for producers.

Chapter 5

Supply Chain Formation with Profit-seeking Producers using LBP$_\mu$

The ultimate goal of many approaches to supply chain formation is surplus maximisation, with a goal of finding a set of producers able to supply goods to a set of consumers which maximises the difference between the total cost of production and the valuations held by consumers for the goods they obtain. In order to achieve this goal, it is assumed that producers compete to sell their outputs at the lowest possible price. A further assumption made frequently in decentralised approaches, such as in Walsh and Wellman [2003], is that producers bid to sell their output goods at a price which allows them to break even and make no profit. Under these assumptions, all of the surplus is enjoyed by consumers, while producers are left with little room for self-interest, or, indeed, any incentive to participate. The work presented in Chapter 4 motivates participation by economically self-interested producers by encoding a fixed profit margin within reserve prices, but the ultimate goal remains that of surplus maximisation.

Although surplus maximisation is by far the most common solution goal, there are some notable instances of work in which alternative goals are pursued.

The Trading Agent Competition Supply Chain Management (TAC SCM) game represents possibly the most well-known instance of the use of agent-based techniques for supply chains in a non-surplus maximisation setting. TAC SCM models a three-tier supply chain, with suppliers of components at the first tier, producers at the second tier and consumers of finished goods at the third tier. The goal of each producer in TAC SCM

is to maximise its own profit by purchasing components from suppliers at the lowest price possible and selling assembled goods to consumers at the greatest price possible. The TAC SCM game is not comparable with our work because it deals with the issue of supply chain management, where supply chains are already formed, rather than supply chain formation.

Babaioff and Nisan [2001] investigates the approximability of a series of social goals when using combinatorial exchanges in a supply chain formation setting. Combinatorial exchanges are a generalisation of both double auctions and standard combinatorial auctions in which bundles of heterogeneous goods are traded between buyers and sellers. As well as the standard surplus maximisation problem, the authors also investigate the approximability of solutions to problems involving volume maximisation - maximising the number of goods traded subject to positive surplus - and social cost minimisation, where the aim is to minimise the difference between the valuations of trading sellers and non-trading buyers. Unfortunately, the work presented in Babaioff and Nisan [2001] is not comparable with our due to the fact that they compute solutions in a centralised manner.

In this chapter, we propose a novel goal for supply chain formation called *profit maximisation*. Profit maximisation eschews the commonly-held assumption that producers are satisfied with making zero or negligible profit by participating, and instead seeks to maximise the amount of available surplus converted into profit by participating producers. Specifically, we aim to maximise the following function:

$$SurplusConverted = \frac{\sum_{p \in P} \mu_p}{\sum_{c \in C*} V_c - \sum_{p \in P*} R_p} \tag{5.1}$$

We divide the sum of the margins of active producers $p \in P$ is divided by the total surplus available in the *optimal* allocation, which is calculated by subtracting the sum of reserve prices R_p of producers in the set of producers in the optimal allocation $P*$ from the sum of consumption values of consumers in the optimal allocation $C*$. In short, the closer the sum of the margins of active producers to the value of the optimal allocation, the more effective the technique is for profit maximisation.

Profit maximisation therefore shifts some of the incentive of participation from consumers to producers, but is still compatible with the assumption that consumers are satisfied as long as they obtain their consumable good at or below the valuation they assign to it. We suggest that this technique is especially applicable in scenarios where a consumer has already agreed to pay a specified price and there is a need for supply chain

participants to intelligently add margins to their outputs. Like surplus maximisation, one of the primary aims of profit maximisation is to find the allocation with the maximum surplus available. By minimising the base costs of participating producers we maximise the available profit.

We propose an LBP-based method for decentralised profit maximisation, referred to as LBP_μ, which allows producers to update their profit margins as the LBP algorithm is being run. The aim of our algorithm is to allow a significant proportion of available surplus to be captured by producers in the form of profit, providing them with a stronger incentive to participate than in approaches which assume price-taking break-even behaviour. Because there exists no other method for profit maximisation in the literature, we test the performance of LBP_μ against an optimal benchmark determined using integer programming, and an LBP-based naive profit-making method.

5.1 LBP_μ Model

In this section, we explain the features of the LBP_μ model which differ from those of our original model, as is described in Chapter 4.

5.1.1 Producers and Consumers

As in standard LBP, producers are individual self-interested business entities capable of consuming and producing goods. Each producer is capable of producing a single type of output good, and in order to do so are required to have obtained a complete set of their input goods. Producers attach a base reserve price R_p to their output good - this is a constant, producer-specific value. In order to try to maximise their profits, producers also add a margin, μ, onto the sale price of their output good. Margins change in value over the course of each run and cannot drop below zero. We give the values we use for R_p and μ in Section 5.3. As with the single-unit case explained in Chapter 4, Consumers aim to acquire a single unit of their consumable good at a price equal to or below their consumption value C_v, which is held static. Allowing consumers to revise their consumption values downwards would lead to allocations with essentially zero surplus available, defeating the purpose of a profit maximisation mechanism, and is therefore not allowed for.

5.1.2 Unary Cost

Like standard LBP, each state of every agent is associated with a cost to the value of the allocation were the agent to be assigned that state. For producers, the unary cost of active states is equal to R_p+μ_p, where R_p and μ_p are the production cost and margin, respectively, of producer p. For consumers, the unary cost of active states is equal to $0 - C_v$ where C_v is consumer c's consumption value. For all agents, the unary cost of inactive states is zero.

5.2 LBP_μ Algorithm

In the previous chapter, we explained how loopy belief propagation uses iterative stages of message passing and belief updates in order to determine the network-wide assignment of states that maximises the value of the allocation. In this section, we describe how we modified the standard min-sum LBP algorithm, through the addition of a margin update step, to allow producers to make a profit.

5.2.1 Margin Update

To allow producers to change the price of their output goods, we add an additional step to the standard min-sum LBP algorithm as is described in Chapter 4, by adding a margin update step. This margin update step occurs at iterations which are multiples of the approximate network diameter, and is conducted after each agent has passed messages and updated their beliefs at that step. The process for finding the approximate network diameter is explained in Section 5.2.5. For example, given an approximate network diameter of 4, producers will update their margins at steps 4, 8, 12 and so on. We implement this restriction, rather than allowing participants to update their margins at every step, to allow enough iterations for the changes in beliefs brought about as a result of these margin updates to propagate throughout the network. Because, as explained below, the nature of LBP_μ means that producers are aware of the maximum surplus available to them, we impose an upper bound δ on the amount each producer may change its margin by at any given margin step. This upper bound is necessary to avoid situations where multiple producers simultaneously attempt to capture all of the available surplus for themselves, increasing the cumulative price of goods in the supply chain to such an extent that no potential solutions exist.

In the following subsections, we present a self-interested strategy that agents participating in the LBP_μ mechanism may choose to follow.

5.2.2 Margin Update Strategy

Assuming the beliefs of producer p are correct, the amount of surplus available to p is equal to p's belief in its inactive state bel_{p_I} minus p's belief in its lowest cost active state bel_{p_A}. As well as the belief values of each agent, our proposed margin update strategy relies on three values: μ_p, δ_p and β_p. μ_p is producer p's current margin. δ_p is the maximum amount by which a producer is permitted to increase or decrease its margin at each margin update step. Finally, β_p is a value used as a "buffer zone" where producers choose to hold their margin at its current value. This ensures that producers do not overshoot the surplus available to them. An explanation of why this might occur is provided in Section 5.6.1. In order to simplify explanation, we assume that all producers use the same β value and that, when changing their margins, all producers increment or decrement their margin by the maximum amount δ. A discussion of the implications of allowing producers to choose their own values for δ and β is provided in Section 5.6. The use of our strategy is not a requirement - LBP_μ allows producers to use any strategy they wish; the only restrictions imposed are that margin updates be conducted at margin update steps, and margins are not raised or lowered at any step by a value greater than δ.

Before updating its margin, each producer p compares its belief value in its lowest cost active state bel_{p_A} with its belief value in its inactive state, bel_{p_I}. For example, if p has three states, x_u, x_v and x_w, with x_u being the inactive state, and p's belief values for these states are $x_u = -0.2$, $x_v = -0.5$ and $x_w = 1.3$, then x_v is p's lowest cost active state, and $bel_{p_I} = -0.2$ and $bel_{p_A} = -0.5$. Because each agent is assigned that state in which they believe holds the lowest cost at allocation, in comparing these values, producers determine, given their current beliefs, whether they would be active or inactive if the allocation were to be performed at the current step. Producers use the following logic to decide whether to change their margin:

- `IF` $bel_{p_I} <= bel_{p_A} \wedge \mu_p >= \delta$ `THEN` $\mu_p := \mu_p - \delta_p$
- `ELSE IF` $bel_{p_I} - \beta > bel_{p_A}$ `THEN` $\mu_p := \mu_p + \delta_p$

This means that if p believes, given bel_{P_I} and bel_{P_A}, that it would be inactive in the allocation, then, if possible, it reduces its margin μ_p by increment δ_p. This is based on the principle that it is better to make a smaller profit than not participate at all. On the

other hand, if bel_{P_A} is sufficiently lower than bel_{P_I}, a situation where p would be active in the allocation, p increases μ_p by δ_p.

5.2.3 Margin Update Example

We now present an example of how this strategy works in LBP_μ in two different scenarios: the first is a situation in which there exists only a single solution to the supply chain, and second is a situation where multiple potential solutions are present. We present two different scenarios because margin update works slightly differently when multiple potential solutions exist. For both scenarios we assume all agents use $\beta = 0.01$ in order to facilitate a clear explanation.

5.2.3.1 Single Solution

Figure 5.1: An instance of the Simple network from Walsh and Wellman [2003] with a set of production costs that allow for one positive-valued solution. In this example, $\mu = 0$, $\beta = 0.01$ and $\delta = 0.01$ for all producers.

Figure 5.1 shows a sample set of costs in the Simple network for which there exists only one positive-valued solution: P2 selling good 2 to P4, which sells good 3 to C1, generating an allocation value (and thus a total surplus) of 0.626. As we describe in Section 4.8.1, allocation values are calculated by subtracting the sum of the reserve prices of active producers from the sum of consumption values of active consumers. The other possible solution to this network, involving P1 and P3, produces a negative total surplus of -0.404. Given our minimum margin update increment of 0.01, the maximum obtainable surplus given the costs shown in Figure 5.1 is therefore 0.62. These values are reflected in the pre-update column of Table 5.1, which shows the beliefs of each participant about their inactive state and their lowest cost active state before the initial margin

update step. Producers P2 and P4 and consumer C1 each believe their lowest cost active state would lead to a negative cost - a surplus - to the allocation of 0.626, while producers P1 and P3 believe their lowest cost active state would lead to a cost of 0.404. According to the margin update rules laid out in Section 5.2.1, this leads P2 and P4 to each increase their margin by 0.01. Although producers P1 and P3 believe that their inactive states hold a lower cost than their active state, they cannot reduce their margins because we assume that all producers begin with margins equal to zero, and that producers are not interested in selling goods at a price below their reserve price. The Pre-Update 2 column of Table 5.1 shows the beliefs of agents before the second margin update step: a cumulative margin increase of 0.02 increases the cost of P2, P4 and C1's lowest cost active states before the next margin update step to -0.606. It also, as a side effect, increases the cost of P1 and P3's inactive states to the same value. Producers P2 and P4 continue to increase their margins at each margin update step until the difference between cost of their inactive state (0.000) and their lowest cost active state (0.006) means that neither of the margin update conditions specified in Section 5.2.1 are triggered. The beliefs and margins of each agent at this point are shown in the Final Beliefs column of Table 5.1. At this point, with the cumulative total of P2 and P4's margins representing the maximum obtainable surplus in this network, and with no agent compelled to update its margin, the LBP_{μ} algorithm converges, as is explained in 5.2.5, and allocation is performed.

Table 5.1: A table showing the belief values of agents in the Simple network, given the set of costs shown in Figure 5.1. The pre-update column shows each agents belief in their inactive state and lowest cost active state immediately before the first margin update step. The pre-update 2 column shows these belief values, and the margin of each agent, immediately before the second margin update step. Finally, the final beliefs column shows beliefs and margins at the point at which convergence is called. All margins immediately before the first margin update step are equal to zero.

	Pre-Update		Pre-Update 2			Final Beliefs		
Agent	Inactive	Lowest Cost Active	Inactive	Lowest Cost Active	Margin	Inactive	Lowest Cost Active	Margin
P1	-0.626	0.404	-0.606	0.404	0.000	-0.006	0.404	0.000
P2	0.000	-0.626	0.000	-0.606	0.010	0.000	-0.006	0.310
P3	-0.626	0.404	-0.606	0.404	0.000	-0.006	0.404	0.000
P4	0.000	-0.626	0.000	-0.606	0.010	0.000	-0.006	0.310
C1	0.000	-0.626	0.000	-0.606	0.000	0.000	-0.006	0.000

5.2.3.2 Multiple Solutions

Figure 5.2: An instance of the Simple network from Walsh and Wellman [2003] with a set of production costs that allow for two positive-valued solutions. In this example, $\mu = 0$, $\beta = 0.01$ and $\delta = 0.01$ for all producers.

Given the set of costs of the agents shown in Figure 5.2, there exist two positive-valued solutions to this network. The first, where P1 sells good 1 to P3, which in turn sells good 3 to C1, produces an allocation value of 0.376. The other possible solution, where P2 sells good 2 to P4 and P4 sells good 3 to C1, allows for an allocation value of 1.046. Thus, the latter of these solutions is the optimal allocation for this network given this set of costs, and 1.04 is the maximum obtainable profit given our margin update increment. We describe how the value of an allocation is determined in 4.8.1.

The Pre-Update column of Table 5.2 shows that these two possible allocation values are reflected in the beliefs of the agents immediately prior to the first margin update step. Following the margin update rule, producers P2 and P4 each increase their margin by 0.01 at the first margin step. The Pre-Update 2 column shows the result that these margin updates have on the beliefs of each agent.

The key difference between the single-solution and multiple-solution cases is revealed in the Final Beliefs column: the presence of an alternative positive-valued solution means that the inactive-state beliefs of agents in the optimal solution carry negative values rather than zeroes as in the single-solution case, meaning there is a smaller distance between the beliefs of agents in the optimal solution about their inactive and lowest-cost active states in multiple-solution cases than in single-solution ones, particularly if all producers not present in the optimal allocation hold their margins at zero as in this example. Because the margin update rules do not compel P1 and P3 to alter their margins at any point given these costs, producers P2 and P4 are able to extend their margins only up to the point at which they still represent the most efficient solution when $\mu_{P1} = 0$ and $\mu_{P3} = 0$.

This is expected behaviour: while there exists additional unclaimed surplus given this solution method, it can only be claimed by P2 and P4 if they are able to participate. If either producer was to extend their margin so that the total margins of these two agents exceeded 0.66 then P1 and P3, with margins of zero, would represent a more efficient solution and P2 and P4 would receive nothing.

Table 5.2: A table showing the belief values and margins of agents in the Simple network, given the set of costs shown in Figure 5.2. The pre-update column shows each agents belief in their inactive state and lowest cost active state immediately before the first margin update step. The pre-update 2 column shows these belief values and the margin of each agent immediately before the second margin update step. Finally, the final beliefs column shows beliefs and margins at the point at which convergence is called. All margins immediately before the first margin update step are equal to zero.

	Pre-Update		Pre-Update 2			Final Beliefs		
Agent	Inactive	Lowest Cost Active	Inactive	Lowest Cost Active	Margin	Inactive	Lowest Cost Active	Margin
P1	-1.046	-0.376	-1.026	-0.376	0.00	-0.386	-0.376	0.000
P2	-0.376	-1.046	-0.376	-1.026	0.01	-0.376	-0.386	0.330
P3	-1.046	-0.376	-1.026	-0.376	0.00	-0.386	-0.376	0.000
P4	-0.376	-1.046	-0.376	-1.026	0.01	-0.376	-0.386	0.330
C1	0.000	-1.046	0.000	-1.026	n/a	0.000	-0.386	0.000

5.2.4 Profit Distribution

When attempting to maximise profits in a supply chain formation scenario, we aim to find the allocation which maximises surplus - the difference between the reserve prices of active producers and the consumption values of active consumers - and convert the difference in surplus between this allocation and the allocation with the second-highest surplus into profits for active producers. This is accomplished through margin updates: active producers raise their prices to the point at which the total of their prices is just slightly cheaper than that of the next-best option. Because we assume that all producers follow the same margin update strategy, and all producers update their margins simultaneously, all active producers tend to profit equally, as is shown in the examples provided in Sections 5.2.3.1 and 5.2.3.2. In networks permitting solutions where multiple consumers acquire goods simultaneously, such as Many-Cons, active producers in the sub-

chains upstream of each active consumer tend to profit equally when all agents follow our margin update strategy, though the profits obtained by producers may vary between sub-chains.

5.2.5 Convergence

As with standard LBP, we make use of a convergence detector agent which is responsible for controlling termination of the algorithm but is otherwise uninvolved, and has no knowledge of the costs, margin or states of any particpating agents. As is described in Section 4.7, the convergence detector agent initiates a distributed depth-first search in order to find a spanning tree of the network, and then a series of distributed breadth-first searches to find the diameter of the spanning tree. The diameter of the spanning tree provides an upper bound for the actual diameter of the graph. This value is used for the margin update step in LBP_μ, and is broadcast to all participating agents by the convergence detector before the algorithm begins. Convergence detection in LBP_μ is performed according to the following rules: at each margin update step, each agent reports whether there was any change to its margin to the convergence detector agent. We assume that all agents report truthfully. If no agent reports a change to its margin for two consecutive margin update steps, the convergence detector agent halts the LBP_μ algorithm and begins the allocation process.

5.2.6 Allocation

The allocation process in LBP_μ is conducted exactly as it is in standard LBP. This process is explained in Section 4.8.

5.2.6.1 Allocation Value

The source of potential profits in LBP_μ is the surplus of the allocation - the difference between the consumption values of consumers that acquire a good in the allocation, and the total production costs of active producers. Allocations which maximise the surplus offer greater opportunities for producers to extend their margins and thus earn more profit. With this in mind, we measure the efficiency of the allocations produced by LBP_μ in the same way as for standard LBP. Section 4.8.1 describes how we determine the value of allocations.

5.2.7 Payments

Payments are calculated in a similar way to standard LBP, as is explained in Section 4.9 with one crucial difference. Active producers now receive a payment equal to their production cost, *plus their margin*, plus the accumulated cost of their input goods from the buyer of their output good. No payments are made to or from the mechanism.

5.2.8 Profit Maximisation

Because the goal of the LBP_μ algorithm is to allow producers to convert the surplus in these allocations into profit, we require a metric in addition to the allocation value in order to be able to judge the quality of the solutions it produces.

$$SurplusConverted = \frac{\sum_{p \in P} \mu_p}{\sum_{c \in C*} V_c - \sum_{p \in P*} R_p} \tag{5.1}$$

Equation 5.1 shows the method for calculating the performance of a technique for profit maximisation. The sum of the margins of active producers $p \in P$ is divided by the total surplus available in the *optimal* allocation, which is calculated by subtracting sum of reserve prices R_p of producers in the set of producers in the optimal allocation $P*$ from the sum of consumption values of consumers in the optimal allocation $C*$.

We use the surplus available in the optimal allocation rather than in the actual allocation to allow this metric to be used as a quality measure independently of the efficiency value. If we were to use the surplus present in the actual allocation then a technique which consistently produces suboptimal allocations with negligible efficiency and converts 100% of this into profit would be judged as superior to a technique which consistently finds the optimal allocation and converts 99% of it into profit.

5.3 Experiments

We perform 100 runs of LBP_μ for each of the networks described in Chapter 3. Production costs are drawn from the interval $U(0, 1)$ and varied between runs. Consumption values are held static between runs. We compare our results to those produced by a min-sum LBP-based naive profit-making method. In the naive method, producers report their margins as $R_p \times z$ where z is a number drawn randomly from the interval $U(0, 1)$. Our rationale for this strategy is that in a situation where producers may wish to make a profit

but are not aware of the amount of surplus available to them, it is likely that they will attach a margin to their goods for sale equal to some proportion of the cost of production. We use LBP as the basis for this strategy to allow for fair comparison with LBP_μ. We run LBP_μ until convergence is reached, as defined in Section 5.2.5. We say that the naive technique converges when no agent reports a change in the state which it believes it holds the lowest cost in consecutive steps past the approximate graph diameter. For both techniques, if convergence is not found then we run the algorithm for a number of iterations equal to $250AD$ where AD is the diameter of the spanning tree of the network. We use values of 0.08 for β for all producers, and a value of 0.01 for δ; these values were experimentally determined to provide a good balance between efficiency and profit, as explained in Section 5.4.1.

5.3.1 Performance Evaluation

We use two different metrics to assess the quality of our results: the distribution of efficiency classes produced by each method, and the average available surplus converted to profit. To determine the distribution of efficiency classes, we calculate the optimal solution for each set of production costs in a centralised manner using lp_solve, a free linear programming solver. Using this value as a benchmark, we categorise each allocation produced by each method into one of four classes: optimal, suboptimal, zero and negative. A result is optimal if the value of the allocation produced by the method is equal to that determined by LP. A suboptimal result is a positive-valued result which is less efficient than the LP value. A zero-valued result means no goods were sold in the allocation. Finally, a negative-valued result means that the sum of the production costs of active producers outweighed the sum of the consumption values of active consumers. Negative-valued allocations are caused by the presence of "dead ends" - producers which acquire an incomplete set of their input goods and so are unable to produce an output good - and are the worst possible result. The average available surplus converted to profit is calculated by dividing the sum of the total profits gained by producers in each network by the sum of the total available surplus in the optimal allocation, as explained in Section 5.2.8.

5.4 Results

5.4.1 Choosing a β value

Figures 5.3 and 5.4 show the average efficiency and average available surplus converted to profit over all networks for a range of β values.

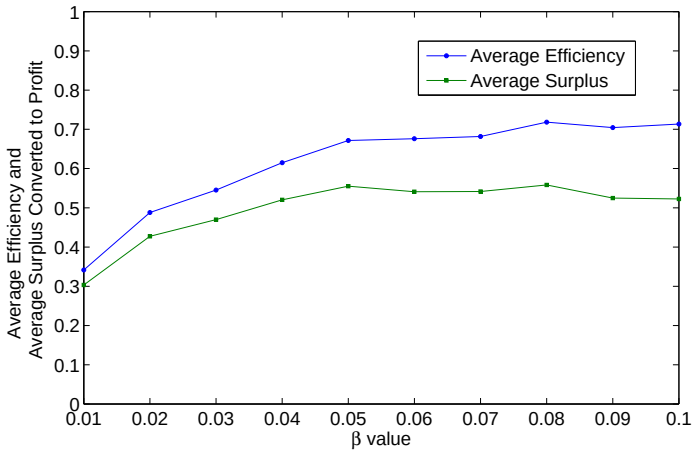

Figure 5.3: Average efficiency and proportion of surplus converted to profit by producers over all networks using global β values from 0.01 to 0.1, in steps of 0.0l. Both average efficiency and profits increase sharply at lower values, stabilising at values above 0.05.

Figure 5.3 shows results for β values from 0.01 to 0.1, in increments of 0.01. Both average efficiency and average available surplus converted to profit are poor at $\beta = 0.01$, but follow similar upward trends as β is increased. The similar trends are because, as mentioned in 4.8.1, the source of potential profits lies in finding allocations which maximise efficiency; at lower β values, where LBP_μ tends to be unable to produce efficient allocations, we also see correspondingly low profits. By raising β, which allows LBP_μ to produce more efficient allocations, we see a similar improvement in the proportion of surplus converted to profit. We did not test $\beta = 0.0$ because, as we explain in Section 5.2.1, a β value of 0.0 leads producers to attempt to equalise their bel_I and bel_A values. This leads to inefficient allocations and little or no profits for producers.

Figure 5.4 shows results for β value between 0.1 and 0.5, in increments of 0.05. We see that average overall efficiency stabilises at around 71% at β values greater than 0.1. At these values, LBP_μ is capable of producing essentially LBP-like allocations, with effi-

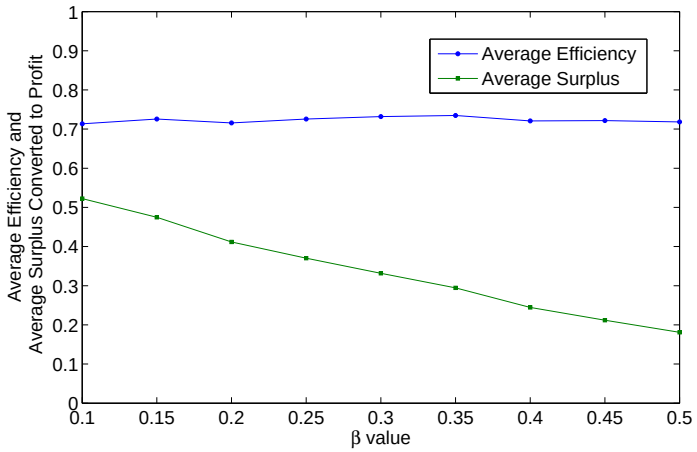

Figure 5.4: Average efficiency and proportion of surplus converted to profit by producers over all networks using β values from 0.1 to 0.5, in steps of 0.05. Profit gradually decreases as global β is increased beyond 0.1, while efficiency remains stable for all values from 0.1 to 0.5.

ciency differences between the two algorithms (standard LBP produces average overall efficiency of 87% over the twelve network instances tested) being due to the different convergence rules used, as is discussed in Section 5.4.3. As expected, average available surplus converted to profit tends to decrease at higher β values - the higher the β value used, the greater the difference there must be between producer p's bel_{p_I} and bel_{p_A} before it will decide to increase its margin.

5.4.2 Average Available Surplus Converted to Profit

Table 5.3 shows the average available surplus converted to profit, as calculated using Equation 5.1, by LBP_μ and a naive LBP-based technique for profit maximisation, details of which are provided in Section 5.3. Average available surplus converted to profit is the most important measure of the success of a technique for profit maximisation.

From Table 5.3, we see that LBP_μ enabled producers to convert at least 43% of the available efficiency into profit for ten of the twelve network instances, with several values considerably higher than this. LBP_μ also consistently allows producers to convert a greater amount of surplus into profit than the naive profit-making approach. In this set of experiments, as for all the sets of experiments in this section, we use a β value of 0.08, which was experimentally determined to be a value which produces good profits

Table 5.3: Average available surplus converted to profit by producers using LBP_μ and a naive LBP-based profit-making technique. A result of 1.000 is equivalent to the conversion of 100% of available surplus in the optimal solutions into profit.

Network	LBP_μ	Naive technique
Simple	0.574	0.237
Unbalanced		
CE	0.679	0.367
No CE	0.636	0.053
Two-Cons		
CE	0.748	0.218
No CE	0.464	0.295
Bigger	0.473	0.285
Many-Cons	0.911	0.334
Greedy-Bad		
CE	0.934	0.328
No CE	0.848	0.113
Harder		
CE	0.341	-0.208
No CE	-0.342	-0.98
Huge	0.431	0.2

on the majority of the networks we tested. While smaller values than this for β might produce more profit on some networks, on others, particularly loopy networks, smaller values may lead to less efficient allocations and smaller profits. From Table 5.4 we see that there exists a moderate amount of spread within our results. This is mostly due to the sometimes strongly negative profits produced when negative-valued allocations are found.

5.4.3 Efficiency

Table 5.5 shows the distribution of efficiency classes produced over 100 runs by each method over each network instance. We see that LBP_μ was able to outperform the naive LBP-based profit-making method on all but two of the network instances tested. Of the two networks in which the naive method performs better, in one - Unbalanced CE - the difference is very slight, while in the other, Two-Cons No CE, both LBP-based methods tend to oscillate between optimal, zero and negative-valued solutions. This oscillation is brought about by the loopy nature of the network. The large difference in efficiency

Table 5.4: Median and interquartile range measures for the average available surplus converted to profit by LBP_μ

Network	LBP_μ median	LBP_μ interquartile range
Simple	0.744	0.357
Unbalanced		
CE	0.931	0.120
No CE	0.760	0.648
Two-Cons		
CE	0.820	0.254
No CE	0.911	1.187
Bigger	0.611	0.901
Many-Cons	0.942	0.054
Greedy-Bad		
CE	0.933	0.073
No CE	0.818	0.293
Harder		
CE	0.000	0.877
No CE	0.000	0.000
Huge	0.442	0.379

results between the two LBP-based methods in this network instance is brought about by the different convergence rules used - the naive method is able to converge at any step in which all agents report that their lowest-cost beliefs have not changed, allowing it to converge at a point in which the solution has converged to an optimal one. On the other hand, LBP_μ uses a more restrictive rule, necessary to allow margin updates, which only allows convergence at multiples of the approximate diameter of the network. This, in some cases, leaves LBP_μ unable to converge at a point where the solution has oscillated to an optimal one.

5.4.4 Average Efficiency

Table 5.6 shows the average efficiency achieved over 100 runs for each network by LBP_μ and the naive-profit making LBP-based techique as a fraction of the available efficiency. An average efficiency of 1.000 indicates that 100% of the available efficiency was captured on each of the hundred runs for that particular network instance, and represents the best possible result. Negative values indicate that over 100 runs the method recorded negative average efficiency in that network. For example, a result showing -1.000 aver-

Table 5.5: Distribution of efficiency classes produced by LBP_μ and an LBP-based approach with producers which follow a naive profit-seeking strategy over twelve network instances. Efficiency classes are Negative, Zero, Suboptimal and Optimal.

	LBP_μ % of instances				Naive profit-seeking LBP % of instances			
Network	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Simple	0.0	0.0	0.0	100.0	0.0	53.0	1.0	46.0
Unbalanced								
CE	7.0	1.0	6.0	86.0	0.0	42.0	7.0	51.0
No CE	0.0	3.0	0.0	97.0	0.0	97.0	2.0	1.0
Two-Cons								
CE	2.0	0.0	0.0	98.0	0.0	48.0	12.0	40.0
No CE	31.0	0.0	16.0	53.0	0.0	3.0	21.0	76.0
Bigger	6.0	1.0	2.0	91.0	0.0	75.0	2.0	23.0
Many-Cons	0.0	0.0	0.0	100.0	0.0	47.0	35.0	18.0
Greedy-Bad								
CE	0.0	0.0	0.0	100.0	0.0	69.0	0.0	31.0
No CE	0.0	0.0	0.0	100.0	0.0	94.0	0.0	6.0
Harder								
CE	1.0	60.0	2.0	39.0	15.0	25.0	0.0	60.0
No CE	24.0	64.0	0.0	12.0	42.0	48.0	0.0	10.0
Huge	3.0	2.0	94.0	1.0	0.0	59.0	0.0	41.0

age efficiency means the method achieved, on average, -100% of the maximum available efficiency value. Because we are measuring results as a fraction of the efficient value, solutions with very high negative efficiency can lead to average efficiency values below -1.000.

We see from Table 5.6 that LBP_μ produces greater efficiency than the naive LBP-based profit-making technique over all but two network instances. If we refer to the average efficiency results for SAMP-SB and SAMP-SB-D from Chapter 4, we also see that LBP_μ essentially equals or improves upon the efficiency results of SAMP-SB for the majority of networks. SAMP-SB-D tends to produce slightly better overall efficiency for most networks than LBP_μ is able to, but this is due in large part to the assumption in this protocol that contracts may be voided after the allocation has been finalised.

Table 5.6: Average efficiency in each network produced by LBP_μ and the naive profit-making LBP-based technique. A result of 1.000 is equal to the capture of an average of 100% of available efficiency, while a result of -1.000 is equal to an average capture of -100% of available efficiency. Note that while 1.000 is the maximum achievable positive value, it is possible to produce negative overall efficiencies below -1.000.

Network	LBP_μ average efficiency	Naive profit-seeking LBP average efficiency
Simple	1.000	0.724
Unbalanced		
CE	0.778	0.722
No CE	0.944	0.066
Two-Cons		
CE	0.916	0.703
No CE	0.399	0.939
Bigger	0.827	0.498
Many-Cons	1.000	0.562
Greedy-Bad		
CE	1.000	0.5
No CE	1.000	0.151
Harder		
CE	0.455	0.49
No CE	-0.330	-0.816
Huge	0.630	0.319

5.4.5 Messages and Bids Before Convergence

Table 5.7 shows a comparison of the number of messages passed by LBP_μ and the naive LBP-based profit maximisation technique over each network, and a comparison of the bandwidth required by each method. It is clear that LBP_μ tends to require a vastly higher number of messages to be passed over most networks than the naive technique, but this is to be expected given the difference in approaches taken by these algorithms. The sole network in which LBP_μ requires less messages and bandwidth is Harder - for both CE and non-CE instances of this network, LBP_μ requires fewer messages to be passed and less bandwidth to be used, but this is due to LBP_μ converging very quickly and frequently to a large number of zero-valued solutions in this network. Although the naive method is functionally identical to the algorithm described in Chapter 4, significant differences exist in the bandwidth requirements for the two methods on some networks.

This is due to the different upper bounds on the number of steps each algorithm may run for - the algorithm in Chapter 4 runs for a maximum of 250 steps, while the naive algorithm, for fair comparison with LBP_μ, runs for a maximum of $250AD$ steps, where AD is the approximate graph diameter. The number of margin update steps (and thus the number of messages and bandwidth) could be reduced by increasing the increment by which producers raise or lower their margins or increasing the β value used; unfortunately, both of these measures would also incur a reduction in the amount of surplus producers are able to convert into profit.

5.5 Mechanism Properties

In this section, we present the mechanism properties of LBP_μ, comparing them to those of standard LBP, SAMP-SB and SAMP-SB-D.

5.5.1 Individual Rationality

A mechanism is individually rational if participants cannot incur negative utility by participating. As with standard LBP, we cannot guarantee the individual rationality of our approach given the potential for dead ends in our allocations, though it could be guaranteed using a process of post-allocation decommitment similar to that which is used in SAMP-SB-D.

Table 5.7: Average numbers of messages passed and average bandwidth required before convergence in each network using LBP_μ and the naive LBP-based approach for profit maximisation

Network	LBP_μ average number of messages passed	LBP_μ average total bandwidth required	Naive LBP average number of messages passed	Naive LBP average total bandwidth required
Simple	728.0	1638.0	48.16	108.36
Unbalanced				
CE	31776.8	103620.0	485.76	1584.0
No CE	3054.4	9960.0	492.2	1605.0
Two-Cons				
CE	1893.5	4598.5	104.16	252.96
No CE	4778.2	11604.2	138.46	336.25
Bigger	30297.6	181785.6	2982.24	17893.44
Many-Cons	6996.0	15449.5	488.64	1079.08
Greedy-Bad				
CE	1600.2	3911.6	120.06	293.43
No CE	710.1	1735.8	119.88	293.04
Harder				
CE	70824.4	386738.4	189367.7	1034047.04
No CE	355695.2	1942283.0	532819.3	2909473.81
Huge	284107.2	901305.7	4844.16	15367.68

5.5.2 Incentive Compatibility

Unfortunately, as with standard LBP, LBP_μ is not incentive compatible for producers or consumers.

5.5.3 Budget Balance

As with standard LBP, LBP_μ involves no payments to or from the mechanism, and is therefore strongly budget balanced.

5.5.4 Allocative Efficiency

The results presented in Table 5.6 suggest that over the networks tested and with the β value used, our approach produces slightly less allocative efficiency than standard LBP. This is to be expected given the constant alteration of unary values brough about by margin updates. Although perfect allocative efficiency can be guaranteed for acyclic networks if β is set to a value large enough so that no producers wish to change their margins at any point, leading to behaviour identical to that of standard LBP, this produces no profit and thus defeats the purpose of the mechanism.

5.6 Discussion

Throughout this chapter, we have made the assumption that all producers use the same values for their buffer zone β and their margin update increment δ. These abstractions were made in order to simplify the presentation of our algorithm, and are similar to the abstractions made in terms of agent bidding behaviours in Walsh and Wellman [2003]: in this work, all producers and consumers bid according to the same sets of rules, and in most cases increase the value of their bids by a global increment value. Nevertheless, we recognise that, in practice, agents may wish to set their own values for β and δ. In this section, we discuss the implications of producers being able to set their own values for these variables.

5.6.1 Producer-specific $beta$ values

Producers are able to increase their margins when the difference between its beliefs in its lowest cost active state and its inactive state exceeds a certain amount - the value of β. A

positive β value is necessary because without it, producers would increase their margins to a point at which the beliefs in their lowest cost active state and their inactive state are equal. This is undesirable because producers assign themselves with the inactive state if they believe their inactive state and lowest cost active state have equal costs.

β values must also be large enough to avoid situations where producers overshoot the amount of surplus available to them, which can lead to an unending sequence of increases and decreases in margins. This situation is possible because all producers update their margins simultaneously. As we explain in 5.2.1, producers increase their margins if their belief in their lowest cost active state minus β is greater than their belief in their lowest cost active state. If $\beta = 0$, the value of this calculation is equal to the amount of surplus available to the agent and the other producers on that agent's sub-chain. While there may be sufficient surplus for one producer to increase its margin, there may not be sufficient surplus for all other producers on that sub-chain to also increase theirs.

5.6.2 Producer-specific δ values

δ is the increment by which producers are able to increase or decrease their margins. We used a global value of 0.01 for δ, with the justification that for all but one of the networks we tested, values for available surplus have a maximum two decimal places.[1] In a best-case scenario - an acyclic network, with a global β value of 0.01 and an available surplus with two decimal places - a value of 0.01 for δ theoretically allows for the possibility of converting all but 0.01 of the available surplus into profit. This value could be decreased to 0.001 or 0.0001 and so on by using δ values of 0.001, 0.0001 and so on, at the cost of slower convergence.

Because producers are aware of the total surplus available to them (and all of the producers in their sub-chain) through the difference between their lowest cost active state and their inactive state, if we allow producers to update their values for δ as the algorithm is running then it is likely that all producers will immediately choose values for δ equal to the total surplus available. This is, of course, undesirable: if all of the producers in a subchain simultaneously try to acquire all of the surplus available to that subchain, the cumulative increase in prices will render that subchain non-viable. It is therefore logical to specify a small upper limit to the margin update increment, as we do, to ensure that producers benefit equally and do not price themselves out of the market unnecessarily.

[1] The Simple network is the exception: Consumer C1 in this network has a consumption value with three decimal places, allowing for the possibility of available surplus with three decimal places.

5.7 Conclusions

In this chapter we formalised a new solution goal for the supply chain formation problem called profit maximisation, and introduced an intelligent decentralised LBP-based technique for solving this problem called LBP_μ. LBP_μ enables producers to alter their margins with the aim of making a profit by participating in the supply chain. Because LBP_μ is the first instance of a technique for profit maximisation in supply chain formation, we tested its ability to convert available surplus into profit against a naive LBP-based technique for profit maximisation. Our results suggest that LBP_μ is reliably able to outperform the naive approach, and converts a significant amount of available surplus into profit over the majority of the network instances we tested. Because the amount of available surplus depends on the efficiency of the underlying allocation, we also examined the efficiency of the solutions produced by LBP_μ, and compared these efficiency results to those of the naive LBP-based profit maximisation method as well as the SAMP-SB and SAMP-SB-D auction protocols from Walsh and Wellman [2003]. LBP_μ was able to match or outperform the naive and auction-based methods on the majority of networks tested, while performing comparably to LBP, and avoided the consistently poor allocations produced by the auction protocols in the absence of competitive equilibrium. LBP_μ is also budget balanced.

Having investigated the performance of an LBP-based approach for profit maximisation, in the next chapter we test LBP's capability to determine allocations in a decentralised supply chain formation scenario involving the exchange of multiple units of goods.

Chapter 6

Multi-Unit Supply Chain Formation using Min-Sum LBP

In this chapter, we propose a framework for the representation of multi-unit supply chains and extend the single-unit loopy belief propagation (LBP) based technique for decentralised SCF presented in Chapter 4 to the multi-unit case. We aim to demonstrate that min-sum LBP is capable of scaling up and continues to produce allocations with strong efficiency when applied to this more complex supply chain formation problem. This extension represents the first instance of a fully decentralised multi-unit supply chain formation scenario, and brings our work into line with the multi-unit property of the current state of the art in centralised supply chain formation techniques, namely mixed multi-unit combinatorial auctions. We also introduce the additional constraints of production capacities and input-to-output good ratios for producers, as well as desired good quantities for consumers, greatly increasing the state space over standard single-unit LBP. We compare the allocations produced by multi-unit LBP to the results of the auction protocols presented in Walsh and Wellman [2003], which we extended to allow for multi-unit exchanges. Our results indicate that LBP is capable of generating consistently optimal allocations in the multi-unit case, outperforming the distributed auction protocols and performing comparably to single-unit LBP, whilst operating in a fully decentralised manner. Our results also suggest that LBP is able to converge to optimal solutions by passing a number of messages far smaller than the number of bids required when using the auction-based approaches.

6.1 Multi-Unit Model

We extend this representation with input good ratios, production capacities and consumer desired good quantities in order to model the multi-unit case. This is shown in Figure 6.1. Production capacities and consumer desired good quantities are measured in whole units of the good in question. A producer with a single input and an input ratio of 2 for that good requires two units of that good in order to produce one unit of its output, four units of that good to produce two of its output, and so on.

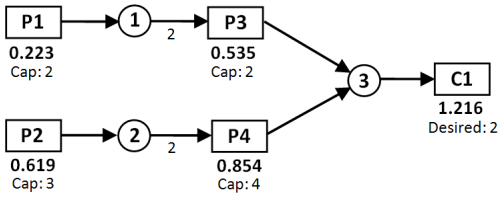

Figure 6.1: The Simple supply chain task dependency network extended to the multi-unit case. Producers (P1, P2, P3, P4) and a consumer (C1) are represented by rectangles, while goods are represented by circles. Edges between vertices indicate potential flows of goods. Numbers immediately below producers represent production costs, while numbers immediately below consumers indicate consumption values. Our extension to the multi-unit case includes the following additional features: the values given below production costs and consumption values indicate production capacities for producers and desired consumable good quantities for consumers, respectively, and are measured in whole units of the good in question. Edges from goods to producers are labelled with the producer's input-to-output ratio for that good.

In our example, we see that producer P1 is able to produce up to 2 units of good 1, at a cost of 0.223 for each unit of good 1 it produces. Producer P3 requires 2 units of good 1 (as signified by the edge from good 1 to P3) to produce a single unit of its output good, good 3. Although P3 has the capacity to produce up to 2 units of good 3, this would require P3 to obtain 4 units of good 1, which is not possible in this network instance given P1's maximum output capacity of 2. Similarly, producer P2 is able to produce up to 3 units of good 2 at a cost of 0.619 per unit, and producer P4 requires 2 units of this good in order to produce one unit of good 3. Consumer C1 desires a maximum of 2 units of good 3, and obtains a consumption value of 1.216 for each unit of good 3 that it acquires.

6.1.1 Producers

Producers are capable of producing multiple units of a single type of output good. At initialisation, each producer is assigned a production capacity CAP_p which specifies the maximum number of units each producer is able to produce of its output good.

In order to produce one unit of their output good, producers are required to acquire a number of units of each of their input goods equal to their input good ratio R_{p_g} for that good. In order to produce two units of their output, producers require twice as many units of their inputs as for one good, three units of output will require three times as many units of inputs, and so on. Input good ratios are assigned to producers at initialisation, and each producer may have different ratios for each of their input goods. Producers which do not require any inputs to produce their output good are known as no-input producers, and form the initial echelon of the supply chain. If a producer requires multiple types of input good, we refer to these goods as complementary. A producer cannot produce its output good unless it acquires the necessary number of all of its input goods.

Producers incur a production cost C_p in producing their output good, which is a producer-specific constant. Production costs model the expense incurred by a producer in producing a single unit of their output, plus some small additional fixed profit margin. They therefore can be said to be equivalent to sale price of a single unit of the producer's output good. The total production cost incurred by a producer is linear with the number of units of its output good that it produces: if a producer produces two units of its output good, it incurs a production cost equal to $2C_p$, a cost of $3C_p$ if it produces three units, and so on.

6.1.2 Consumers

Consumers seek to acquire a number of units of their consumable good no greater than their desired consumable good quantity DES_c. In each network, each consumer is assigned a static consumption value V_c: this is the valuation the consumer holds for obtaining a single unit of its consumable good. Similar to production costs, the total value a consumer receives is linear with the number of goods a consumer obtains: if a consumer acquires two units of its consumable good it receives $2V_c$, if it acquires three units it receives $3V_c$ and so on.

6.1.3 States

Due to the fixed structure of the networks, for each agent there exists a finite number of purchases and sales (if the agent is a producer) in which the agent is viable, i.e. it acquires the necessary number of units of its input goods and sells the corresponding quantity of its output good. We encode each of these tuples of exchange relationships as states. For producers, each state defines a list of suppliers, a quantity bought from each supplier, a list of buyers and the quantity sold to each buyer. For consumers, a state lists a set of suppliers and the quantity bought from each supplier. For example, a possible state for producer P3 in Figure 6.1 is "Buy 2 units of good 1 from P1 and sell to 1 unit of good 3 to C1". The number of states an agent possesses increases with the number of producers able to supply its input good(s), its production capacity or number of desired consumable good quantity, and the number of producers or consumers able to consume its output good. Because we model each type of good as an identical commodity, a producer or consumer can buy multiple units of the same good from different producers. For example, C1 derives equal value from acquiring 2 units of good 3 from Producer P2 and 1 unit of good 3 from Producer P3 as it does from acquiring 3 units of good 3 from either Producer P2 or P3. As well as a list of active states, we also allow for the inactive state, where the agent does not acquire or produce any goods.

6.1.4 Unary Cost

Each agent associates each of its states with a cost. For all agents, the cost of being in the inactive state is zero. For producers, all active states incur a positive cost, equal to the total reserve price, which is in turn equal to the producer's R_p multiplied by the number of units the producer produces in that state. Consumers assign a negative cost to all states in which they acquire a good. If the consumer acquires a single unit of their consumable good in the state in question, the cost is equal to $0 - V_c$, where V_c represents the consumer's consumption value, the value the consumer assigns to the acquisition of its consumable good. If the consumer acquires two goods, the cost is equal to $0 - 2 * V_c$, with the cost decreasing linearly as the number of goods the consumer obtains increases.

6.1.5 Pairwise Cost

Pairwise costs encode the compatibility of two of the states of a pair of neighboring agents. Two states are compatible if agent u's state lists agent v as a buyer and the list

of sellers in v's state includes u, and the number of units sold by u to v in u's state is equal to the number of units bought by v from u in v's state and vice versa. They are also compatible if agent u's state does not list agent v as a buyer and v's state does not list agent u as a seller and vice versa, or if both states are inactive states. If the states are compatible, the pairwise cost is equal to zero. If the two states do not meet any of these conditions, they are incompatible, and the pairwise cost of this combination of states is equal to positive infinity.

6.1.6 Convergence

Convergence detection is performed by a dedicated convergence detector agent, as outlined in Section 4.7.

6.1.7 Allocation

Allocation is performed as for the single-unit LBP model, as explained in section 4.8, with one difference. In order for an agent to be considered active, the sellers and buyers of its goods must not only list the agent in question in their final states, but the quantity values encoded in the states must also align *exactly* in order to be valid.

For example, the following combination of final states in the simple network are *invalid*, and would result in edges being removed between all of the listed agents:

- P1 : Sell 4 units of good 1 to P3.
- P3 : Buy 6 units of good 1 from P1; sell 3 units of good 3 to C1.
- C1 : Buy 3 units of good 3 from P3.

6.1.8 Allocation Value

We calculate the allocation value for the multi-unit case in a similar way to the standard single-unit LBP-based approach. This is shown in Equation 6.1. We first sum the product of the consumption value V_c of each active consumer c in the set of active consumers C and the number of goods obtained by c, A_c. From this, we subtract the sum of the product of the reserve price R_p of each active producer p in the set of active producers P and the

number of goods manufactured by p, M_p to produce a final allocation value.

$$Val = \sum_{c \in C} V_c A_c - \sum_{p \in P} C_p M_p \tag{6.1}$$

6.1.9 Payments

An active producer p is paid a fee by each of the buyers of its output good, at a price equal to the value of Eq. 6.2.

$$F_{bp} = R_p M_{pb} + \frac{M_{pb}}{M_p} \sum_{i \in SP} F_{pi} \tag{6.2}$$

F_{bp} is the fee paid from buyer b to producer p, R_p is p's reserve price, M_{pb} is the number of goods manufactured by p for buyer b, M_p is the total number of goods produced by producer p, F_{pi} is the fee paid by p to each supplier i from p's set of suppliers SP. Buyers therefore pay p's marginal cost of producing each good they purchase plus a proportion of p's input good costs commensurate with the proportion of p's total output they purchase. No payments are made to or by the mechanism.

6.2 Experiments

We test LBP and multi-unit implementations of two decentralised auction protocols from Walsh and Wellman [2003] over the complete set of networks presented in Chapter 3. We extended SAMP-SB and SAMP-SB-D according to the suggestion proposed in Walsh and Wellman [2003], by using multiple copies of each agent to represent each unit of a given producer's capacity or consumer's number of desired goods. Because this representation does not allow for the use of input good ratios, as well as the experiments with heterogeneous ratios we also test LBP with all ratios set to a value of 1, which we refer to as ratioless LBP. Ratioless LBP is identical to the multi-unit version of LBP we explain in this chapter with the exception that inputs are converted to outputs on a 1:1 basis. This is to allow for fair comparison with the results of the auction protocols. Ratios serve to constrain the number of positive-valued solutions in the network, which may present the algorithm with a more difficult problem, particularly in loopy networks. Conversely, the use of ratios greater than 1 also constraints the state space of each agent, potentially reducing the bandwidth required to form solutions. We perform 100 runs of ratioless

LBP, standard multi-unit LBP, SAMP-SB and SAMP-SB-D for each network, varying input ratios and consumer desired goods (for LBP) as well as production costs and production capacities (for all methods) between each run. We discard runs in which the optimal allocation value, determined using mixed integer programming, is non-positive. Production costs are drawn from the distribution $U(0, 1)$, production capacities from the distribution $(4 \ldots 5)$, consumer desired goods from $(2 \ldots 3)$, and input good ratios from $(1 \ldots 2)$. We use these fairly large values for production capacities and consumer desired goods and fairly low values for input good ratios to allow for feasibility in larger networks; however, as long as a positive-valued solution exists, the performance of LBP is largely unaffected by the values used. Consumption values are fixed at the values given underneath each consumer (C1, C2 and so on) over every run.

6.2.1 Performance Evaluation

We run ratioless and standard multi-unit LBP on each network until a convergent state is reached, using the value of the allocations produced as a measure of the quality of our solutions. If LBP does not converge before 50 iterations, we record the result as a zero-valued allocation, which indicates that no solution was found. We use lp_solve, a free mixed integer programming solver, in order to provide a benchmark optimal allocation value to compare LBP against. To provide a basis for comparison with other decentralised multi-unit SCF techniques, we also compare the results of LBP with those produced by multi-unit implementations of the double auction protocols presented in Walsh and Wellman [2003], namely SAMP-SB and SAMP-SB-D.

6.3 Results

6.3.1 Efficiency Classes

Table 6.1 shows the distribution of efficiency classes produced by LBP without ratios, LBP with ratios, and our multi-unit implementations of the SAMP-SB and SAMP-SB-D double auction protocols from Walsh and Wellman [2003]. We see that the performance of LBP and SAMP-SB are roughly equivalent on the majority of networks, regardless of whether ratios are enabled or disabled. Efficiency results produced by LBP are also comparable to those of SAMP-SB-D for many networks, despite the latter's advantage of being able to decommit from inefficient solutions. As expected, LBP produces 100% optimality on Simple and Many-Cons, both of which acyclic networks. We also see that

all ratios being set to a value of 1 typically leads to more efficient results for LBP, with what would appear to be one exception - the Huge network. However, as the following section shows, despite a larger number of optimal results, LBP with ratios set to 1 produces a greater amount of average efficiency for this network than LBP with larger ratios. The fact that LBP cannot be guaranteed to converge in graphs with more than a single loop means that we can offer no definitive reason for why LBP with larger ratios appears to be able to produce a larger percentage of optimal solutions than LBP with ratios set to 1 for this network.

We suggest that the overall trend towards slightly better efficiency for LBP with ratios set to 1 is because the use of larger ratios constrains the number of positive-valued solutions in the network, presenting the algorithm with a slightly more difficult problem. The presence of larger numbers of suboptimal allocations and fewer zero and negative allocations for LBP with ratios set to 1 appears to support this. With regards to the statistical significance of our results, over 10 sets of 100 runs, the network with the largest average standard deviation over all efficiency classes was Two Cons for both ratioless and standard multi-unit LBP, with standard deviations of 1.61% and 1.69% respectively. These values indicate that LBP offers consistent performance over all of the networks tested.

Table 6.1: Distribution of efficiency classes produced by LBP without ratios, LBP with ratios, SAMP-SB and SAMP-SB-D. Classes are Negative, Zero, Suboptimal and Optimal.

	Ratioless LBP				LBP with ratios				SAMP-SB				SAMP-SB-D		
	% of instances				% of instances				% of instances				% of instances		
Network	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Zero	Sub	Opt
Simple	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	1.0	99.0	0.0	1.0	99.0
Unbalanced	0.0	1.0	19.0	80.0	0.0	10.0	22.0	67.0	0.0	0.0	12.0	88.0	0.0	4.0	96.0
Two-Cons	0.0	0.0	2.0	98.0	0.0	0.0	2.0	98.0	6.0	0.0	20.0	74.0	0.0	5.0	95.0
Bigger	0.0	3.0	0.0	97.0	3.0	2.0	1.0	94.0	0.0	0.0	1.0	99.0	0.0	0.0	100.0
Many-Cons	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	23.0	0.0	38.0	39.0	0.0	4.0	96.0
Greedy-Bad	0.0	4.0	29.0	67.0	0.0	14.0	17.0	69.0	12.0	2.0	12.0	74.0	5.0	11.0	84.0
Harder	41.0	5.0	10.0	44.0	45.0	26.0	2.0	27.0	11.0	0.0	41.0	48.0	1.0	29.0	70.0
Huge	0.0	3.0	91.0	6.0	4.0	22.0	11.0	63.0	0.0	0.0	11.0	89.0	0.0	2.0	98.0

6.3.2 Average Efficiency

Table 6.2 shows the average efficiency produced by each method - multi-unit LBP without ratios, multi-unit LBP with ratios, and our multi-unit implementations of the SAMP-SB and SAMP-SB-D auction protocols. In keeping with the results presented in the previous section, we see that the performance of both LBP-based methods is roughly equivalent to that of SAMP-SB on most networks, whilst also matching the results produced by SAMP-SB-D on many network instances. The auction-based methods outperform the LBP-based methods on large, loopy networks such as Bigger, Huge and Harder, while LBP tends to perform better on acyclic networks such as Simple and Many-Cons. From Table 6.3 we see that for most networks our results are consistent, with little spread.

Table 6.2: Average efficiency in each network produced by both multi-unit versions of LBP, and multi-unit implementations of the SAMP-SB and SAMP-SB-D protocols from Walsh and Wellman [2003]. A result of 1.000 is equal to the capture of an average of 100% of available efficiency, while a result of -1.000 is equal to an average capture of -100% of available efficiency. Note that while 1.000 is the maximum achievable positive value, it is possible to produce negative overall efficiencies below -1.000.

Network	LBP without ratios average efficiency	LBP with ratios average efficiency	SAMP-SB average efficiency	SAMP-SB-D average efficiency
Simple	1.000	1.000	0.999	0.999
Unbalanced	0.962	0.872	0.964	0.998
Two-Cons	0.986	0.983	0.963	0.998
Bigger	0.969	0.813	0.995	1.000
Many-Cons	1.000	1.000	0.425	0.997
Greedy-Bad	0.91	0.839	0.666	0.923
Harder	0.11	-0.058	0.686	0.947
Huge	0.625	0.583	0.989	0.998

Table 6.3: Median and interquartile range measures of the average efficiency results produced by LBP without ratios and LBP with ratios.

Network	LBP without ratios median	LBP without ratios interquartile range	LBP with ratios median	LBP with ratios interquartile range
Simple	1.000	0.000	1.000	0.000
Unbalanced	1.000	0.000	1.000	0.159
Two-Cons	1.000	0.000	1.000	0.000
Bigger	1.000	0.000	1.000	0.000
Many-Cons	1.000	0.000	1.000	0.000
Greedy-Bad	1.000	0.079	1.000	0.297
Harder	0.854	1.648	0.000	1.761
Huge	0.637	0.307	1.000	1.000

Table 6.4: A table showing wins, draws and losses in the multi-unit scenario between LBP and SAMP-SB. One X in the LBP column means that only ratioless LBP was able to outperform SAMP-SB. Two Xs means that both ratioless and LBP with ratios outperformed SAMP-SB, which does not use ratios. A draw is a result with less than 1% difference between the best-performing LBP method and SAMP-SB and is shown with an X in both columns.

Network	LBP	SAMP-SB
Simple	X	X
Unbalanced	X	X
Two-Cons	XX	
Bigger		X
Many-Cons	XX	
Greedy-Bad	XX	
Harder		X
Huge		X

6.3.3 Messages and Bids Before Convergence

From Table 6.5 we see that, in most cases, the LBP-based methods tend to require fewer messages to be sent in order to converge to a solution than is required in the auction-

based methods, and the total bandwidth required also tends to be smaller. The number of bids and price quotes exchanged in SAMP-SB and SAMP-SB-D are identical - the only difference between the protocols is a post-allocation decommitment stage - so we do not include values for SAMP-SB-D in this table. It is also clear that despite the differences in average efficiency between LBP with ratios and LBP without ratios, both methods converge at roughly the same point in all networks. The difference in bandwidth requirements between the methods is because the use of input good ratios greater than 1 constraints the state space. When price quotes are taken into account, the frequency of information exchange in SAMP-SB tends to be orders of magnitude greater than in LBP, offsetting the fact LBP messages tend to encode more information and thus are of a larger size than bids in SAMP-SB.

Table 6.5: Average numbers of messages passed before convergence in each network using multi-unit LBP without and with ratios compared with the average numbers of bids placed in each network before quiescence in the SAMP-SB protocol from Walsh and Wellman [2003].

Network	LBP without ratios average number of messages passed	LBP without ratios average bandwidth required	LBP with ratios average number of messages passed	LBP with ratios average bandwidth required	SAMP-SB average number of bids placed	SAMP-SB average number of bids placed and price quotes sent
Simple	51.76	315.89	52.96	296.88	210.64	2419.49
Unbalanced	494.96	11849.44	518.42	8795.64	1201.21	21098.79
Two-Cons	106.54	1019.02	108.64	965.29	905.38	12009.91
Bigger	2859.84	386634.6	2964.96	269491.9	2239.43	71276.2
Many-Cons	498.72	3517.91	488.64	2545.52	5797.36	68017.69
Greedy-Bad	118.62	836.64	119.7	789.58	1406.37	18379.73
Harder	16315.68	2413542.0	16748.88	1270947.0	4048.03	180133.4
Huge	4732.8	116968.1	4678.86	96585.71	8803.88	158705.21

6.4 Conclusions

In this chapter, we proposed a framework for the representation of multi-unit supply chains and presented a novel technique for multi-unit decentralised supply chain formation based upon the min-sum loopy belief propagation algorithm. By extending the task dependency network representation of supply chain networks proposed by Walsh and Wellman [2003] to the multi-unit case, converting these networks into pairwise MRFs, and mapping the capabilities of each agent into a set of states, we were able to allow agents to exchange beliefs about the optimal structure of the allocation. This is done whilst granting agents only local information about the structure of the network, and requires our agents to share no more private information about their costs (because production costs represent an actual cost plus a small, fixed profit margin) than would be revealed in an open auction or a series of negotiations. Our LBP-based method performed essentially equivalently to two well-known auction-based approaches operating under comparable conditions over the networks tested, and continued to produce good results for most networks even while operating under more strict constraints. Our results also suggest that LBP is able to converge to optimal solutions much more quickly for most networks than the auction-based methods we tested against. The game theoretic properties discussed in Section 4.11.5 also hold for the multi-unit case.

In Chapter 7 we apply multi-unit LBP to a dynamic decentralised supply chain formation scenario, and test its performance under conditions where participants are able to change any of their properties or to enter or leave the network while the algorithm is running.

Chapter 7

Dynamic Supply Chain Formation using Min-Sum LBP

As discussed in Chapter 1, one of the key benefits of automated computational techniques for supply chain formation is their ability to allow organisations to become more adaptable and responsive to emerging opportunities, and to take advantage of these opportunities more efficiently. So far, we have demonstrated the efficacy of LBP in static environments - aside from the dynamic pricing mechanism laid out in Chapter 5, the properties of potential participants have remained unchanged once the process of supply chain formation has begun. This means that producers and consumers are locked in to participating once the process is underway, and, conversely, that potential additional participants who might be able to play a part in creating a more efficient allocation are locked out. In order to be able to promise greater adaptability and responsiveness to participants, it is desirable for a mechanism to be able to cope with changes to the parameters of the potential supply chain right up until the moment at which the allocation is finalised. This allows for the inclusion of new participants who join as the supply chain is being formed, the ability to deal with the departure of existing participants, and a means to take into account changes to the properties of participants such as reserve prices and production capacities. Auction-based protocols are inherently unaffected by these issues, but the nature of LBP as a message-passing algorithm allows for the possibility that the beliefs of agents may be rendered inaccurate by these changes.

In this chapter we aim to demonstrate that min-sum LBP continues to perform well under a dynamic supply chain formation scenario in which participants are able to enter and leave the market and to change their properties as the supply chain is being formed.

As we showed in Chapter 5, LBP is generally able to cope well with alterations to the reserve prices of participants while the algorithm is underway. This chapter presents a much more difficult scenario by allowing for a wide variety of potential changes to the structure and properties of the original network.

7.1 Model

We implement the reconfiguration scenario in our multi-unit LBP model from Chapter 6. The potential alterations to the network we allow for can be grouped into three categories - new entrants, departures and property changes. A full list of possible alterations, sorted by category is shown in Table 7.1. We explain the details of each of these three categories in the following subsections.

Table 7.1: A full list of the alterations we allow for, sorted by category.

Category	Alteration Type
New Entrant	Producer Added
New Entrant	Consumer Added
Departure	Producer Removed
Departure	Consumer Removed
Property Change	Production Cost
Property Change	Ratio
Property Change	Production capacity
Property Change	Consumption value
Property Change	Consumer desired goods

7.1.1 New Entrants

In the interest of offering a mechanism capable of rapid response to new entrants into the market, we allow for the possibility of new producers and consumers entering the process at any point before the final allocation is determined.

7.1.2 Departures

If a producer or consumer signals that it wishes to leave the market, it is removed, along with its edges, from the network. States of neighboring agents which list the departee as a buyer or seller are also removed. In order to prevent manipulation of the mechanism,

once a participant has left the process it is not allowed to rejoin. As with all other changes, departures are performed *before* convergence has been reached. This avoids the possibility of producers or consumers leaving after they have committed to buy or sell goods.

7.1.3 Property Changes

We allow for the alteration of each of the following properties: for producers, reserve prices, input-to-output good ratios and production capacities may be changed; for consumers, the consumption values and desired goods quantities may be changed.

7.1.4 Convergence

Convergence in our reconfiguration scenario is determined using a convergence detector agent which operates in a similar fashion to the one used in our original and multi-unit LBP-based supply chain formation models. We briefly restate the procedure for detecting convergence here; a full explanation is provided in Section 4.7. Once the approximate diameter of the graph has been determined, the convergence detector agent activates each agent and the LBP process begins. At each step, each agent reports to the convergence detector agent as to whether the state which it believes holds the lowest cost has changed since the previous step. If no agent reports a change, and the total number of iterations is greater than the approximate graph diameter, then the convergence detector agent terminates the running of LBP and begins the process of allocation.

The key difference in our reconfiguration scenario is that when an agent changes its properties, or a neighbouring agent enters or leaves the network, it reports to the convergence detector agent, which in turn sets the earliest point at which convergence can be called to $s + d$ where s is the current step, and d is the approximate diameter of the graph. This is to allow enough steps for changes in beliefs as a result of the alteration to fully propagate around the network, and is done every time an alteration is made.

As we explain in Section 4.7, although the convergence detector agent is only involved in commencing and terminating the running of LBP, it could still be claimed that the presence of such an agent limits the decentralisation of our approach. However, as with LBP in a non-reconfiguration setting, a participating "coordinator" agent can be used in lieu of a convergence detector agent if a fully decentralised approach is required, at a cost of small increase in the number of messages required to converge. If this agent decides

to leave the process before a convergent state is reached, it passes its responsibilities on to a randomly chosen neighbouring agent.

The procedure for the operation of such an agent in a reconfiguration setting is much the same as is described in Section 4.7, except that agents also report whether a change in their properties has occurred, and whether any neighbouring agents have left or joined the network. If such a change has occurred, the coordinator agent updates the earliest point at which convergence can be called in the same way as is done by the convergence detector agent.

7.1.5 Allocation

Allocations and allocation values are determined in the same way as for the multi-unit LBP case, as explained in Sections 6.1.7 and 6.1.8.

7.1.6 Payments

Payments are performed in an identical manner to the multi-unit case. This is explained in Section 6.1.9.

7.2 Experiments

We perform three sets of experiments for each network, testing LBP's ability to deal with alterations to the structure of each network and to the properties the participants within to various extremes. For all experiments, initial producer reserve prices are drawn from the distribution $U(0, 1)$, input good ratios from the interval $[1, 2]$, capacities from $[4, 5]$ and consumer desired goods from the interval $[2, 3]$. Parameters specific to changes involving new entrants are explained in the following subsections.

7.2.1 New Entrants: Producers

In order to experimentally model the performance of min-sum LBP as producers enter the market, we assign a tier value to each good in the original, unaltered network. Goods produced by producers with no inputs are tier 1 goods, goods produced by producers which consume tier 1 goods are tier 2 goods, and so on. For the purposes of these experiments, when a new producer enters the market, it is assigned an output good drawn from

a uniform distribution of the set of goods in the network. The producer is then assigned an appropriate set of input goods from the tier below its output good. The number of inputs each producer is assigned is drawn randomly from the interval $[1, T_g]$, where T_g is the total number of goods in the appropriate tier. If the producer is assigned a tier 1 output good, it is given no inputs. We assign input goods in this way in order to avoid scenarios where producers are able to bypass multiple echelons of the original supply chain by processing a low-tier good into an output from a much higher tier, which simplifies the problem of determining the optimal allocation, and also to prevent producers from producing outputs of a lower tier than their input goods, which is unrealistic. Producers are also assigned a random reserve price, input good ratios and a capacity, at the values specified in Section 7.2.

7.2.2 New Entrants: Consumers

Consumers are randomly assigned a consumable good from a uniform distribution of the goods in the final tier of the original network. Consumption values for new consumers are set at a value randomly drawn from a uniform distribution of the values plus or minus 10% of the consumption value of consumer C1 in the original network. We use consumption values from this range to try to at least partially preserve the dynamics of the original network. The number of goods new consumers desire is set in the same way as for consumers present at initialisation. The value used for this property can be found in Section 7.2.

7.2.3 Experimental Settings

In the following subsections, we explain the details of each of our three experimental settings.

Setting 1 In the first set of experiments we run, we test LBP's ability to cope with a single incidence of each of the nine possible changes listed in Table 7.1. This scenario tests LBP's ability to cope with minor alterations to the original network. After the change has been made, we recompute the optimal value for the altered network. We perform 100 runs of LBP on each network for each change. In each run, a random step is selected between step 1 and AD, the approximate graph diameter, with equal likelihood for each step, as determined by the convergence detector agent. LBP continues to run until convergence is detected, as is specified in section 7.1.4, or, if convergence is not reached, until the algorithm has run for 250 steps.

Setting 2 In the second set of experiments, we test LBP's performance under a series of random changes. This allows us to assess LBP's performance under more challenging conditions than in Setting 1. The number of changes for each run is drawn from the interval $[2, 5]$ and is varied between runs. The step at which the first change occurs follows the same method as in Setting 1. The steps at which each of the subsequent changes occur are drawn from the interval $[C_{prev}, AD]$, where C_{prev} is the step at which the previous change occured and AD is the approximate graph diameter. These values are recomputed for every run. After all changes have been made, we recompute the optimal value for the altered network. For this and each subsequent set of experiments, once the final change occurs, LBP continues to runs until convergence or until 250 steps have been completed.

Setting 3 Finally, the third set of experiments model a scenario similar to experimental setting 2, but this time allow for the possibility of a larger number of random alterations. This tests LBP's ability to cope under very challenging conditions, with numerous changes being made during the running of the algorithm. In this set of experiments, the number of changes per run is drawn from the interval $U(6, 10)$, and is varied between runs. After all changes have been made, we recompute the optimal value for the altered network.

7.3 Results

7.3.1 Setting 1: Single Change, Single Type

Tables 7.2 and 7.3 show the efficiency classes produced by LBP over 100 runs of each alteration type over each network. Given the absence of a comparable decentralised supply chain reconfiguration technique in the literature, the main basis of comparison with these results is with those of multi-unit LBP *with ratios*, as presented in Chapter 6.

In the Simple network, for which multi-unit LBP with ratios achieved 100% optimality, the results of Table 7.2 suggest that the effect of most alterations is reasonably minor, with at most 4% of these optimal results being converted into zero or negative-valued allocations for eight of the nine alterations we tested. The sole exception to this is when a new consumer is added to the network: in this case, 20% of the original optimal results are lost to zero or negative-valued allocations. Efficiency loss when a new consumer added is a common occurence on many of our networks; this is because the addition of new vertices and edges to the original graph often also leads to the creation of one or

more cycles. As we explain in Section 2.4.2, there are no performance guarantees for LBP on graphs with multiple cycles.

For the Unbalanced network, for which multi-unit LBP with ratios produced optimal results 67% of the time in a non-reconfiguration setting with a relatively large proportion of suboptimal results, we see that reconfiguration actually leads to a slight improvement in the proportion of optimal results in several instances. We speculate that the reason for this might be that certain changes, such as a producer being removed, may lead to the removal of cycles or a widening of the difference in efficiency between optimal and suboptimal solutions, making it easier for LBP to find optimal solutions. We also notice that LBP is able to deal with the removal of the sole consumer in the network and consistently produces optimal allocations (of value zero) for this alteration on this and all other networks with a single consumer.

Results for all other networks - Two-Cons (98% optimal results in the non-reconfiguration multi-unit case), Bigger (94%), Many-Cons (100%), Greedy-Bad (69%) Harder (27%) and Huge (63%) - follow in a similar vein, with most alterations tending not to significantly change the proportion of optimal results produced. Certain alterations, in particular the removal of producers and consumers, appear to consistently improve the results LBP is able to produce for most networks. Others, most notably the addition of producers, tend to mean LBP is able to produce slightly fewer optimal allocations. Again, we attribute improvements in performance brought about by departures of producers to the removal of cycles leading to less frequent double-counting of messages and more accurate agent beliefs. Conversely, new producers and consumers joining the network tend to reduce performance slightly by introducing additional cycles to the networks.

Average efficiency results for experimental setting 1 are presented in Table 7.4, and follow a similar pattern to the efficiency classes results. For all but the Harder network, average efficiencies tend to be roughly similar to the values for multi-unit LBP with ratios in a non-reconfiguration setting, as is shown in Table 6.2. Again, adding a producer seems to have the most pronounced adverse effect, with efficiency loss of around 25% for most networks when this alteration is made.

Table 7.2: Distribution of efficiency classes produced by LBP under experimental setting 1, where a change of a single type occurs once per run.

	Simple network % of instances				Unbalanced network % of instances				Two-Cons network % of instances				Bigger network % of instances			
Change	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Production Cost	0.0	1.0	0.0	99.0	1.0	12.0	22.0	65.0	3.0	0.0	2.0	95.0	4.0	2.0	0.0	94.0
Ratio	1.0	3.0	0.0	96.0	1.0	6.0	14.0	79.0	3.0	0.0	1.0	96.0	4.0	0.0	1.0	95.0
Consumption Value	0.0	0.0	0.0	100.0	0.0	7.0	23.0	70.0	0.0	0.0	4.0	96.0	2.0	0.0	0.0	98.0
Producer Added	0.0	4.0	0.0	96.0	4.0	17.0	34.0	45.0	3.0	0.0	18.0	79.0	6.0	15.0	4.0	75.0
Consumer Added	10.0	1.0	8.0	81.0	4.0	0.0	52.0	44.0	11.0	0.0	9.0	80.0	16.0	0.0	2.0	82.0
Producer Removed	0.0	0.0	0.0	100.0	0.0	10.0	8.0	82.0	0.0	0.0	0.0	100.0	3.0	0.0	0.0	97.0
Consumer Removed	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0
Production Capacity	0.0	1.0	0.0	99.0	1.0	9.0	18.0	72.0	2.0	1.0	0.0	97.0	1.0	2.0	0.0	97.0
Desired Goods	0.0	0.0	0.0	100.0	1.0	5.0	15.0	79.0	6.0	0.0	0.0	94.0	9.0	0.0	0.0	91.0
No Changes	0.0	0.0	0.0	100.0	0.0	10.0	22.0	67.0	0.0	0.0	2.0	98.0	3.0	2.0	1.0	94.0

Table 7.3: Distribution of efficiency classes produced by LBP under experimental setting 1, where a change of a single type occurs once per run.

	Many-Cons network % of instances				Greedy-Bad network % of instances				Harder network % of instances				Huge network % of instances			
Change	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Production Cost	0.0	0.0	0.0	100.0	0.0	14.0	15.0	71.0	44.0	26.0	2.0	28.0	4.0	21.0	12.0	63.0
Ratio	0.0	0.0	0.0	100.0	0.0	4.0	18.0	78.0	46.0	24.0	3.0	27.0	7.0	27.0	7.0	59.0
Consumption Value	0.0	0.0	0.0	100.0	0.0	13.0	10.0	77.0	57.0	12.0	1.0	30.0	8.0	21.0	11.0	60.0
Producer Added	1.0	4.0	9.0	86.0	0.0	18.0	28.0	54.0	52.0	25.0	4.0	19.0	3.0	29.0	9.0	59.0
Consumer Added	0.0	0.0	0.0	100.0	0.0	1.0	28.0	71.0	51.0	17.0	16.0	16.0	3.0	29.0	9.0	59.0
Producer Removed	0.0	0.0	0.0	100.0	0.0	14.0	0.0	86.0	38.0	24.0	5.0	33.0	10.0	17.0	9.0	64.0
Consumer Removed	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	1.0	25.0	1.0	73.0
Production Capacity	0.0	0.0	0.0	100.0	0.0	12.0	20.0	68.0	50.0	22.0	3.0	25.0	8.0	23.0	10.0	59.0
Desired Goods	0.0	0.0	0.0	100.0	0.0	9.0	14.0	77.0	51.0	18.0	3.0	28.0	1.0	27.0	16.0	56.0
No Changes	0.0	0.0	0.0	100.0	0.0	14.0	17.0	69.0	45.0	26.0	2.0	27.0	4.0	22.0	11.0	63.0

Table 7.4: Average efficiency for each type of change produced by LBP in experimental setting 1. A result of 1.000 is equal to the capture of an average of 100% of available efficiency.

	Network							
Change	Simple	Unbalanced	Two-Cons	Bigger	Many-Cons	Greedy-Bad	Harder	Huge
Production Cost	0.996	0.868	0.926	0.767	1.000	0.863	-0.022	0.576
Ratio	0.976	0.860	0.93	0.808	1.000	0.930	-0.310	0.382
Consumption Value	1.000	0.859	0.973	0.904	1.000	0.882	-0.496	0.413
Producer Added	0.954	0.709	0.812	0.752	0.886	0.824	-0.556	0.576
Consumer Added	0.747	0.720	0.724	0.670	1.000	0.917	0.19	0.581
Producer Removed	1.000	0.855	1.000	0.857	1.000	0.824	-0.227	0.308
Consumer Removed	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.621
Production Capacity	0.985	0.838	0.966	0.932	1.000	0.845	-0.715	0.425
Desired Goods	1.000	0.909	0.871	0.691	1.000	0.908	-0.466	0.592
No Changes	1.000	0.872	0.983	0.813	1.000	0.839	-0.058	0.583

7.3.2 Setting 2: Multiple Changes, Multiple Types

Table 7.5 shows the efficiency classes produced in experimental setting 2, where randomly chosen alterations occur between 2 and 5 times per run. Because removing a consumer leads to 100% optimality in most networks, and negates the effect of further shocks, we do not allow this alteration to be chosen as a random shock in this setting or in experimental setting 3. A set of 100 runs were performed for each network. Table 7.6 shows the average efficiency produced by LBP in this setting. It is clear from these results that multiple alterations have little effect on LBP's performance, with only very slight degradations in total optimality for most networks. Table 7.7 shows that spread within our results is only slightly greater than for the static multi-unit case.

Table 7.5: Distribution of efficiency classes produced by LBP under experimental setting 2, where randomly chosen changes occur between 2 and 5 times per run. Results from multi-unit LBP with ratios where no changes take place are included for reference.

	Dynamic LBP % of instances				Static LBP % of instances			
Network	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Simple	1.0	1.0	1.0	97.0	0.0	0.0	0.0	100.0
Unbalanced	1.0	10.0	28.0	61.0	0.0	10.0	22.0	67.0
Two Cons	5.0	0.0	5.0	90.0	0.0	0.0	2.0	98.0
Bigger	13.0	4.0	2.0	81.0	3.0	2.0	1.0	94.0
Many-Cons	2.0	2.0	2.0	94.0	0.0	0.0	0.0	100.0
Greedy-Bad	0.0	11.0	21.0	68.0	0.0	14.0	17.0	69.0
Harder	53.0	18.0	6.0	23.0	45.0	26.0	2.0	27.0
Huge	7.0	16.0	16.0	60.0	4.0	22.0	11.0	63.0

Table 7.6: Average efficiency for each type of change produced by LBP in experimental setting 2. A result of 1.000 is equal to the capture of an average of 100% of available efficiency. Results from multi-unit LBP with ratios where no changes take place are included for reference.

Network	Dynamic LBP average efficiency *Setting 2*	Static LBP average efficiency
Simple	0.954	1.000
Unbalanced	0.806	0.872
Two Cons	0.822	0.983
Bigger	0.528	0.813
Many-Cons	0.940	1.000
Greedy-Bad	0.818	0.839
Harder	-0.118	-0.058
Huge	0.511	0.583

Table 7.7: Median and interquartile range measures for the average efficiency produced in Setting 2.

Network	Setting 2 LBP median	Setting 2 LBP interquartile range
Simple	1.000	0.000
Unbalanced	1.000	0.216
Two-Cons	1.000	0.000
Bigger	1.000	0.000
Many-Cons	1.000	0.000
Greedy-Bad	1.000	0.018
Harder	-0.125	1.770
Huge	1.000	1.032

7.3.3 Setting 3: Many Changes, Multiple Types

Table 7.8 shows the efficiency classes produced in experimental setting 3. In this setting, randomly chosen alterations occur between 6 and 10 times per run, simulating a supply chain formation scenario requiring a great deal of adaptation by the mechanism. We allow for all of the alterations explained in Section 7.1 with the exception of the departure

of consumers. As previously stated, this is because the departure of a consumer reliably leads to 100% optimality in most networks. When compared to the results of experimental setting 2, we see that LBP's performance degrades gracefully with a larger number of changes per run. For some networks, the results appear to imply that performance is actually better in this setting. We suggest this is a statistical quirk and would not be borne out if more runs were conducted. Total optimality and average efficiency are very slightly worse that experimental setting 2 and, for most networks, are roughly comparable to the results produced when no reconfiguration is performed at all. As with setting 2, Table 7.10 shows relatively little increase in the spread of our results as compared to the static multi-unit case.

7.3.4 Game Theoretic Properties

In this section, we explain how the game-theoretic properties of our LBP-based approach are altered by allowing participants to change their properties, or to leave the process at any point before the allocation is determined.

7.3.4.1 Individual Rationality

As with standard single-unit LBP, LBP_{μ} and multi-unit LBP, we cannot guarantee individual rationality for LBP with reconfiguration. Allowing agents to leave the process at any point before the allocation is determined grants them greater ability to avoid negative utility, but we are unable to guarantee non-negative utility for all agents that do not choose to leave the process.

7.3.4.2 Incentive Compatibility

Although we allow producers to update their reserve prices in a similar way to LBP_{μ}, margin update steps are not enforced and producers may choose to change their margins by any value. Because of this, incentive compatibility cannot be guaranteed.

7.3.4.3 Budget Balance

Our approach continues to involve no payments to or from the mechanism, and is therefore strongly budget balanced.

Table 7.8: Distribution of efficiency classes produced by LBP under experimental setting 3, where randomly chosen changes occur between 6 and 10 times per run. Results from setting 2 and static multi-unit LBP are provided for comparison.

	Dynamic LBP *Setting 3* % of instances				Dynamic LBP *Setting 2* % of instances				Static LBP % of instances			
Network	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt	Neg	Zero	Sub	Opt
Simple	2.0	0.0	3.0	95.0	1.0	1.0	1.0	97.0	0.0	0.0	0.0	100.0
Unbalanced	3.0	11.0	29.0	57.0	1.0	10.0	28.0	61.0	0.0	10.0	22.0	67.0
Two Cons	4.0	1.0	8.0	87.0	5.0	0.0	5.0	90.0	0.0	0.0	2.0	98.0
Bigger	14.0	5.0	7.0	74.0	13.0	4.0	2.0	81.0	3.0	2.0	1.0	94.0
Many-Cons	0.0	2.0	2.0	96.0	2.0	2.0	2.0	94.0	0.0	0.0	0.0	100.0
Greedy-Bad	1.0	17.0	13.0	69.0	0.0	11.0	21.0	68.0	0.0	14.0	17.0	69.0
Harder	55.0	21.0	5.0	19.0	53.0	18.0	6.0	23.0	45.0	26.0	2.0	27.0
Huge	20.0	14.0	20.0	46.0	7.0	16.0	16.0	60.0	4.0	22.0	11.0	63.0

Table 7.9: Average efficiency for each type of change produced by LBP in experimental setting 3. A result of 1.000 is equal to the capture of an average of 100% of available efficiency. A comparison with setting 2 is also shown. Results from setting 2 and static multi-unit LBP are provided for comparison.

Network	Dynamic LBP average efficiency *Setting 3*	Dynamic LBP average efficiency *Setting 2*	Static LBP average efficiency
Simple	0.911	0.954	1.000
Unbalanced	0.713	0.806	0.872
Two Cons	0.801	0.822	0.983
Bigger	0.520	0.528	0.813
Many-Cons	0.989	0.940	1.000
Greedy-Bad	0.793	0.818	0.839
Harder	-0.074	-0.118	-0.058
Huge	0.379	0.511	0.583

7.3.4.4 Allocative Efficiency

The allocative efficiency of LBP in a reconfiguration environment tends to degrade gracefully as the number of changes occurring in each run increases. In instances where only one type of change occurs, the degradation from the results of a non-reconfiguration multi-unit environment depends on the type of change that occurs.

7.4 Conclusions

In this chapter, we experimentally evaluated LBP's performance in a setting where the algorithm is forced to adapt to changes in the properties or composition of participating agents. We performed three sets of experiments: one in which we measure the extent to which performance is affected by each type of change, a second measuring performance when multiple random changes occur in each run, and finally a scenario where a larger number of random changes occur in each run. Our results suggest that LBP is capable of dealing with changes to the properties of participants with almost no efficiency loss. Some compositional changes, such as producers and consumers entering the network, tend to produce a slight loss of efficiency, while producers and consumers leaving the network have the opposite effect, with average efficiency tending to improve when this occurs. Our experiments with a series of random changes indicate that the performance of LBP degrades gracefully as the amount of changes increases, allowing us to draw

Table 7.10: Median and interquartile range measures for the average efficiency produced in Setting 3.

Network	Setting 3 LBP median	Setting 3 LBP interquartile range
Simple	1.000	0.000
Unbalanced	1.000	0.000
Two-Cons	1.000	0.000
Bigger	1.000	0.000
Many-Cons	1.000	0.000
Greedy-Bad	1.000	0.000
Harder	-0.083	1.556
Huge	0.891	1.222

the conclusion that, as long as it is given sufficient additional time to converge, LBP is generally robust in the face of on-the-fly alterations to the problem scenario.

Chapter 8

Conclusions and Future Work

Automated support for supply chain formation holds many potential benefits for businesses, including faster and more rational selection of trading partners, increased adaptability and cost savings.

In this thesis, we have made the following contributions to the area of automated supply chain formation:

- We developed a novel, non-market-based technique for the decentralised supply chain formation problem based upon the min-sum loopy belief propagation algorithm.
- We applied our LBP-based technique to several different classes of the decentralised supply chain formation problem, including two classes, profit maximisation and dynamic supply chain formation, which have received very little or no attention in the multi-agent systems literature.
- We presented experimental analysis of our technique, and demonstrated that our technique equals the existing state-of-the-art decentralised techniques for the majority of problem instances studied, and is often capable of producing better solutions.

In Section 8.1, we summarise the contributions made in this thesis in greater detail, and in Section 8.1.1 we discuss potential directions for future work.

8.1 Summary of Contributions

We produced a novel technique for the decentralised supply chain formation problem based upon the min-sum loopy belief propagation algorithm. Loopy belief propagation and its min-sum variant have been applied successfully to many varying problems in AI and computer science in general, but have heretofore not seen use for supply chain formation, a problem which is typically dealt with using market-based protocols. We argued that min-sum LBP presents a number of advantages over market-based protocols and is well-suited as an approach to decentralised supply chain formation. In order to apply min-sum LBP to this problem, we describe in Chapter 3 how we converted the task dependency network formulation for supply chains presented in the existing state of the art decentralised approach [Walsh and Wellman, 2003] into a Markov random field formulation which preserves the important features of the original task dependency networks while being suitable for use with the min-sum algorithm.

We applied the min-sum algorithm to a decentralised supply chain formation setting identical to the scenario investigated in Walsh and Wellman [2003] in Chapter 4. The work presented in this section of the thesis was originally published in shorter form in Winsper and Chli [2010]. We presented a formalism for representing the states of agents in supply chain networks to allow for compatibility with LBP, and explained our approach to casting the monetary costs and values of agents typically used in market-based methods into cost values suitable for use with LBP. We experimentally tested our technique and the two auction protocols Walsh and Wellman [2003] over twelve different problem instances. The first of these auction protocols, SAMP-SB, is a decentralised auction protocol which provided the main basis of comparison to our results. The second, SAMP-SB-D, is a modification of SAMP-SB that conducts a stage of post-allocation decommitment by producers committed to unprofitable contracts. This post-allocation decommitment stage is not allowed for in our model. As such, SAMP-SB-D does not represent a fair point of comparison, but is nevertheless included as an indication of the best-case performance of a decentralised auction protocol for this problem. Our technique was able to match or outperform SAMP-SB for eleven of the twelve problem instances studied, and match or outperform the performance of SAMP-SB-D on ten of the twelve problem instances. The disparity in results was particularly strong on problem instances where the costs of producers did not permit competitive equilibrium - the results produced by the auction protocols for these problems instances were very poor, while LBP, as a non-market-based approach, was unaffected. Our method also required fewer exchanges of information to produce a solution than the auction protocols in eleven of

the twelve problem instances tested.

A novel solution goal for supply chain formation, profit maximisation, was introduced in Chapter 5. We proposed this goal as a means to provide further incentive for producers to participate in a supply chain formation mechanism. Many approaches from the literature, including Walsh and Wellman [2003], assume that producers are content to recoup their costs and participate without making a profit. The difference between the prices consumers are willing to pay and the price they are charged for their consumable good is referred to as the surplus. Our technique for profit maximisation in supply chain formation, LBP_μ, allows producers to convert a proportion of this surplus into profit for themselves by altering their margins at designated iterations of the algorithm on the basis of the belief values they hold about each of their states. We proposed a rational margin update strategy for use in LBP_μ, the parameters of which were experimentally tuned to provide the best possible balance between available surplus and producer profits. We conducted a series of experiments to determine the performance of LBP_μ and our rational pricing strategy as a means for profit maximisation. Because LBP_μ is the first instance of a technique for profit maximisation in supply chain formation, we introduced a naive LBP-based profit maximisation technique to serve as a basis for comparison. LBP_μ allowed producers to convert more surplus into profit for all of the twelve network instances studied, and was able to produce solutions with greater available surplus, a secondary objective for this solution goal, for ten of twelve network instances. LBP_μ required a significantly greater number of information exchanges than the naive method, but this was expected given the difference in approaches.

In Chapter 6 we extended our model to the multi-unit case. This required us to propose a framework for the representation of multi-unit supply chains, which took into account multi-unit specific factors including production capacities, input-to-output good ratios and consumer desired goods. It also required extension of the states of agents to model quantities of goods. To our knowledge, this work was the first instance of a technique for decentralised multi-unit supply chain formation. Applying our model to the multi-unit case allowed us demonstrate the ability of the LBP algorithm to scale up to more complex supply chain formation problems. It also brought the problem domain studied into line with the multi-unit scenarios examined using state-of-the-art centralised approaches. In order to provide a decentralised point of comparison to our approach, we extended the auction protocols of Walsh and Wellman [2003] to the multi-unit case. Because our extension of Walsh and Wellman [2003] did not permit the use of input to output good ratios, we tested two version of multi-unit LBP - one where ratios are set for each pro-

ducer to a randomly chosen integer value between 1 and 2, and another version to allow comparison to SAMP-SB in which all ratios are set to 1. Our results suggested that the comparable version of LBP slightly outperformed SAMP-SB, producing equal or greater efficiency for five of the eight network instances tested. Results for LBP with random ratios were only slightly less efficient than the version with ratios set to 1. LBP required fewer instances of information exchange than the auction protocols over all of the eight network instances.

Chapter 7 introduced a supply chain reconfiguration scenario which granted participants the freedom to change their properties or to enter or leave the supply chain formation process at any point before the allocation is determined. This scenario allowed us to investigate the performance of LBP in an environment which grants participants greater autonomy, and again represented the first instance of a decentralised approach to this problem. We classified the types of changes available to participants into two broad categories: property changes and structural changes. We performed a series of experiments investigating the performance of LBP when faced with one instance of each type of change per run, when faced with a relatively small number of randomly chosen changes, and when a large number of randomly chosen changes are made in each run. Our results suggested that LBP's performance is largely unaffected by property changes. This is expected behaviour given the relatively small difference in efficiency between standard LBP and LBP_{μ}, which also allows producers to modify their properties. Performance degrades slightly from the static multi-unit scenario for structural changes which involve the entrance of new producers or consumers - these changes create a more difficult problem for the algorithm. Conversely, departures by producers or consumers, which make the problem easier, allowed LBP to produce better results than in the static scenario. Performance degraded gracefully when larger numbers of alterations were made in each run.

8.1.1 Future Work

Our LBP-based models for supply chain formation were able to produce excellent results in each of the classes of the supply chain formation problem we applied them to. There were, nevertheless, some network instances for which LBP was unable to consistently reach a convergent state. In these cases, the algorithm tends to oscillate between several different solutions, and the actual solution chosen is merely the one in which the algorithm rests once it has run for the pre-designated number of steps. Performance, in terms of numbers of messages sent, could be improved in this situation if producers were able

to report an oscillation in their beliefs to the convergence detector agent, which could call convergence early. This does, however, raise the question of which phase of the oscillation the algorithm should be terminated at, given the convergence detector agent's lack of information about the beliefs and states of participants or the structure of the network. Such a decision could be made in an informed manner if the convergence detector agent had more knowledge of the properties of the network, at the cost of introducing a centralised element to the algorithm.

An alternative method for improving efficiency results would be to implement a phase of post-allocation decommitment similar to that which is employed by the SAMP-SB-D auction protocol we use as a point of comparison. Post-allocation decommitment increases the efficiency of allocations in which dead ends are present by allowing producers who were unable to produce or find a buyer for their output good to decommit from contracts to purchase inputs. This removes the possibility that producers might make a loss by participating, and in doing so ensures the individual rationality of the mechanism. However, as discussed in Walsh and Wellman [2003], post-allocation decommitment has several drawbacks. It compromises the utility of participants affected by decommitments, who may in turn be forced to decommit themselves, and introduces the possibility that producers might choose to decommit even when not part of a dead end if, for example, the amount of profit they would make is not as large as desired.

It is possible that the efficiency results of LBP_μ could be improved by allowing for the possibility of backtracking to global state configurations with greater efficiency during the running of our algorithm. This would, in particular, potentially improve the efficiency results produced by LBP_μ for the no-CE case of Two-Cons, in which the double counting of messages frequently leads to zero-valued allocations. However, the pursuit of a more globally efficient allocation raises the possibility of decreasing profit for some producers, compromising their self-interest. Future work could include an experimental analysis of the strategic implications of introducing the ability for producers to alter their β values during the running of the algorithm.

The extensions made in Chapters 6 and 7 give our model greater fidelity to the considerations of real-world supply chain formation than existing decentralised techniques, which tend to model scenarios in which goods are exchanged in single units, where factors such as capacities are not taken into account, and the potential for new entrants, departures and property changes are not modelled. Further extensions have the potential to increase the fidelity of the model to a point where industrial application may become a possibility.

We suggest that the most important next steps in enriching the model are extensions to

take into account scheduling factors such as delivery date commitments, lateness penalties and storage costs. Implementation of these extensions would bring the fidelity of the scenario to a level commensurate with that of the TAC SCM game [Collins et al., 2006], which is currently the most realistic decentralised agent-based supply chain-related scenario. These extensions would require further improvements to the expressive power of states, and have the potential to require significant expansion of the number of states required by each agent, slowing the speed of the algorithm. This problem could be mitigated by limiting the time horizons modelled in the states of each agent, and allowing agents to prune states which specify dates or other constraints which are incompatible with the states of their neighbours.

Finally, this thesis has focused exclusively on the use of techniques based on the min-sum loopy belief propagation algorithm. We believe there exists scope for the application of alternative algorithms for graphical inference to this problem if issues regarding centralisation and modification of the original network structure can be reconciled.

Appendix A

Two Cons Network Example

Our linear models are written in accordance with the .lp file format, the native format of lp_solve. We begin each linear model by defining the objective function, the variable we wish to maximise. In the Two Cons network, as with all other networks, the objective function is the sum of the values gained by consumers. For a complete version of the Two Cons example model, please see the following section.

```
1 max: TotalVal;
```

The value of the objective function produced by MILP is used as the optimal benchmark against which the values produced by LBP are compared. If lp_solve returns a value of 0 for the objective function, there exists no positive-valued solution to that network given the supplied values for production costs, capacities, ratios and consumer desired good quantities. In this case, we discard the network instance, compute new values for the aforementioned variables, and run LP again using the new values.

Lines 3-7 of our Two Cons model assign values to variables specifying the production capacities of each of the producers in our network. We use Java to write these values directly into the model before every run.

```
3 P1Cap = 4;
...
7 P5Cap = 3;
```

Lines 9-10 specify the number of goods desired by each consumer in the network instance. Again, these are supplied to the model by Java.

```
9 C1Desired = 3;
10 C2Desired = 2;
```

Lines 12-16 of the Two Cons model specify equalities between the total number of goods produced by each producer and the number of goods sold to each of the buyers of their output good. For example, line 12 specifies that producer P1's total output (P1Prod) is equal to the sum of P1's sales to P3 (P1P3) and P4 (P1P4).

```
12 P1Prod = P1P3+P1P4;
...
16 P5Prod = P5C1;
```

Lines 18-22 of the model impose constraints on the total output of each producer according to the production capacities defined in lines 3-7.

```
18 P1Prod <= P1Cap;
...
22 P5PRod <= P5Cap;
```

Lines 24-27 encode the relationship between each producer's set of input goods ratios and the number of each of these goods they need to consume in order to produce their output good. Line 24 specifies that P3 needs to acquire twice as many of its sole input good, sold by producer P1, as the number of goods it produces and sells to the sole consumer of its output good, C1.

```
24 2 P3C1 = P1P3;
...
27 3 P5C1 = P2P5;
```

In lines 29-33, we define the relationship between the number of goods bought by each consumer from each of their possible supplies and the total number of purchases made by each consumer.

```
29 C1PurchasesFromP3 = P3C1;
...
33 C2TotalPurchases = C2PurchasesFromP4;
```

Lines 34-35 specify an upper limit on the total number of purchases made by each consumer; no consumer may purchase more goods than their desired total.

```
34 C1TotalPurchases <= C1Desired;
```

Line 37 assigns a variable to the product of each producers production cost and total number of goods they produce, while line 38 assigns a variable to the product of each consumer's consumption value and the number of consumable goods they acquire. Production costs and consumption values are supplied to the model by Java.

```
37 SumOfPCs = 0.5 P1Prod + 0.6 P2Prod + 0.1 P3Prod +
0.2 P4Prod + 0.3 P5Prod;
38 TotalVal = 1.23 C1TotalPurchases + 2.17 C2TotalPurchases;
```

Line 40 defines how the objective function is to be calculated. The value of the allocation is equal to the sum of the total value obtained by consumers minus the sum of production costs incurred by active producers.

```
40 TotalVal = ConVals – SumOfPCs;
```

Finally, line 41 imposes an integer value restriction on each the exchange relationships in the network. We impose this restriction due to our assumption that all goods are non-divisible.

```
41 int P1P3, P1P4, P2P4, P2P5, P3C1, P4C2, P5C1;
```

Complete Two-Cons .lp formulation

Below is the Two-Cons Network MIP formulation in .lp format.

```
max: TotalVal;

P1Cap = 1;
P2Cap = 1;
P3Cap = 1;
P4Cap = 1;
P5Cap = 1;

C1Desired = 1;
C2Desired = 1;

P1Prod = P1P3+P1P4;
P2Prod = P2P4+P2P5;
P3Prod = P3C1;
P4Prod = P4C2;
P5Prod = P5C1;

P1Prod <= P1Cap;
P2Prod <= P2Cap;
P3Prod <= P3Cap;
P4Prod <= P4Cap;
P5Prod <= P5Cap;

1 P3C1 = P1P3;
1 P4C2 = P1P4;
1 P4C2 = P2P4;
1 P5C1 = P2P5;

C1PurchasesFromP3 = P3C1;
C1PurchasesFromP5 = P5C1;
C2PurchasesFromP4 = P4C2;
C1TotalPurchases = C1PurchasesFromP3+C1PurchasesFromP5;
C2TotalPurchases = C2PurchasesFromP4;
```

```
34 C1TotalPurchases <= C1Desired;
35 C2TotalPurchases <= C2Desired;
36
37 SumOfPCs = 0.5 P1Prod + 0.6 P2Prod + 0.1 P3Prod
 + 0.2 P4Prod + 0.3 P5Prod;
38 ConVals = 1.23 C1TotalPurchases + 2.17 C2TotalPurchases;
39
40 TotalVal = ConVals - SumOfPCs;
41 int P1P3, P1P4, P2P4, P2P5, P3C1, P4C2, P5C1;
```

Appendix B

Graph Diameter Algorithm

The distributed depth-first search algorithm proceeds as follows: upon initialization of the network, the convergence detector agent designates a random agent as the root node. This agent then randomly picks a neighboring agent and adds it to the candidate spanning tree. The updated candidate spanning tree is sent to the chosen agent, and this agent then randomly chooses one of its own neighbours which is not part of the candidate spanning tree. It then updates the tree and passes control to the chosen agent, with the process continuing until the chosen agent has no neighbors which are not currently part of the candidate spanning tree. In this situation, the active agent backtracks, passing control back to the agent which originally activated it. This agent then chooses another of its own neighbors which is not part of the candidate spanning tree. This process continues until control is passed back to the root node and the root node has no unexplored edges.

With each node aware of who its neighbors are in the final spanning tree, we use a series of distributed breadth first searches to find the diameter of the tree. The convergence detector designates one agent randomly as the root node. This node sends a message to each of its neighbors in the spanning tree informing them that they are one level away from the root. These agents then send a message to each of their neighbors from which they have not already received a message indicating that they are two levels from the root, and so on. Once an agent has sent messages to each of its neighbours, it sends a message back to the agent which activated it, indicating its current level. This value is passed backwards through the tree until it reaches the root node. The root node then sends the maximum of these values, equal to the maximum shortest path between the root and any other node in the spanning tree, to the convergence detector agent. The convergence detector agent then assigns another agent as the root node, and the process is repeated until the diameter of the tree is determined.

Appendix C

Notation

Symbol	Meaning
β_p	Producer p's buffer zone
δ_p	The maximum increment by which producer p may increase or decrease its margin
μ_p	Producer p's margin
A_c	The number of units of goods acquired by consumer c
AD	The approximate graph diameter
bel_{pI}	Producer p's belief in its inactive state
bel_{pA}	Producer p's belief in its lowest cost active state
$bel_u(x_u)$	Agent u's belief about it's state x_u
C	The set of consumers in a supply chain network.
$C*$	The set of consumers in the optimal allocation
c	A consumer in C
C_{prev}	The last step at which a change occured in the supply chain reconfiguration scenario

Symbol	Meaning
F_{bp}	The fee paid from buyer b to producer p
F_{pi}	The fee paid by producer p to each of its suppliers i
$f_v(x_v)$	The unary cost of agent v being in state x_v
$g_{uv}(x_u, x_v)$	The pairwise cost of linked agents u and v being in states x_u and x_v
M_p	The total number of units of goods manufactured by producer p
M_{pb}	The number of units of goods manufactured by producer p for buyer b
$m_{w \to u}(x_u)$	A message sent from neighbouring agent w to agent u
	encoding w's beliefs about u's state x_u
N_u	The neighbours of agent u
P	The set of producers in a supply chain network.
$P*$	The set of producers in the optimal allocation
p	A producer in P
R_p	The reserve price of producer p
T_g	The total number of goods in tier T of a supply chain network
V	The set of agents in a network
v	An agent in V
V_c	The consumption value of consumer c
w	A neighbour in N

References

M. Babaioff and N. Nisan. Concurrent auctions across the supply chain. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, EC '01, pages 1–10, New York, NY, USA, 2001. ACM. ISBN 1-58113-387-1. 20, 65

M. Babaioff and W.E. Walsh. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. In *ACM Conference on Electronic Commerce*, 2003. 19, 20

J. Cerquides, U. Endriss, A Giovannucci, and J.A. Rodríguez-Aguilar. Bidding Languages and Winner Determination for Mixed Multi-Unit Combinatorial Auctions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 469–476, 2007. 14, 21

J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson. The supply chain management game for the 2007 trading agent competition, 2006. 17, 22, 122

C. Crick and P. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2003. 23, 25, 39

R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983. 18

E.W. Dijkstra and C.S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, 1980. 47

A. Farinelli, A. Rogers, A. Petcu, and N.R. Jennings. Decentralized 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*, 2008. 25

P.F. Felzenszwalb and D.R. Huttenlocher. Efficient belief propagation for early vision. In *Computer Vision and Pattern Recognition*, volume 1, pages 261–268, 2004. 25

M.S Fox, M. Barbuceanu, and R. Teigen. Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems*, 12:165–188, 2000. 22

B. Frey and D.J.C. MacKay. A Revolution: Belief Propagation in Graphs With Cycles. In *Neural Information Processing Systems*, pages 479–485. MIT Press, 1998. 25

A. Giovannucci, M. Vinyals, and J.A. Rodriguez-Aguilar. Computationally-efficient winner determination for mixed multi-unit combinatorial auctions, 2008. 21

M. He, N.R. Jennings, and H. Leung. On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering*, 15:985–1003, 2003. 19

H.S. Kim and J.H. Cho. Supply chain formation using agent negotiation. *Decision Support Systems*, 49(1):77 – 90, 2010. 18

K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI*, pages 56–61, 2000. 21

D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*, volume 1. Cambridge University Press, 1st edition, 2003. 24

C. Magnien, M. Latapy, and M. Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *J. Exp. Algorithmics*, 13:10:1.10–10:1.9, 2009. 47, 48

R.J. McEliece, D.J.C. MacKay, and Cheng Jung-Fu. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998. 25

P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1997. 47

R. Myerson and M. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, April 1983. 19

F. Nachira, P. Dini, M. Nicolai, A. Le Louarn, and L. Rivera Lèon. *Digital Business Ecosystems*, volume 1. European Commission, 1st edition, 2007. 11

B. Ottens and U. Endriss. Comparing winner determination algorithms for mixed multi-unit combinatorial auctions. In *Proceedings of the 7th International Joint conference on Autonomous Agents and Multiagent systems*, AAMAS '08, pages 1601–1604, 2008. 21

D. Pardoe and P. Stone. Tactex-05: a champion supply chain management agent. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1489–1494. AAAI Press, 2006. 17, 22

S. Parsons, J.A. Rodriguez-Aguilar, and M. Klein. Auctions and bidding: A guide for computer scientists. *ACM Comput. Surv.*, 43(2):1–59, 2011. 19

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, volume 1. Morgan Kaufmann, 1st edition, 1988. 13, 23

T. Sandholm. Expressiveness in mechanisms and its relation to efficiency. In *Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC)*, 2008. 12

M. Vinyals, J. Cerquides, A. Farinelli, and J.A. Rodriguez-Aguilar. Worst-case bounds on the quality of max-product fixed-points. In *Advances in Neural Information Processing Systems*, pages 2325–2333, 2010. 24

T. Voice, R. Stranders, A. Rogers, and N.R. Jennings. A hybrid continuous max-sum algorithm for decentralised coordination. In *ECAI 2010, 19th European Conference on Artificial Intelligence*, pages 61–66, 2010. 23, 39

W.E. Walsh and M.P. Wellman. Modeling supply chain formation in multiagent systems. In *proceedings of Agent-Mediated Electronic Commerce (AMEC)*, 1999. 17, 31, 35

W.E. Walsh and M.P. Wellman. Decentralized Supply Chain Formation: A Market Protocol and Competitive Equilibrium Analysis. *Journal of Artificial Intelligence Research*, 19:513–567, 2003. 11, 12, 19, 20, 21, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 46, 50, 51, 53, 54, 56, 58, 63, 64, 69, 71, 84, 86, 87, 92, 93, 96, 99, 100, 118, 119, 121

W.E. Walsh, M.P. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *Second ACM Conference on Electronic Commerce*, pages 260–269, 2000. 19, 21

M. Wang, H. Wang, D. Vogel, K. Kumar, and D.K.W. Chiu. Agent-based negotiation and decision making for dynamic supply chain formation. *Engineering Applications of Artificial Intelligence*, 22(7):1046–1055, 2006. 18

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000. 24

M. Wellman and W. Walsh. Distributed quiescence detection in multiagent negotiation. In *AAAI-99 Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, pages 317–324, 2000. 50

M.P. Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. *Computational Intelligence*, 21:2005, 2003. 17

M.P Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. *Computational Intelligence*, 21:1–26, 2005. 22

M. Winsper and M. Chli. Decentralised Supply Chain Formation: A Belief Propagation-based Approach. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 1125–1126, 2010. 14, 38, 118

M. Winsper and M. Chli. Using the max-sum algorithm for profit maximisation in supply chain formation. *Journal of Autonomous Agents and Multiagent Systems*, Under Review, 2012a. 14

M. Winsper and M. Chli. Using the max-sum algorithm for supply chain formation in dynamic multi-unit environments. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2012b. 15

M. Winsper and M. Chli. Using the max-sum algorithm for supply chain formation in dynamic multi-unit environments. *Journal of Autonomous Agents and Multiagent Systems*, Under Review, 2012c. 15

M. Winsper and M. Chli. Decentralized supply chain formation using max-sum loopy belief propagation. *Computational Intelligence*, In Press, 2012d. 14, 38

M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009. ISBN 0470519460, 9780470519462. 17