Computational Intelligence, Volume 27, Number 000, 2011

Decentralized Supply Chain Formation using Max-Sum Loopy Belief Propagation

MICHAEL WINSPER AND MARIA CHLI

Computer Science, Aston University, Birmingham, UK

Supply chain formation is the process by which a set of producers within a network determine the subset of these producers able to form a chain to supply goods to one or more consumers at the lowest cost. This problem has been tackled in a number of ways, including auctions, negotiations, and argumentation-based approaches. In this paper we show how this problem can be cast as an optimization of a pairwise cost function. Optimizing this class of energy functions is NP-hard (Boykov *et al.,* 2001) but efficient approximations to the global minimum can be obtained using loopy belief propagation (LBP). Here we detail a max-sum LBP-based approach to the supply chain formation problem, involving decentralized message-passing between supply chain participants. Our approach is evaluated against a well-known decentralized double-auction method and an optimal centralized technique, showing several improvements on the auction method: it obtains better solutions for most network instances which allow for competitive equilibrium[†] while also optimally solving problems where no competitive equilibrium exists, for which the double-auction method frequently produces inefficient solutions.

Key words: Supply chain formation; Max-sum algorithm; Loopy belief propagation.

1. INTRODUCTION

As the drive for efficiency and adaptability becomes an increasing focus in industry, together with rising levels of uncertainty about market conditions, the ability to quickly form effective, mutually beneficial trading partnerships becomes increasingly important. Although the concept of virtual enterprises - ad-hoc coalitions of businesses formed to pool resources and create synergies in order to respond to emergent business opportunities - may have yet to reach the level of popularity its proponents had hoped for, the principles of such arrangements, along with outsourcing, remain integral to the business processes of many organizations (Nachira *et al.*, 2007).

Traditional non-computational approaches to supply chain formation are inefficient processes, with time wasted on contract tendering and negotiations. Time constraints and human irrationality may lead to the establishment of inefficient supply chains, 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.

Agent-based computational approaches to supply chain formation model potential supply chain participants - businesses capable of forming a link in the yet-to-be-completed chain - as boundedly-rational self-interested computational agents. These agents deliberate between themselves, typically either through negotiations or auctions, about the subset of agents capable of forming the most efficient supply chain. At the conclusion of these de-

[†]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.

[‡] Address correspondence to Michael Winsper, Computer Science, Aston University, Birmingham, B4 7ET, UK; email: winsperm@aston.ac.uk

liberations, which are typically completed in a fraction of the time required of the manual approach, the supply chain is formed instantly.

Agent-based approaches to SCF may either be centralized or decentralized. Centralized approaches typically make use of combinatorial auctions to determine allocations, with the NP-hard winner determination problem usually being solved by using integer programming. The use of integer programming implies complete knowledge of the bids of all agents, meaning that it relies on an assumption of centralization which may not always present in the real world. *red*Centralized approaches also encounter problems with scalability when applied to larger problem instances. Decentralized approaches to the SCF problem make only minimal assumptions about the participating agents, giving them a wide range of applicability, but present a difficult problem in determining the optimal allocation, given that agents only possess local information about the structure of the network and the capabilities of other agents.

We suggest that LBP is a useful technique for the supply chain formation problem for a number of reasons. The algorithm is distributed, meaning that the scalability issues present in integer programming-based approaches, such as combinatorial auctions and MDPs, are avoided. LBP is also decentralized, with participants acting only on the basis of local information. This presents an advantage over other techniques for graphical inference 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 article, building on previous work in Winsper and Chli (2010), we propose a max-sum loopy belief propagation (LBP) based approach to decentralized supply chain formation which is capable of producing efficient results over a range of network topologies. With our use of LBP, we are frequently able to produce results comparable to that of a centralized approach while working in a distributed and decentralized manner. The use of this message passing-based technique also allows us to take full advantage of the graphical structure of our networks.

In section 2, we provide details of previous models of decentralized supply chain formation, and explain why LBP is a useful approach for supply chain formation when networks are represented graphically. In section 3, we provide details of our model, inspired by work previously conducted in Walsh and Wellman (2003), and provide the details of how we applied the LBP algorithm to the decentralized supply chain formation problem. Section 4 describes our experimental procedure, while section 5 shows our results and compares them to the results obtained by Walsh and Wellman (2003) and the optimal centralized method. Section 6 provides some conclusions about our work, while section 7 identifies areas of related future research.

2. BACKGROUND

2.1. Supply Chain Formation

Multi-agent systems enable us to model a number of properties characteristic of supply chains, including uncertainty, decentralized decision making by self-interested agents and the process of self-organization 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 decentralized supply chain formation and related work can be classified into one of 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.

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. While the usefulness of the standard Contract Net protocol to supply chain formation is limited by its myopic, greedy approach and subsequent inability to deal with resource contention, a number of other negotiation-based approaches have been applied to the supply chain formation problem. Wang *et al.* (2006) uses argumentation-based negotiation for decision making in supply chain formation, while Kim and Cho (2010) proposes 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, outperforming branch-and-bound search. 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 centralization which precludes the application of these methods in areas where this assumption is not valid.

The other main approach to supply chain formation involves modeling the supply chain as a network of 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, many auctions possess a number of interesting game-theoretic properties such as incentive compatibility and individual rationality, and auctions are often able to form satisficing solutions to the supply chain formation problem.

Perhaps the most comprehensive series of studies on supply chain formation using auctions comes from Walsh *et al.*, 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).

In Walsh and Wellman (2003), the authors propose a market protocol with bidding restrictions referred to as SAMP-SB, which uses a series of simultaneous ascending double auctions. SAMP-SB was shown to be capable of producing highly-valued allocations - solutions which maximize the difference between the costs of participating producers and the values obtained by participating consumers - over several network structures, although it frequently struggled on networks where competitive equilibria did not exist. The authors also proposed a similar protocol 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 recognized as an imperfect approach, however, due to the possible problems created by rendering the results of auctions as non-binding.

Babaioff and Walsh (2003) proposes a one-shot double auction mechanism, referred to as Trade Reduction auctions, based upon existing work that sacrifices perfect allocative

efficiency in order to guarantee incentive compatibility, individual rationality and budget balance. The authors propose both a centralized 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 centralization which, as mentioned earlier, may not always be valid.

In Walsh *et al.* (2000), the authors use a combinatorial auction protocol on a subset of the networks in Walsh and Wellman (2003) to attempt to find allocations under strategic bidding behavior 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 behaviors 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.

Recent work has seen the proposal of mixed multi-unit combinatorial auctions (MMUCAs) for supply chain formation (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 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, decentralized solver for MMUCAs remains an ongoing area of research.

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 decentralized coordination (Crick and Pfeffer, 2003; Voice *et al.*, 2010). Winsper and Chli (2010) recently applied an LBP-based approach to the supply chain formation problem, noting that the passing of messages in LBP is comparable to the placing of bids in standard auction-based approaches. The results presented in this paper suggest that LBP is capable of consistently optimal allocations over a range of network structures. The decentralized and distributed nature of LBP also allows for the avoidance of the scalability issues present in centralized approaches such as combinatorial auctions. In this article we provide additional detail on this promising approach to the supply chain formation problem, and present additional results which serve to further illustrate the advantages of this approach.

2.2. Probabilistic Graphical Models

Probabilistic 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, known as Bayesian networks (BNs), represent qualitative dependence between variables - an arc from node i to node j indicates that i causes j, as well quantitative statistical dependence - an arc between nodes also corresponds to a conditional probability of the state of a child node given its parent(s) - $p(x_j|x_i)$ represents the probability of x_j given 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. LBP is easily applicable as a tool for approximate inference in both BNs and MRFs.

Though the task dependency networks of Walsh and Wellman (2003) we use for the base representation of our supply chain networks do not encode dependence - in this case, an arc from i to j now means i is able to supply a good which j is able to consume, rather than any notion of causality - they are easily convertible into a form structurally identical to MRFs when explicit representation of goods is removed. We are able to use the production costs of producer agents and consumption values of consumer agents as analogues for evidence at each node, and encode a simple series of compatibility constraints as pairwise cost functions. From this point, with what is essentially a pairwise MRF with evidence at each node, we are able to use LBP as a means for finding the optimal allocations in our networks.

2.3. Loopy Belief Propagation

LBP is a decentralized and distributed approximate inference scheme involving the application of Pearl's belief propagation algorithm (Pearl, 1988) to graphical models containing cycles. It 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. Because we are interested in finding the optimal state configuration of the network as a whole rather than the most likely state of any one node, we use a well-known variant of LBP, the max-sum algorithm, to estimate the maximum a posteriori (MAP) assignment of our supply chain networks.

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 max-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).

Max-sum LBP is well-suited as a means for allocation determination in supply chain formation for a number of reasons. First, as mentioned earlier, the formalism introduced in Walsh and Wellman (2003) for the representation of supply chains as task dependency networks - bipartite directed acyclic graphs with nodes representing producers, consumers and goods linked by edges representing potential flows of goods, allows for easy conversion into pairwise MRFs suitable for inference once explicit representation of goods in the graphs is removed. 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. LBP operates in a decentralized and distributed manner, properties important for the realistic representation of separate self-interested business entities. Finally, LBP is able to quickly and reliably produce exact results (in our case, this corresponds to a result with

optimal efficiency) in tree-structured and single-cycle networks while still often being able to produce good approximations of the optimal in more loopy networks.

3. MODEL

The base representation of our supply chain networks is as task dependency networks in the form of bipartite directed acyclic graphs. An example of this representation is given in Figure 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.

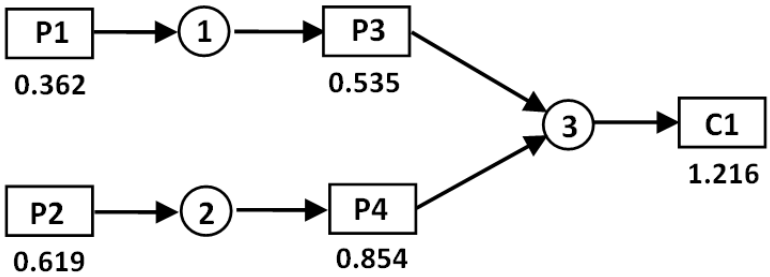

FIGURE 1: A sample supply chain network - Simple - from Walsh and Wellman (2003). Producers (P1, P2, P3, P4) and consumers (C1) are represented by rectangles, while goods are represented by circles. Edges between vertices indicate potential flows of goods. Numbers below producers represent production costs, while numbers below consumers indicate consumption values.

This representation, first proposed in Walsh and Wellman (2003), allows for the clear statement of network structures while retaining fidelity to the structure of real-world supply chains. For example, in Figure 1, we see that producer P1 is able to produce good 1 at a cost of 0.36, which producer P3 needs to consume in order to produce good 3, at a cost 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 accumulated cost, in this example P3, leaving C1 with a final positive consumption value of $1.216 - (0.362 + 0.535) = 0.319$. In line with Walsh and Wellman (2003), goods represent a single unit of a commodity which is non-divisible, and equivalent in all aspects other than price; for reasons of simplicity and clarity, we do not attempt to model aspects such quality, multi-unit transactions or delivery constraints in this article, although this representation subsumes the multi-unit case.

3.1. Agents

Our supply chain networks are made up of multiple interlinked producers aiming to supply a good or goods to one or more consumers.

3.1.1. *Producers.* Producers are capable of producing a single unit of a single type of output good, and to do so are required to have obtained a single unit of each of the goods in their set of input goods, which may be zero, one, or many. This single-unit restriction is made to allow for comparison with Walsh and Wellman (2003); our representation is readily generalizable to the multi-unit case. 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. 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. Producers assign a reserve price R_p to their output good, which is a producer-specific constant encoding the cost of producing the good plus an additional fixed profit margin.

3.1.2. *Consumers.* Consumers require a single unit of a single good from their set of 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 of its consumable goods.

3.2. Conversion to MRF form

To convert the task dependency networks given 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.

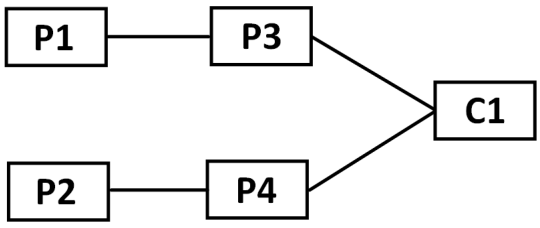

FIGURE 2: The Simple supply chain network converted into MRF form. Edges now link agents directly, and are undirected.

3.3. 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 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.

3.4. 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 minimize these costs and thus the cost function given below:

$$\epsilon(x_1, \ldots, x_N) = \sum_{v \in V} f_v(x_v) + \sum_{(u,v) \in E} g_{uv}(x_u, x_v) \tag{1}$$

Where $x_1, , x_N$ 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.

3.4.1. *Unary Cost.* Each agent associates each of its states with a cost. These values represent the cost to the value of Equation 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.

3.4.2. *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.

3.4.3. *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 1) in the Simple network, as shown in Figure 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 1: Pairwise costs between $P1$ and $P3$, and $P3$ and $C1$, in the Simple network.

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

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

3.5. Supply Chain Formation using max-sum LBP

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 initializing 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.

3.5.1. *Belief Update.* For each of agent u's possible states, we use equation 2 to calculate u's belief in that state. At initialization, each agent holds a belief of zero about each of its states.

$$bel_u(x_u) = f_u(x_u) + \sum_{w \in N_u} m_{w \to u}(x_u) \tag{2}$$

$bel_u(x_u)$ 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 $w \in N_u$.

3.5.2. *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).

$$m_{u \to v}(x_v) = min_{x_u}\Big(bel_u(x_u) - m_{v \to u}(x_u) + g_{uv}(x_u, x_v)\Big) \tag{3}$$

Equation 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. Before we perform allocation, we determine the "final state" of each agent - the state, at convergence, in which the agent believes holds the lowest cost.

3.6. 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, which controls termination but is otherwise uninvolved in the workings of the algorithm, preserving the distributedness of our approach.

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. If the current number of iterations is greater than the size of the spanning tree - as explained in the following paragraphs - and all agents reported that their lowest-cost state has remained the same as the previous iteration, then the convergence detector agent halts the algorithm, and allocation is performed. The process of allocation is outlined in Section 3.7.

As mentioned in Section 2.3, 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 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 shortest distance between all pairs of nodes - the diameter of the tree - is determined.

It is important to note that while the use of a convergence detector agent serves to shorten the running time of the algorithm, it is not required for the algorithm to produce solutions and thus does not represent a single point of failure. We present below two potential alternatives to the use of a dedicated convergence detector agent for situations requiring full decentralization.

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. 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. 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. 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.

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 affect the quality of solutions produced.

3.7. Allocation

Once the final states of each of the agents have been determined, we can perform the process of allocation. 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 their outgoing edges. At the conclusion of this process, producers with an outgoing edge are regarded as having produced a good in the allocation, while consumers with an incoming edge have regarded as having acquired their consumable good in the allocation.

3.8. Allocation Value

We determine the value of our allocations by the equation 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 equivalent to Equation 1.

$$Val = \sum_{c \in C} V_c - \sum_{p \in P} R_p \tag{4}$$

3.9. 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. Active consumers may acquire goods at a cost below their consumption value. 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.

3.10. Alternative Approaches

There are, of course, other possible approaches to this problem: as well as market-based approaches such as auctions and negotiations, fully centralized techniques such as mixed integer programming are capable of finding optimal allocations in fractions of a second; various global search algorithms would also be capable of finding optimal allocations. Myopic techniques such as greedy search may generate optimal solutions on networks without resource contention, such as the Simple network. 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 5), may lead to infeasible solutions. As mentioned in Section 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 decentralized applications, and such modifications, whether they involve clustering nodes or cutting edges, serve to destroy the trading relationships (and thus the logical paths for flows of information) originally present in the network. LBP is able to deal with resource contention and produce efficient results whilst preserving the original structure of the network, and operates in a distributed and decentralized manner.

4. EXPERIMENTS

4.1. Network Structures

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 three additional networks of our own creation. These networks exhibit a variety of structural properties intended to show the performance of LBP under varying conditions. Upon initialization 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 following figures, over every run. We implemented our system in Java.

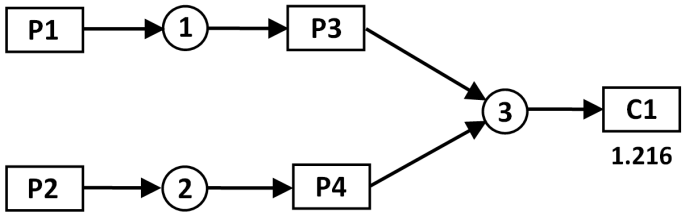

FIGURE 3: Simple network, from Walsh and Wellman (2003)

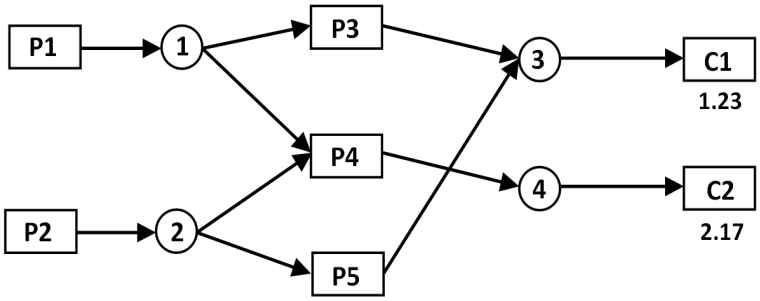

FIGURE 4: Two-Cons network, from Walsh and Wellman (2003)

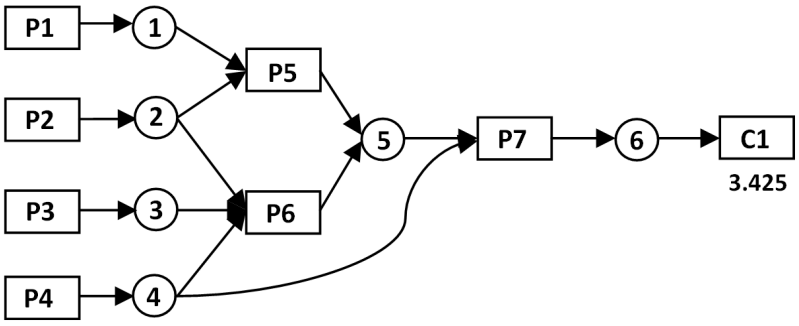

FIGURE 5: Greedy-Bad network, from Walsh and Wellman (2003)

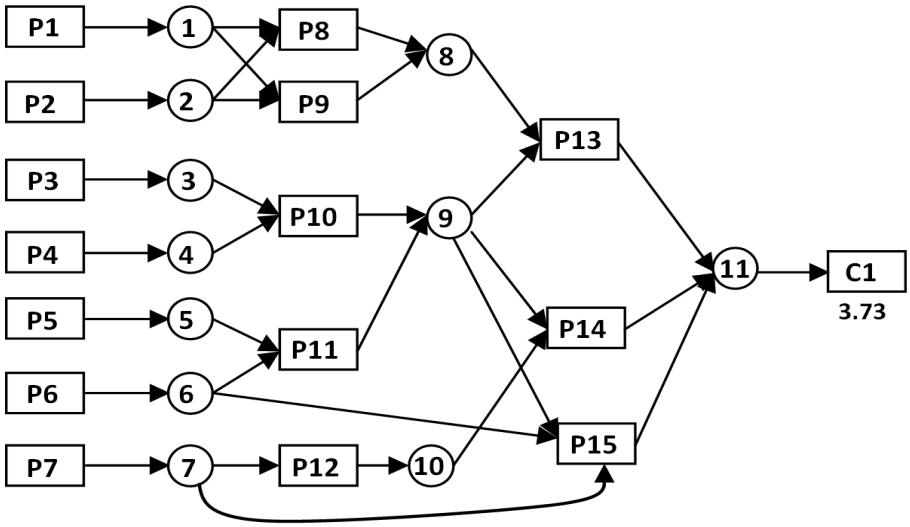

FIGURE 6: Unbalanced network, from Walsh and Wellman (2003)

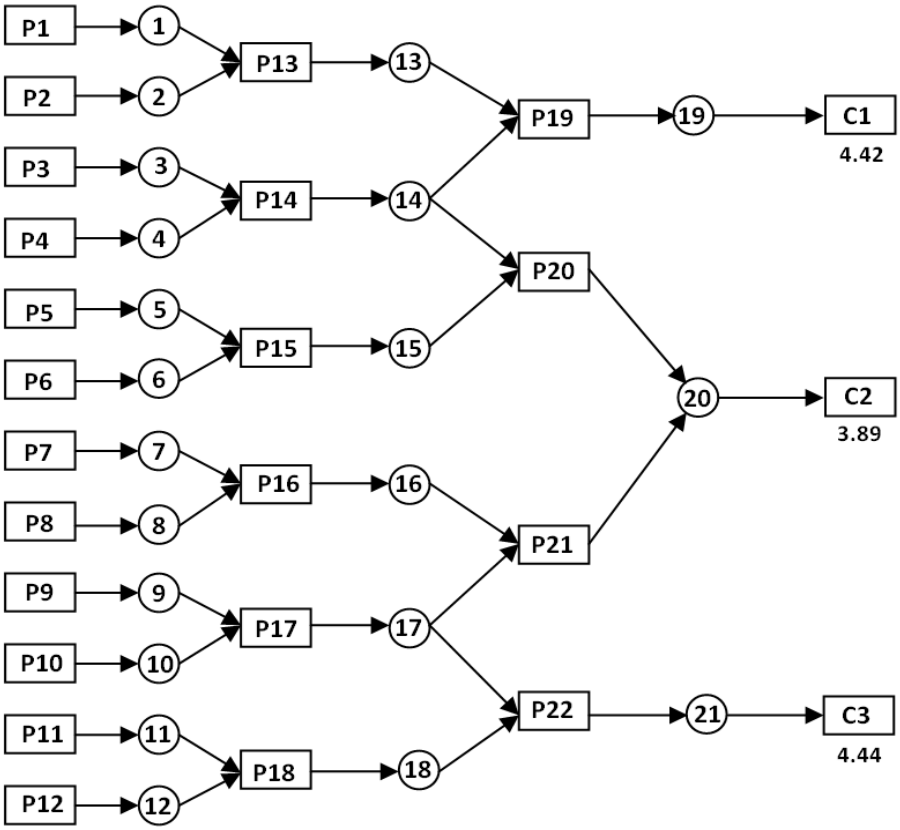

FIGURE 7: Many Cons network, from Walsh and Wellman (2003)

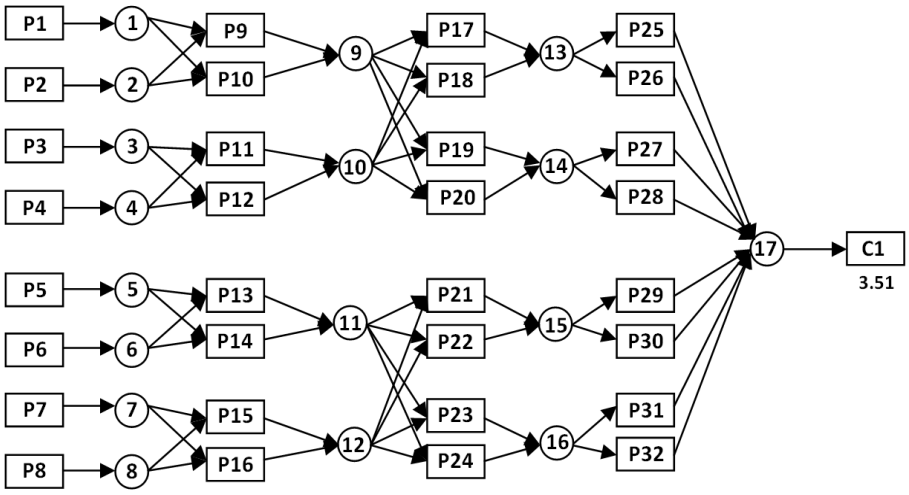

FIGURE 8: Bigger network, from Walsh and Wellman (2003)

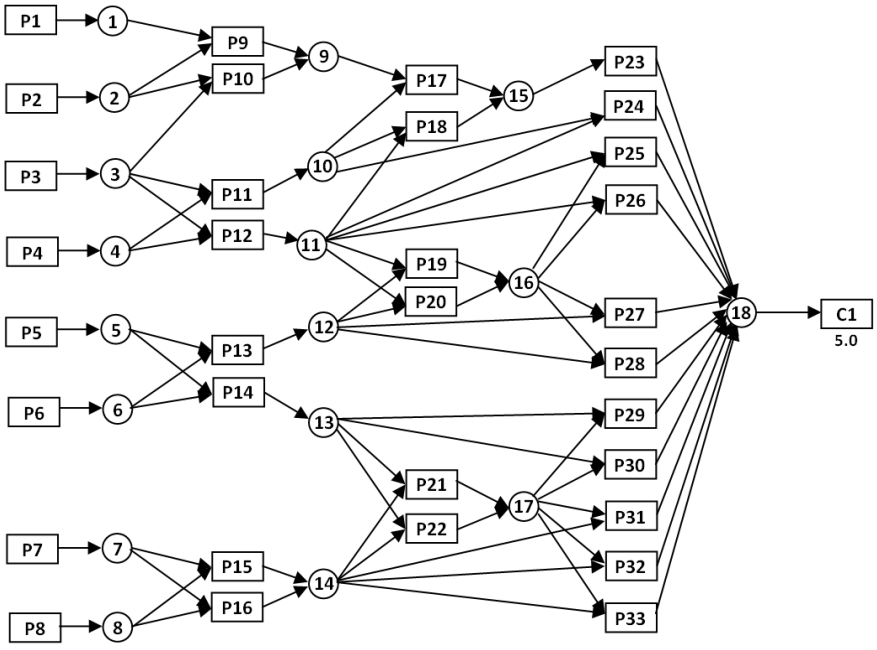

FIGURE 9: Harder network, from Walsh and Wellman (1999)

The Simple network, shown in Figure 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 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 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

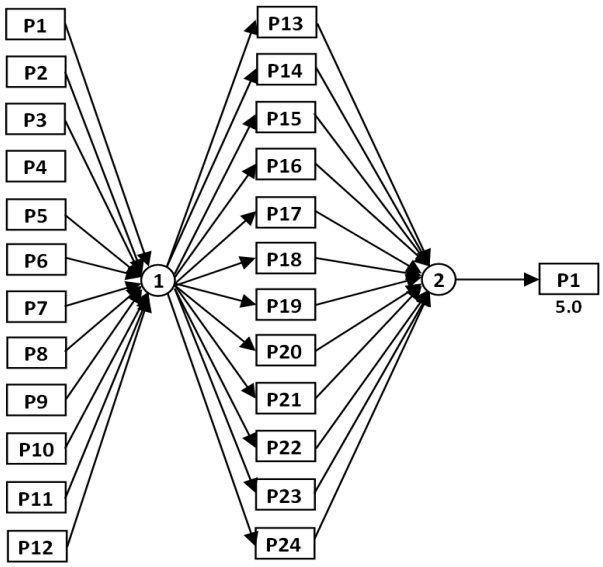

FIGURE 10: Three-tier Many-Alts network

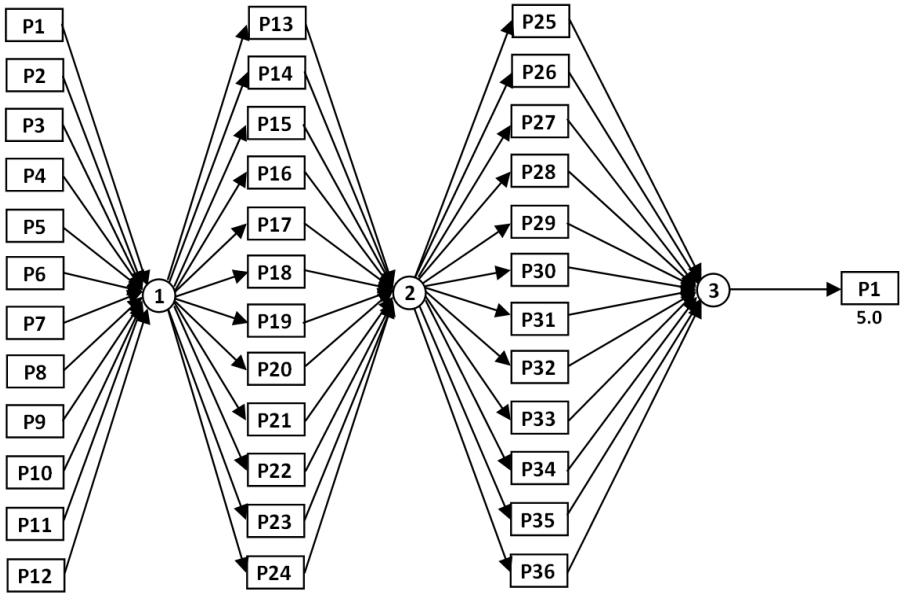

FIGURE 11: Four-tier Many-Alts network

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 6 shows Unbalanced, a larger network with several instances of complementarity. The Many-Cons network, shown in Figure 7 is a larger tree-structured network in which multiple consumers can be satisfied simultaneously. The Bigger network, in Figure 8 is a large-scale network with many feasible solutions. Harder, shown in Figure 9 can be seen as

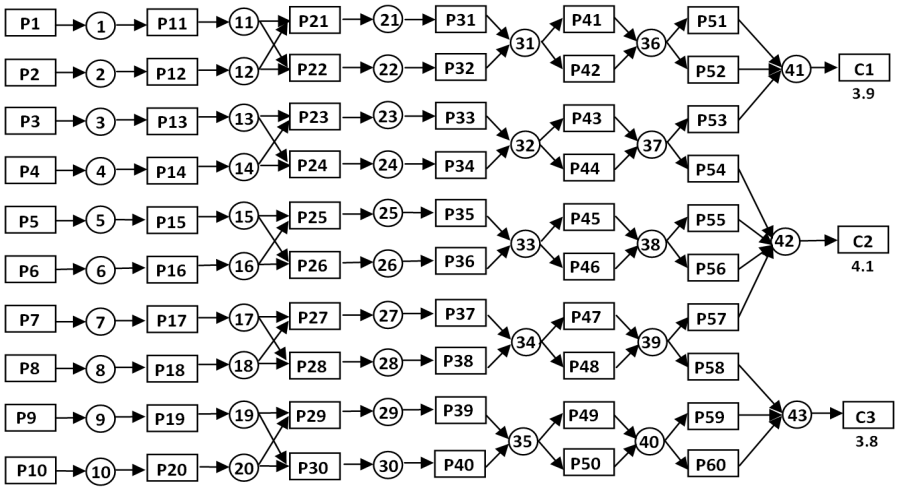

FIGURE 12: Huge network

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. The three-tier Many-Alts network, shown in Figure 10, models a scenario when there exists a large number of buyers and sellers of an identical good, similar to a commodities market. The four-tier Many-Alts network, in Figure 11, increases the complexity of the problem by introducing an additional tier of producers. Producers in the second tier of this network are faced with 144 possible alternatives for sourcing and sales. Finally, the Huge network, shown in Figure 12, models a very large-scale supply chain, with six tiers of production and three consumers.

4.2. 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. 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, with a single winning buyer with a bid at or above the (M+1)st price and a single winning seller with a bid at or below this price. Buyers and sellers bid 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 3.7, with production costs of active producers taking the place of reserve prices in the allocation value calculation.

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 3.7. 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.

4.3. 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. Competitive equilibrium, as defined in Walsh and Wellman (2003), is a set of reserve prices/production costs 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.

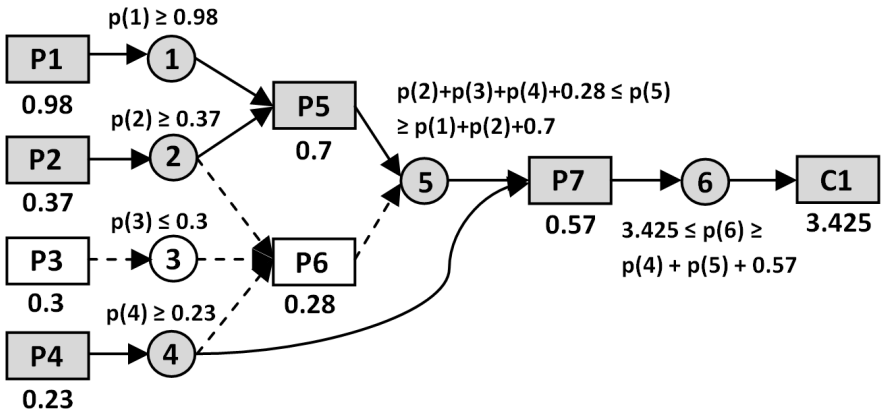

FIGURE 13: An instance of the Greedy-Bad network with sample prices that do not permit competitive equlibrium

Figure 13 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. 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.

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.

4.4. Efficiency

We divide our results into one of four efficiency classes: negative, zero, suboptimal and optimal. Recall equation 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 maximizes 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:

4.4.1. *Negative.* A negative-valued allocation is an allocation in which the reserve prices (LBP)/production costs (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 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 give a clearer picture of the performance of our proposed approach.

4.4.2. *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.

4.4.3. *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.

4.4.4. *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 centralized benchmark, determined by local search.

5. RESULTS

5.1. 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. This is equivalent to our zero result in which no convergence is reached. 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.

Table 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	60.0	0.0	3.0	37.0	3.0	42.0	55.0
No CE	1.0	87.0	0.0	12.0	100.0	0.0	0.0	0.0	99.0	0.0	1.0
Huge	0.0	3.0	96.0	1.0	0.0	0.0	37.0	63.0	0.0	37.0	63.0
Many-Alts 3	0.0	9.0	0.0	91.0	0.0	0.0	0.0	100.0	0.0	0.0	100.0
Many-Alts 4	0.0	9.0	0.0	91.0	0.0	0.0	0.0	100.0	0.0	0.0	100.0

We see from Table 2 that LBP is able to match SAMP-SB's performance for networks Simple and Bigger, while producing less efficiency on Huge and slightly less on the two Many-Alts networks. 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 6, 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 no 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 vastly more efficient.

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.

5.2. Average Efficiency

Table 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.

We see from Table 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, LBP strongly outperforms SAMP-SB-D in the absence of competitive equilibria, capturing at worst 18% more and at best 96% more of the available efficiency in networks Harder and Greedy-Bad, respectively.

Table 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	-0.691	0.684
No CE	0.192	-2.664	0.006
Huge	0.646	0.908	0.911
Many-Alts 3	0.91	1.000	1.000
Many-Alts 4	0.91	1.000	1.000

5.3. Messages and Bids Before Convergence

Table 4: Average numbers of messages passed before convergence and the average total bandwidth required in each network using the proposed LBP-based technique compared with the average numbers of bids placed in each network before quiescence and the average number of bids placed and price quotes sent 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
Many-Alts 3	5336.64	188665.9	107.43	1962.99
Many-Alts 4	11504.0	515839.4	179.88	3670.55

Table 4 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 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. The other exception is with the two Many-Alts networks. Although LBP converges reliably at or near the minimum number of iterations for this network, the large number of messages

passed is due to the strong interconnectedness of these networks - each of the 12 producers in each tier is connected to every producer in the previous and next tiers of the network, and must pass a message to each of them at each iteration.

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.

5.4. 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. Since LBP is a distributed and decentralized algorithm, computational scalability is not an issue. This is also true of SAMP-SB and SAMP-SB-D.

We see from Figures 14, 15 and 16 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 difficulties when applied to loopy networks.

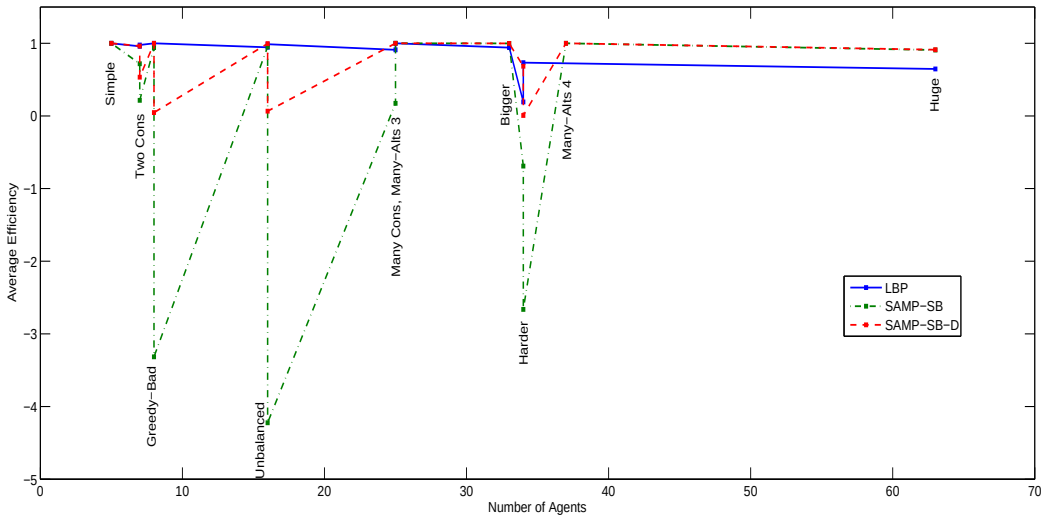

FIGURE 14: A graph showing how efficiency in each of the protocols varies with the number of agents in a network.

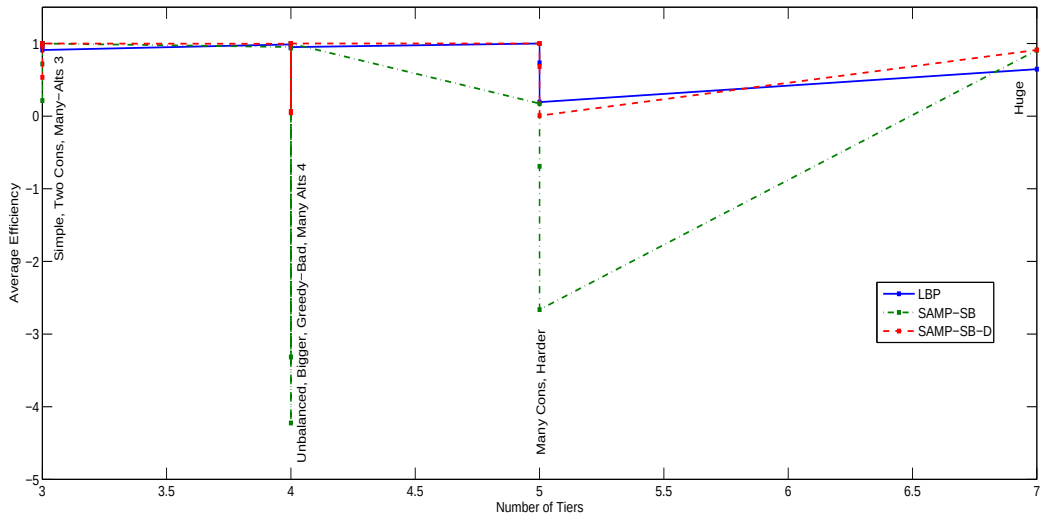

FIGURE 15: A graph showing how efficiency in each of the protocols varies with the number of tiers in a network.

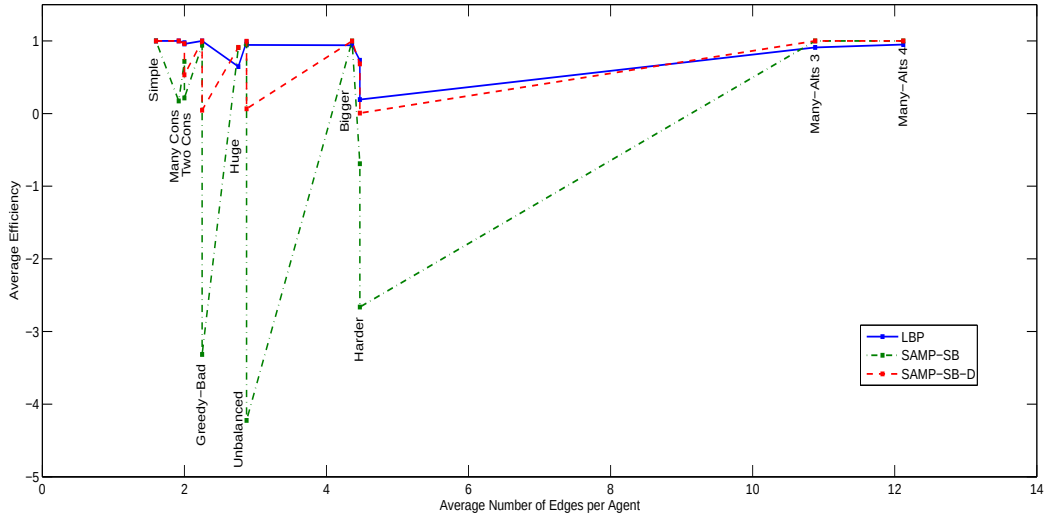

FIGURE 16: A graph showing how efficiency in each of the protocols varies with the average interconnectedness of the agents in a network. We measure average interconnectedness as the total number of edges of each agent in the network divided by the number of agents.

5.5. 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.

5.5.1. *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 using a similar process of post-allocation decommitment.

5.5.2. *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.

5.5.3. *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.

5.5.4. *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.

6. CONCLUSIONS

In this paper, we present a new method for decentralized 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 decentralized message passing to propagate beliefs held by our agents, is able to perform significantly better at finding efficient allocations for most networks than the established approach utilizing simultaneous ascending double auctions we compare it to, whilst making no assumptions of centralization.

For the majority of networks tested, we were able to show that max-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.

We believe that our method provides an interesting avenue for future research by merit of its ability to produce more efficient allocations than an established auction protocol in a comparable scenario whilst operating in a decentralized manner. Our agents share limited information about their reserve price and production capabilities to their neighbors in the network. This means that participants reveal no more private information using our method than in an open auction. By allowing our agents to share information about their reserve price, and about which goods they wish to buy and sell, we are able to produce highly efficient allocations over a range of network topologies.

7. FUTURE WORK

In using Walsh and Wellman (2003) as a basis for our work, we traded the possibility of potentially complex extensions in favor of an expressive graphical representation of supply chain networks and a basis for fair comparison. While the abstractions we have made (abstractions are present to some degree in all agent-based approaches to supply chain formation) serve to limit the practical value of our approach were it to be applied as-is, work is ongoing to increase the fidelity of our model to the numerous constraints present in real-world supply chain formation. Through the implementation of such extensions, we aim to enrich our model to a point at which makes application of our technique possible in a more realistic scenario, such as the TAC SCM game or simulations of real-world supply chains.

The most obvious first extension would be to introduce a stronger pricing element into our approach. At present, reserve prices in our model are fixed. By allowing producers to change their margins (and thus the prices of their goods) during the running of LBP we grant them increased autonomy, though at the cost of making the process of determining the optimal allocation more difficult. Promisingly, LBP has been shown to be resistant to the effects of alterations to observations (equivalent to our unary costs) in the related area of sensor network communication (Crick and Pfeffer, 2003).

Further potential extensions might involve expanding the properties of goods to take into account factors such as quality, quantity, delivery dates and default penalties. Producers could be improved by implementing properties to model production capacity and the possibility of strategic behavior - at present all agents are truth-telling - while consumers might be imbued with richer preferences over goods. A temporal, dynamic aspect could be introduced, with trading relationships forming and dissolving over time, with trust and reputation interesting issues to be taken into account in this scenario. *These additions would require agents to be outfitted with a much larger number of states than in the approach presented in this article, but a larger number of states is not a great obstacle to application of the algorithm to more realistic problem scenarios because of its distributed nature, the simplicity of the calculations made by each agent, and the compact representation of both states and beliefs.*

REFERENCES

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

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *Pattern*

Analysis and Machine Intelligence, IEEE Transactions on, **23**(11), 1222 –1239.

Cerquides, J., Endriss, U., Giovannucci, A., and Rodrguez-Aguilar, J. (2007). 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.

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

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

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

Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. (2008). 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*.

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

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

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

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

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*, volume 1. Cambridge University Press, 1st edition.

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

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

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

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

Ottens, B. and Endriss, U. (2008). Comparing winner determination algorithms for mixed multi-unit combinatorial auctions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, AAMAS '08, pages 1601–1604.

Pardoe, D. and Stone, P. (2006). 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.

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

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

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

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

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

Walsh, W. and Wellman, M. (2003). Decentralized Supply Chain Formation: A Market Protocol and Competitive Equilibrium Analysis. *Journal of Artificial Intelligence Research*, **19**, 513–567.

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

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

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

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

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

Winsper, M. and Chli, M. (2010). Decentralised Supply Chain Formation: A Belief Propagation-based Approach. In *ECAI 2010, 19th European Conference on Artificial Intelligence*, pages 1125–1126.