On Market-Inspired Approaches to Propositional Satisfiability ⋆

William E. Walsh *

IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532, USA

Makoto Yokoo

NTT Communication Science Laboratories, 2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan

Katsutoshi Hirayama

Kobe University of Mercantile Marine, 5-1-1 Fukaeminami-machi, Higashinada-ku, Kobe 658-0022, Japan

Michael P. Wellman

University of Michigan AI Laboratory, 1101 Beal Ave, Ann Arbor, MI 48109-2110, USA

Abstract

We describe three market-inspired approaches to propositional satisfiability. The first is based on a formulation of satisfiability as production on a supply chain, where producers of particular variable assignments must acquire licenses to fail to satisfy particular clauses. Experiments show that although this general supply-chain protocol can converge to market allocations corresponding to satisfiable truth assignments, it is impractically slow. We find that a simplified market structure and a variation on the pricing method can improve performance significantly. We compare the performance of the three market-based protocols with distributed breakout algorithm and GSAT on benchmark 3-SAT problems. We identify a tradeoff between performance and economic realism in the market protocols, and a tradeoff between performance and the degree of decentralization between the market protocols and distributed breakout. We also conduct informal and experimental analyses to gain insight into the operation of price-guided search.

Key words: satisfiability, market-oriented programming, distributed constraint satisfaction, auctions

1 Introduction

Multiple agents must often engage in activities with complex, interrelated dependencies. These dependencies may arise from contention for limited resources, scheduling constraints, or direct dependencies of activity on the results of other agents' activities. Often, such problems of choosing activities, allocating resources, determining schedules, and composing results from individual agents' actions can be modeled as constraint satisfaction problems (CSPs).

As members of the class of NP-complete problems, CSPs are widely considered to be intractable. Even the best algorithms for NP-complete problems generally require exponential time in the worst case. The problem of solving CSPs in the multiagent context is further complicated by their decentralized nature, which imposes constraints on the distribution of information and authority among participants. In a ***decentralized system***, the information state of an individual is considered private, and is disseminated only by voluntary communication acts. This contrasts with centralized systems, in which it is generally assumed that a single entity (the "center") can obtain knowledge of the entire information state, for example by compelling communication. Decentralization constraints clearly restrict the computations performed by individual participants.

Yokoo and Hirayama [3,4] formalize CSPs with decentralization constraints as distributed constraint satisfaction problems (DisCSPs) and, with others, have designed a variety of effective algorithms, such as the distributed breakout (DB) algorithm [5]. These approaches are generally distributed adaptations of centralized algorithms, and can perform very effectively.

Markets provide another model of decentralized systems with clearly delineated boundaries of knowledge and lines of communication. Typically, participants (agents) maintain knowledge of only resources of direct interest, and interact with other agents only indirectly through market institutions, such as auctions. The market-based approach has become increasingly popular in recent years, as evidenced by the growing prevalence in the AI literature of research in the design and analysis of computational market systems and their underlying mechanisms. Markets have been proposed to solve a diversity of problems, with climate control [6], power load management [7], allocating com-

---

⋆ Includes material previously presented at the Seventeenth National Conference on Artificial Intelligence (AAAI-00) [1], and the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01) [2].

* Corresponding author.
Email addresses: wwalsh1@us.ibm.com (William E. Walsh), yokoo@cslab.kecl.ntt.co.jp (Makoto Yokoo), hirayama@ti.kshosen.ac.jp (Katsutoshi Hirayama), wellman@umich.edu (Michael P. Wellman).

putational servers [8], multicommodity flow [9], and belief aggregation [10] being just a sample.

Shoham and Tennenholtz [11] directly pose the question "What can a market compute, and at what expense?" They provide answers for some interesting cases, applying concepts from economic mechanism design and communication complexity. Different behavioral assumptions can support conclusions about additional cases. For instance, over fifty years ago, Samuelson [12] considered how markets could decentralize the solution of linear programming problems. More generally, adopting market protocols in the framework of general equilibrium theory can be seen to yield a computational model capable of solving convex programming problems [13]. However, none of these lines of analysis provide answers with respect to the sort of combinatorial optimization problems of most interest in AI and multiagent systems research.

To address this gap, we examine here the possibility of using markets to solve propositional satisfiability (SAT) problems in a decentralized manner. We refer to a market protocol applied to the solution of SAT problems as a ***MarketSAT*** protocol. As the original NP-complete problem, SAT is considered fundamental, has been thoroughly studied, and indeed remains the object of active research. Studies in AI have led to a greater understanding of its difficulty [14], and steady improvements in centralized algorithms, starting with the success of GSAT [15].

We begin our exploration of market-inspired approaches to SAT by showing how to transform SAT to a supply chain formation problem. We can then employ a previously-developed market protocol for general supply chain formation problems [16–18], thus obtaining a MarketSAT protocol. In Section 2 we describe the supply chain formation problem and show how to reduce SAT to supply chain formation in the context of an NP-completeness proof. In Section 3 we describe the Original MarketSAT protocol (MS-O), based on the supply chain formation market protocol we previously developed. We experimentally evaluate MS-O in Section 4 and find that, while MS-O can solve SAT problems, it is prohibitively slow. In Section 5 we develop variant MarketSAT protocols to address the performance shortcoming of MS-O. In Section 6 we discuss the intuition behind the price-guided search of the MarketSAT protocols and show that they are incomplete. In Section 7 we evaluate the protocols along different dimensions. In Section 7.1 we describe experiments with the variant MarketSAT protocols. These experiments show that the variants perform significantly better than Original MarketSAT, but not as well as distributed breakout. We also investigate possible causes for the performance discrepancies. We further evaluate the protocols in terms of degree of decentralization (Section 7.2) and economic interpretation (Section 7.3), identifying tradeoffs between these dimensions and performance. We conclude in Section 8.

2 Supply Chain Formation

2.1 The Supply Chain Formation problem

Decentralized supply chain formation, informally, is the problem of assembling a network of agents that can transform basic goods into composite goods of value, given local knowledge and communication [17]. The term "good" refers to any discrete resource or task for which the results cannot be shared between agents.

More precisely, we formulate the problem as follows [19,18]. A ***task dependency network*** is a directed, acyclic graph, (V, E), representing dependencies among agents and goods. $V = G \cup A$, where G is the set of goods and $A = C \cup \Pi$ is the set of agents, comprised of consumers C and producers Π. Edges, E, connect agents with goods they can use or provide. There exists an edge $\langle g, a \rangle$ from $g \in G$ to $a \in A$ when agent a can make use of one unit of g, and an edge $\langle a, g \rangle$ when a can provide one unit of g. When an agent can acquire or provide multiple units of a good, separately indexed edges represent each unit. For instance, edges $\langle a, g \rangle_1$ and $\langle a, g \rangle_2$ would represent the fact that agent a can provide two units of good g. The goods can be traded only in integer quantities.

A ***consumer*** wishes to acquire one unit of one good among a set of possible goods. A ***producer*** can produce a single unit of an ***output*** good conditional on acquiring a certain number each of some fixed set of ***input*** goods. A producer must acquire each of its inputs to provide its output. Figure 1 shows an example task dependency network. The boxes and oblong shapes represent agents, the circles represent goods, and the arrows represent edges. The agents on the right-hand side (agents c1 and c2) are consumers and the other agents are producers. Observe that producer a1 requires no inputs to produce output good 1, while producer a4 requires both of its input goods 1 and 2 to produce output good 4.

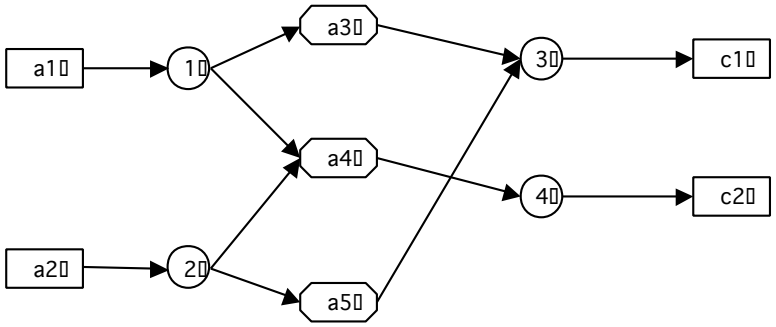

Fig. 1. A task dependency network.

An ***allocation*** is a subgraph $(V', E') \subseteq (V, E)$. For $g \in G$, an edge $\langle a, g \rangle \in E'$ means that agent a provides g, and $\langle g, a \rangle \in E'$ means a acquires g. An agent is in an allocation graph iff it acquires or provides a good. A good is in an allocation graph iff it is bought or sold.

A producer is ***active*** iff it provides its output. A ***producer is feasible*** iff it is inactive or acquires all its inputs. Consumers are always feasible. An ***allocation is feasible*** iff all producers are feasible and all goods are in ***material balance***, that is, the number of edges into a good equals the number of edges out.

A ***solution*** is a feasible allocation such that one or more consumers acquire a desired good. If $c \in C \cap V'$ for solution (V', E'), then (V', E') is a ***solution for*** c. Figure 2 shows one solution for the task dependency network from Figure 1. The shaded agents and goods are in the solution, and the solid arrows indicate exchanges of goods realized in the solution. The dashed arrows indicate exchanges not realized in the solution. Observe that there is one other solution for the network, containing agents c1, a1, a2, a3, and a5, as well as good 3, but *not* agents a4 and c2, nor good 4.

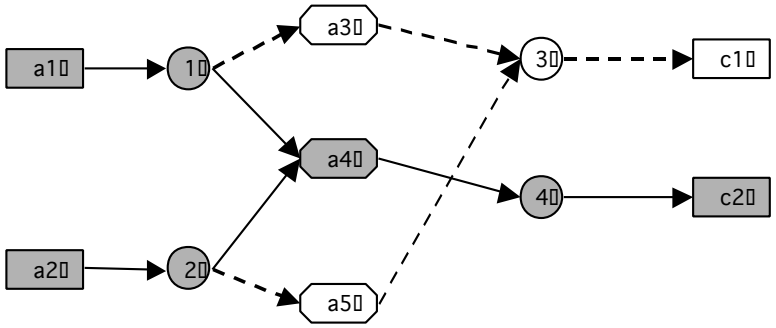

Fig. 2. A solution for the task dependency network from Figure 1. Shaded vertices and solid edges are in the solution.

2.2 Supply Chain Formation is NP-Complete

Due to constraints on resource availability, it is not possible to satisfy both consumers in a solution simultaneously. Consumer c1 wants good 3, which can be provided by agent a3 or a5. If we have a solution for c2, then a1 must provide good 1 to a4. But since a1 can produce only one unit of good 1 (recall that an edge indicates potential production or acquisition of a single unit of a good) a3 cannot acquire good 1. Similarly, a5 cannot acquire good 2, hence c1 cannot be in a solution concurrently with c2. As we shall see in Section 2.2, resource constraints are a key feature that entail NP-completeness of supply chain formation in task dependency networks. In Section 3 we describe a protocol

for satisfying these constraints in a highly decentralized fashion, giving us a decentralized protocol for solving satisfiability problems.

The class of NP problems contains all problems that can be solved in polynomial time by a nondeterministic computer. The ***NP-complete*** problem class contains all problems in NP that are at least as hard (in terms of required computation) as every other problem in NP [20]. It is widely believed that NP-complete problems are intractable, requiring exponential time to solve, in the worst case.

To develop the market approach to solving satisfiability problems, we show that the problem of forming supply chains is NP-complete. To show this, we describe a reduction transformation of SAT to the supply chain formation problem. The transformation combined with a particular market protocol give us MS-O. First, we describe the standard SAT problem and formulate supply chain formation as a decision problem.

Definition 1 (Satisfiability) *(SAT)*

Instance: *Set U of variables, and collection Q of clauses, where each $q \in Q$ is a set of literals over U.*
Question: *Is there a truth assignment $t : U \rightarrow \{T, F\}$ that satisfies each $q \in Q$?*

Definition 2 (Supply chain formation decision problem) *(SUPP-CHAIN).*

Instance: *A task dependency network, (V, E), with agents, goods, and edges as described above.*
Question: *Is there a solution $(V', E') \subseteq (V, E)$?*

Theorem 3 *SUPP-CHAIN is NP-complete.*

Proof concept. Standard methodology for proving NP-completeness [20] requires the following:

(1) Show that SUPP-CHAIN is in the problem class NP. To do so, we need to demonstrate that a proposed solution can be verified in polynomial time.
(2) Show that any instance of SAT can be reduced to an instance of SUPP-CHAIN in polynomial time. The reduction transformation must be such that the answer to the SUPP-CHAIN decision problem instance is "yes" iff the answer to the SAT problem instance is "yes".

For use in the reduction transformation, we say that truth assignment t to variable u ***fails to satisfy*** a clause q, denoted by $\mathrm{FTS}(q, u, t)$, iff either:

(1) $t = T$, $u \notin q$, and $\bar{u} \in q$, *or*, (2) $t = F$, $\bar{u} \notin q$, and $u \in q$. The key observation behind the reduction is that in a satisfying truth assignment, at least one variable must satisfy any given clause, hence at *most* $|q| - 1$ variable assignments can *fail to satisfy* clause q. Thus the transformation ensures that, in order to produce a truth assignment for a variable, a producer must acquire ***licenses*** to fail to satisfy clauses. These licenses are the scarce resources (only $|q| - 1$ are available for clause q) to be allocated.

The task dependency network corresponding to a SAT instance includes goods of the following types:

- g_q: license to fail to satisfy clause q
- g_u: an assignment to variable u
- g_c: a satisfying overall assignment

and agents of the following types:

- s_q^i: for each $i \in [1,\ |q| - 1]$, producer of a license to fail to satisfy q
- π_u: producer of a positive assignment to u
- $\pi_{\bar{u}}$: producer of a negative assignment to u
- π_c: producer of an overall assignment (from individual variable assignments)
- c: consumer of the overall assignment

As described below, we construct the network in such a way as to ensure that only satisfying assignments can be produced, primarily by controlling availability and necessity of failure-to-satisfy licenses.

PROOF. *SUPP-CHAIN is in NP.* It is straightforward to verify that an allocation is a solution by observing whether c obtains a good and by counting the edges incident on each good and producer.

Reduction. We transform an instance of SAT to an instance of SUPP-CHAIN.

(1) For each $q \in Q$, and each $i \in [1,\ |q| - 1]$, add g_q to G, add s_q^i to Π, and add $\langle s_q^i, g_q \rangle$ to E.
(2) For each $u \in U$, do the following:
   - add g_u to G,
   - add π_u to Π, add $\langle \pi_u, g_u \rangle$ to E, and for each $q \in Q$ such that $u \notin q$ and $\bar{u} \in q$, add $\langle g_q, \pi_u \rangle$ to E,
   - add $\pi_{\bar{u}}$ to Π, add $\langle \pi_{\bar{u}}, g_u \rangle$ to E, and for each $q \in Q$ such that $\bar{u} \notin q$ and $u \in q$, add $\langle g_q, \pi_{\bar{u}} \rangle$ to E.
(3) Add c to C, g_c to G, $\langle g_c, c \rangle$ to E, π_c to Π, and $\langle \pi_c, g_c \rangle$ to E. For each $u \in U$, add $\langle g_u, \pi_c \rangle$ to E.

Figure 3 shows the transformation for an example with $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

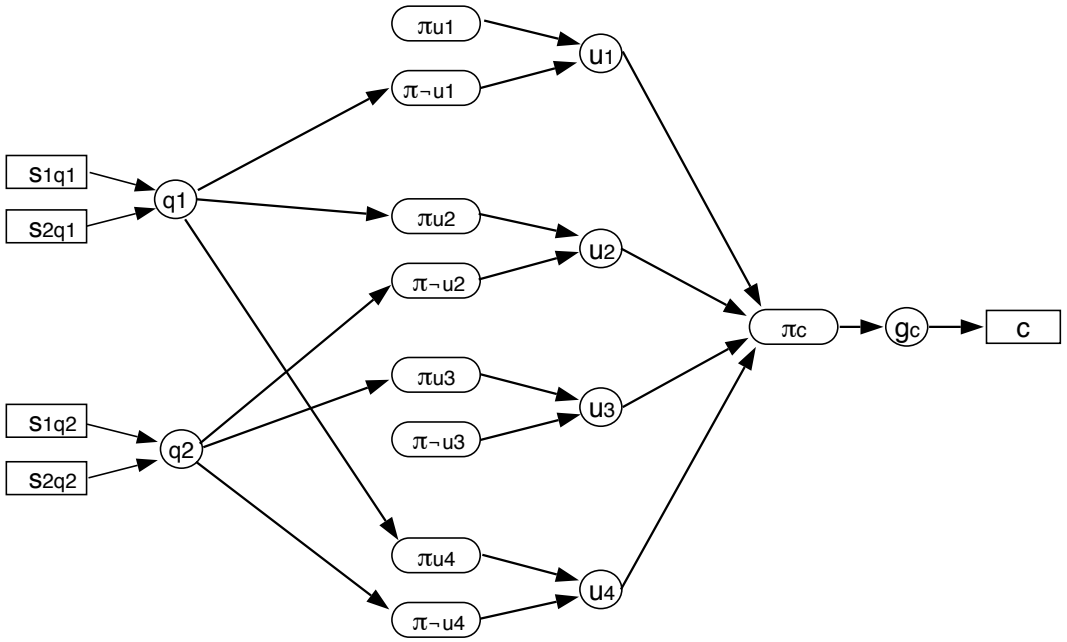

Fig. 3. The transformation for $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

A "yes" configuration for SUPP-CHAIN is a "yes" configuration for SAT. A solution (V', E') must be in material balance, which implies that, for each $u \in U$, either $\langle \pi_u, g_u \rangle \in E'$ or $\langle \pi_{\bar{u}}, g_u \rangle \in E'$. If the former is true, then, we assign $t(u) = T$, otherwise $t(u) = F$.

Since there are $|q| - 1$ units of any g_q available, each of which is in material balance, there are at most $|q| - 1$ edges in E' of the type $\langle g_q, \pi_u \rangle$ or $\langle g_q, \pi_{\bar{u}} \rangle$. But then, there are at most $|q| - 1$ variables in $q \in Q$ whose assignments fail to satisfy q, and hence at least one variable whose assignment does satisfy q under t. Thus, t is a satisfying truth assignment.

Figure 4 shows a feasible allocation that gives a satisfying truth assignment $t(u_1) = T$, $t(u_2) = F$, $t(u_3) = F$, $t(u_4) = T$, for the transformation network shown in Figure 3. Observe that, although $\pi_{\bar{u_4}}$ is inactive, it acquires a unit of q_2. For the purposes of the SAT transformation, it is irrelevant whether inactive producers obtain their inputs, so long as the entire allocation is feasible.

A "yes" configuration for SAT is a "yes" configuration for SUPP-CHAIN. If t is a satisfying truth assignment in the instance of SAT, then we can map t to a solution $(V', E') \subseteq (V, E)$. First, add g_c, π_c, and each $g_u \in G$ to V'. Add $\langle \pi_c, g_c \rangle$ to E', and for each $g_u \in G$, add $\langle g_u, \pi_c \rangle$ to E'. As mentioned above, we can arbitrarily allocate excess units of the licenses, so long as we maintain feasibility. Hence, Figure 3 shows one way to map truth assignment

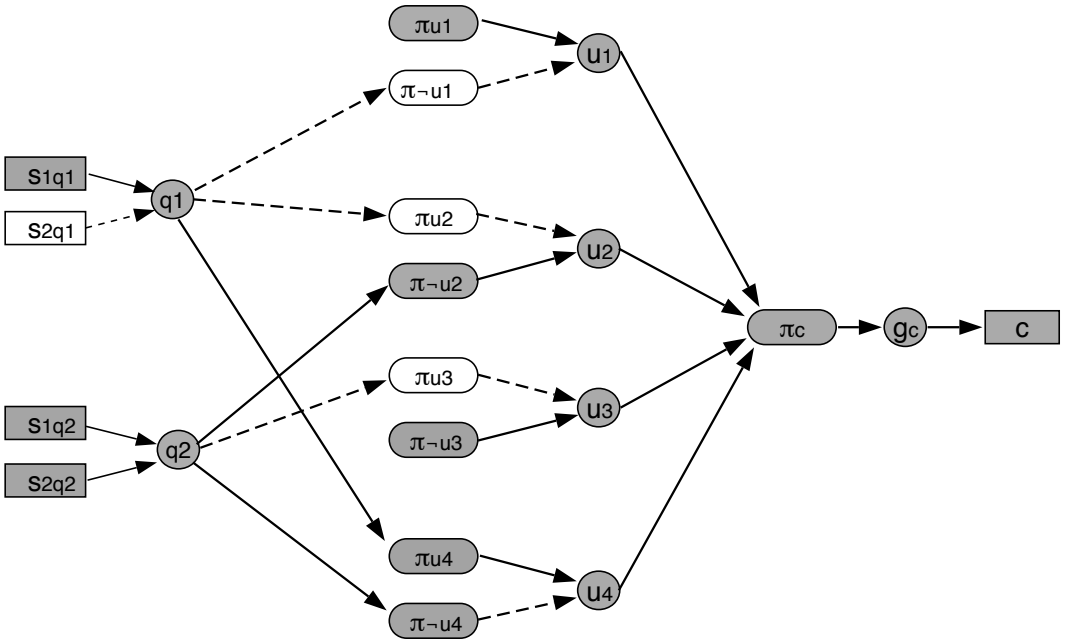

Fig. 4. A solution for the network from Figure 3, which gives a satisfying truth assignment $t(u_1) = T$, $t(u_2) = F$, $t(u_3) = F$, $t(u_4) = T$. Shaded vertices and solid edges are in the solution.

$t(u_1) = T, t(u_2) = F, t(u_3) = F, t(u_4) = T$, for the SAT instance $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

For $u \in U$, if $t(u) = T$, then add π_u to V' and add all input and output edges of π_u to E'. For each $q \in Q$ such that $u \notin q$ and $\bar{u} \in q$, add a new s_q^i to V' and $\langle s_q^i, g_q \rangle$ to E'. If instead, $t(u) = F$, then perform similar operations with $\bar{u}$ and u reversed. Since t is satisfying, for each q in Q there are at most $|q| - 1$ variables whose assignments fail to satisfy q, hence at most $|q| - 1$ producers require g_q as input in (V', E'). Hence, (V', E') is feasible. It is also a solution because $\langle \pi_c, g_c \rangle \in E'$.

The transformation runs in polynomial time. There are $\sum_{q \in Q}(|q| - 1)$ clause suppliers s_q^i, each with one output. There are are $2|U|$ literal producers π_u, each with at most $|Q|$ inputs and one output. The producer π_c has $|U|$ inputs and one output. There are $|Q|$ clause goods g_q and $|U|$ variable goods g_u. Thus the complexity of the transformation is $O(|U||Q|)$. □

3 Original MarketSAT Protocol

Given full knowledge of a SAT problem, we could employ a centralized algorithm known to be effective (e.g., GSAT [15]). But in a decentralized system, agents have only localized knowledge about the problem or the intermediate

states of any solution procedure, and self-interested agents need incentives to participate. In the market framework, price systems, by indicating the relative value of goods, help guide agents' local decision making. We model agent self-interest in terms of recovering costs and acquiring value. We assume that all producer costs are zero. The consumer c obtains value $v_c(g_c)$ for obtaining a single unit of good g_c.

In previous work [19,17], we developed a market protocol—Simultaneous Ascending $(M+1)$st-Price auction with Simple Bidding (SAMP-SB)— for the supply chain formation problem. We found that the protocol can effectively form supply chains in task dependency networks while requiring that an agent has knowledge only of valuations or costs, its goods of interest, and the price information revealed by the auctions thereof. We can thus obtain a market protocol for SAT—which we call ***Original MarketSAT*** (MS-O)—by applying SAMP-SB to SAT task dependency networks.

In the protocol, agents negotiate through auction mediators, one for each good. An auction in turn determines the price and allocation of its respective good. For MS-O, and the other MarketSAT protocols we subsequently describe, we assume reliable *synchronous* message passing, whereby all agents update their bids simultaneously in synchronous ***rounds*** and each auction updates and disseminates information to bidders only after receiving all bids. This assumption serves only to focus study and simplify exposition, for the protocols could be implemented in an asynchronous environment.

3.1 Auction Mechanism

There is a separate auction for each good in the SAT task dependency network. Agents interact with the auctions by submitting bids for goods they wish to buy or sell. A bid specifies the price below/above which the agent is willing to buy/sell. When an auction receives a new bid from each of its bidders, it sends each of its bidders a ***price quote*** specifying the price that would result, and the number of units the good the bidder would win, if the auction ended in the current bid state. Agents may then choose to revise their bids in response to the notifications (unchanged bids remain standing at their current value in the auction).

The auctions run simultaneously, and each auction requires that an agent's successive buy bids increase by no less than some (generally small) positive number δ_b and successive sell bids increase by no less than δ_s.

Bidding continues until ***quiescence***, a state where all messages have been received, no agent chooses to change the value of its bids, and no auction changes any aspect of its price quote. At this point, the auctions ***clear***; for

PROCEDURE **MS-O Auction Mechanism**
Each agent submits initial bids to auctions according to its bidding policy
UNTIL quiescence DO
CONCURRENTLY FOR EACH auction g DO
// Receive bids
Admit all buy bids that increase by δ_b
Admit all sell bids that increase by δ_s
Replace standing bids with new bids
MS-O Report Price Quote
END
CONCURRENTLY FOR EACH agent DO update bids according to its bidding policy
END
// Clear auctions
CONCURRENTLY FOR EACH auction DO
$g \leftarrow$ good handled by the auction
FOR EACH bidder a DO
report the final values $p(g)$ and $\omega_a(g)$ to a, as computed by the last price quote
END

Fig. 5. The MS-O auction mechanism.

PROCEDURE **MS-O Report Price Quote** by the auction for good g
$M \leftarrow$ number of sell bids
$p(g) \leftarrow (M+1)$st highest price of all bids
$\alpha(g) \leftarrow M$th highest price of all bids
FOR EACH buyer a that bids strictly above $p(g)$ DO $\omega_a(g) \leftarrow 1$
FOR EACH seller a that bids strictly below $p(g)$ DO $\omega_a(g) \leftarrow 1$
Match the maximum number of buyer and seller bids at $p(g)$ one-to-one, breaking ties in favor earlier bids, and set $\omega_a(g) \leftarrow 1$ for each respective bidder a
Report $p(g)$, $\alpha(g)$, and $\omega_a(g)$ to each bidder a

Fig. 6. The report price quote procedure for the MS-O auction mechanism.

each good g, each bidder a is notified of the final ***price***, denoted by $p(g)$, and whether it is winning its bid (note that agents want at most one unit of any given good in a MarketSAT task network), denoted by the indicator $\omega_a(g) \in \{0, 1\}$.

Each auction runs according to (M+1)st-price rules [21–23]. The (M+1)st price auction is a variant of the (second-price) Vickrey auction [24], generalized to allow for the exchange of multiple units of a good. Given a set of bids including M units offered for sale, the (M+1)st-price auction sets a price, $p(g)$ equal to the $(M+1)$st highest unit offer over *all* of the bids. The price can be

said to separate the winners from the losers, in that the winners include all sell bids strictly below the price and all buy bids strictly above the price. To maximize exchange at the trading price, some agents that bid at the $(M+1)$st price also win; in case of ties, bids submitted earlier have precedence. Winning buy and sell bids are matched one-to-one.

When issuing price quotes, the auction reports both $p(g)$ and the ***ask price***, $\alpha(g)$ of the good g. The ask price specifies the amount above which a buyer would have to bid in order to buy the good, given the current set of bids. The ask price is determined by the Mth highest of all bids in the auction, hence $\alpha(g) \geq p(g)$. The auction also reports $\omega_a(g)$ to each agent a, which, before the auction clears, indicates the agent's current, tentative winning state based on current bid information. [1]

Figures 5 and 6 detail the operation of the auction mechanism. As an example, consider a state of the auction for good g with two sell offers at 7 and 4, and three buy offers at 8, 5, and 3. Here, $M = 2$, so $p(g) = 5$ (the third highest offer) and $\alpha(g) = 7$ (the second highest offer). The sell offer at 4, which is lower than $p(g)$, and the buy offer 8, which is higher than $p(g)$, will match. The buy offer for 5, which is equal to $p(g)$, will not match because the other sell offer, which at price 7 is strictly higher than $p(g)$.

3.2 Bidding Policies

The strategic problem defined by the auction mechanism and the task dependency network is of a complexity well beyond our ability to derive optimal solutions in the game-theoretic sense. Therefore, we propose simple bidding policies based on myopic behavior and local information. Although we believe that policies of this form are often plausible, we acknowledge that in this context they are clearly not optimal from the agents' perspectives. Further strategic analysis is necessary to establish how agents would reasonably behave in this environment.

Let the current going prices be p. Recall that a consumer desires only one good in a SAT-reduced task dependency network. [2] We assume that a consumer initially bids zero for its good of interest. The consumer increases its bid minimally above the current good price whenever it is not winning the bid

[1] This added information is necessary to determine the current winning state when the agent's bid is exactly at $p(g)$.

[2] The consumer bidding policy we describe here is a specialization of a policy we described for general task dependency networks[19], which includes consumer preferences over multiple goods.

and its next bid would not exceed its value. Figure 7 details the policy for updating bids.

PROCEDURE **MS-O Consumer Update Bids** by consumer wanting good g
  IF losing bid for g AND $v_c(g) - p(g) - \delta_b \geq 0$ THEN bid $p(g) + \delta_b$ for g

Fig. 7. The consumer bidding policy for the MS-O protocol.

PROCEDURE **MS-O Producer Update Bids** by producer π with output g_π in MS-O
  IF π is winning its bid for g_π THEN
    FOR EACH input g of π DO
      IF π is losing its bid for g THEN increase that bid by δ_b
    END
  END
  FOR EACH input g of π DO
    IF π is winning its bid for g, THEN $\hat{p}_\pi(g) \leftarrow p(g)$
    ELSE $\hat{p}_\pi(g) \leftarrow \max(\alpha(g),\ p(g) + \delta_b)$
  END
  $\beta(\pi, g_\pi) \leftarrow$ previous bid by π for g_π
  IF $\hat{p}_\pi(g) > \beta(\pi, g_\pi)$ THEN bid $\max[\beta(\pi, g_\pi) + \delta_s\ ,\ \sum_{\langle g,\pi\rangle \in E} \hat{p}_\pi(g)]$ for g_π

Fig. 8. The producer bidding policy for the MS-O protocol.

We assume that producer π initially bids zero for its output g_π, and then changes the bid in an attempt to recover the ***perceived costs*** $\hat{p}_\pi(g)$ of its inputs. If the consumer is currently winning a good, it perceives the cost to be $p(g)$. If it is losing, the producer perceives the cost to be the maximum of $\alpha(g)$ and $p(g) + \delta_b$, because it would have to bid at least that much to ensure it wins the good, assuming all other agents' bids remained fixed. A producer initially bids zero for each of its input goods and gradually increases these bids to ensure feasibility. Figure 8 details the policy for updating bids.

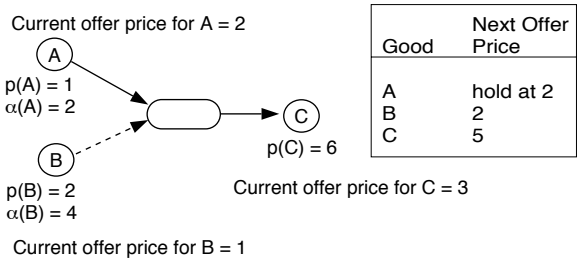

Fig. 9. A producer's next bids, according to SAMP-SB, when $\delta_b = 1$ and $\delta_s \leq 2$.

Figure 9 shows how a producer would bid next as a function of the current prices and its current bids, when $\delta_b = 1$ and $\delta_s \leq 2$. Relevant current price quotes are shown under the goods, exogenously chosen for illustrative purposes. The dashed arrow from good B indicates that the producer is currently losing B. The solid arrows indicate bids the producer is winning.

3.3 Properties of Original MarketSAT

SAMP-SB exhibits several salient properties that carry over from general task dependency networks [19,16] to MS-O. The protocol ensures feasibility in quiescence, by design. The ascending bids and bounded consumer value ensure that it converges to quiescence. With synchronous bidding, MS-O reaches quiescence after a number of bidding rounds polynomial in the network size and the magnitude of the consumer value.

The experimental results we present suggest that MS-O can require a large number of rounds to reach a solution in quiescence (although it is not guaranteed to find a solution, as we show in Section 6.2). Since the number of rounds is bounded by the consumer value, this value effects a tradeoff between the runtime and the chance of finding a solution.

4 Experiments with Original MarketSAT

To understand the performance of MS-O, we performed a battery of experiments comparing MS-O to Distributed Breakout (DB) [5], an adaptation of the breakout algorithm [25] to decentralized environments, and also to GSAT [15], a centralized random-restart hill-climbing algorithm.

4.1 Construction

For these experiments we focus on the behavior of the protocols on hard, benchmark SAT instances. These experiments were conducted using satisfiable (unforced, filtered) Uniform Random 3-SAT problems at the phase transition (between 4.26 and 4.3 clause/variable ratio for these instances) from the SATLIB benchmark library. [3] It is generally considered that the hardest problems occur at a phase transition (also called the crossover point) in the clause/variable ratio where 50% of the problems are satisfiable [26–28]. For this reason, problems at the phase transition have been widely used to benchmark SAT algorithms. Satisfiable problems are used because many SAT algorithms (including all the ones we study here) are incomplete and will run until the specified maximum iterations, with indeterminate results, on unsatisfiable problems. [4]

[3] `http://www.satlib.org/benchm.html`

[4] To generate hard satisfiable problems, it is important to generate problems randomly and filter the unsatisfiable instances, as we described. Forcing the problems to satisfy a particular assignment makes them much easier [29]. The filtering method of

Given that agents and auctions run in parallel, a plausible measure of runtime in a MarketSAT protocol is bidding rounds. Within a given round, the clause auctions are the bottleneck, running in $O(|U| \lg |U|)$ time (though incremental updating techniques can improve amortized performance [23]). This measure is analogous to the cycles commonly measured in experiments of DB [5]. For GSAT, we measured the number of flips, as is customary for centralized algorithms.

To consider alternative measures, recall from the reduction transformation that for each $u \in U$, there are producers π_u and $\pi_{\bar{u}}$. The number of times they "swap" the license to sell g_u is directly analogous to flipping a variable. In any given round of MS-O, zero or multiple such "flips" can take place. However, our experiments show that the number of MS-O flips is approximately linear in the number of rounds (regression result: flips $= 0.5\text{rounds} - 11.9$, $R^2 = .99$).

If the cost of computing a heuristic in a centralized algorithm is relatively expensive, then the algorithm may actually run slower in practice than another that performs more flips. In such cases, a finer-grained performance metric such as CPU cycles may be more appropriate. In decentralized protocols however, communication delay can be a significant factor in practical performance. For DB and the MarketSAT protocols we study, the agent and auction computations are straightforward, hence we expect that communication delay would dominate runtime in a distributed implementation of the protocols. With the synchrony assumption, we consider the number of rounds to be the most appropriate performance measure for our distributed protocols.

For MS-O, we set $\delta_b = \delta_s = 1$. To bound the computational time, for a problem instance of n variables, we ran MS-O and DB for at most $1000n$ rounds. We ran GSAT with at most 200 restarts and at most $5n$ flips per start. For MarketSAT, a round is a synchronous bidding round, and for DB, a round corresponds to a cycle as described by Yokoo and Hirayama [5]. Agents participate until they reach quiescence (with a satisfying solution), or until the maximum number of rounds is reached. We recognize that the artificial limit on the bidding rounds reduces the plausibility that agents may use the specified bidding policy of MS-O and the subsequent MarketSAT protocols we consider (see Section 7.3). Although it would be more natural to indirectly bound the run time with a bound on the consumer value (Section 3.3), we chose the direct limit on rounds to consistently compare the performance of the protocols.

---

problem generation can be infeasible for large problems because it requires the use of a complete satisfiability testing algorithm, such as the Davis-Logemann-Loveland procedure [30], to filter unsatisfiable instances. Achlioptas et al. [29] recently showed how to generate hard satisfiable problems efficiently using quasigroups with holes.

4.2 Results

The performance is summarized in Table 1, showing the number of variables (n), fraction of runs that successfully resulted in satisfying solutions (success ratio), and mean, median, and standard deviation of flips (for GSAT) or rounds (for the other protocols).

n	Success Ratio	Mean	Median	σ
		Protocol: MS-O		
20	0.95	3.46×10^3	9.63×10^2	5.51×10^3
50	0.40	3.90×10^4	5.00×10^4	18.1×10^3
		Protocol: GSAT		
20	1.00	2.73×10^2	1.09×10^2	5.45×10^2
50	1.00	1.26×10^3	5.78×10^2	1.83×10^3
		Protocol: DB		
20	1.00	3.52×10^1	2.05×10^1	4.51×10^1
50	1.00	2.34×10^2	6.45×10^1	8.29×10^2

Table 1
Performance of protocols on Uniform Random 3-SAT problems at the phase transition, with n variables. GSAT is measured by flips, and all other protocols are measured by rounds.

The results from the 20-variable experiments show that MS-O can solve SAT problems, albeit significantly slower than GSAT or DB. For the 50-variable problems, MS-O can solve only 40% of the problems within the specified number of rounds, showing that MS-O is prohibitively slow for even moderately small problems. Note that for the 50-variable problems, a heavy tail on the upper end of the distribution of rounds was truncated because MS-O was stopped at the maximum rounds in 60% of the runs. Hence, if we were to always run MS-O until it found a solution, we would expect the mean and median measures to be much higher.

5 Variant MarketSAT Protocols

MS-O demonstrates that a market-inspired protocol can solve general combinatorial problems in a highly decentralized fashion. However, MS-O's unimpressive performance seems to bode poorly for the prospect that market mod-

els could provide a practical approach for decentralizing combinatorial problems. It is important that we not immediately discount the market approach with this evidence, as we might conceive of a myriad of other market, or market-inspired, approaches that may prove to be more effective.

To guide our exploration of alternate market protocols, we should identify aspects of MS-O that may contribute to its poor performance. A number of features of the task dependency network encoding of SAT problems may reduce performance. Because the assignments are represented as separate agents, the two assignment agents for a particular variable can coordinate only indirectly, through the auction for their shared variable, to determine which assignment is (tentatively) chosen. A more direct encoding might have a single agent represent a variable. Such a variable agent could directly determine the currently best assignment based on the price of licenses for both assignments, without the need to wait a bidding round for its variable auction to report a new price quote. [5] The presence of the overall assignment producer, π_c, and the end consumer, c, can also slow down MS-O, because bids and prices must generally pass through these agents before updates can be propagated to the license auctions. To see this, consider a state in which, for a particular variable u, neither assignment producer is winning the right to make an assignment to the variable, and price of g_c is higher than the current offer price of c. Neither assignment producer will update its bids for any license until first, c adjusts its offer price higher than the current bid of π_c for g_c, and then π_c increases its offer price for u above the offer price of one of the producers for assignments to u. This process could potentially take many rounds.

With these observations, we could formulate an alternate, simplified market representation of a SAT problem. As suggested above, we dispense with the overall assignment producer π_c and consumer c, as well as the variable goods g_u. As with the original MarketSAT, we make available $|q| - 1$ licenses to fail to satisfy clause q. We also assume that the license producers s_q^i exist as before. For each variable u we have a ***variable agent*** a_u that determines the assignment u. In order to select an assignment, agent a_u must obtain a license g_q for all clauses q its chosen assignment fails to satisfy. A selection of assignments by variable agents corresponds to a satisfying SAT assignment iff every agent acquires all the licenses necessary for its assignment. Figure 10 shows a simplified transformation of the same SAT instance shown transformed in Figure 3.

[5] This alternate encoding centralizes the information and control for a particular variable, as compared to the original task dependency network encoding. One might argue, though, that it is more natural to have agents represent variables, rather than assignments.

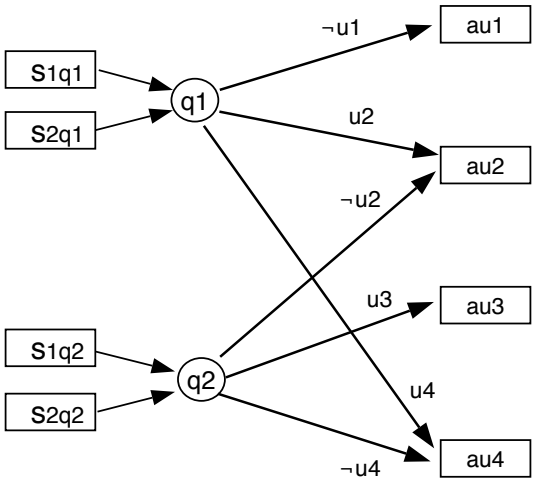

Fig. 10. A simplified transformation for the same SAT instance represented in Figure 3, $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

In addition to the simplified network structure, we also explore alternate auction mechanisms for the license goods, along with different bidding policies.

In the following sections we describe two variants on MS-O using the simplified SAT market structure described in this section. As with MS-O, agents submit bids to the auctions for licenses and the auctions respond with price quotes. Bidding occurs synchronously in rounds, and the protocols continue until quiescence. The license producers continue to place one-time sell offers with price zero, hence we focus on the bidding of the variable agents. As we see from the experimental results in Section 7.1, these variant protocols find solutions much more quickly than MS-O.

5.1 Uniform Pricing

The ***MarketSAT protocol with uniform pricing*** (MS-U) is based on MS-O, but adapted to the simplified structure. An auction allows an agent to place a bid containing an offer to buy one license at a specified price. A variable agent may update its bid only if it increases the price of the offer by at least some publicly known increment δ_b. Agents may not withdraw bids.

The auctions are identical to those in MS-O. Since, as in MS-O, the license producers s_q^i do not change their bids, the (M+1)st auction rules compute prices and (tentative) allocations based on the current offers from variable agents and the number of licenses available, $M = |q| - 1$. To be more concrete in the MarketSAT context, Figure 11 details the auction rules specifically for MS-U.

PROCEDURE **MS-U Report Price Quote** by the auction for good g_q,
a license to fail to satisfy clause q
IF there are more than $|q|-1$ buy offers THEN
$p(g_q) \leftarrow |q|$th highest price of all bids
$\alpha(g_q) \leftarrow (|q|-1)$st highest price of all bids
END
ELSE IF there are exactly $|q|-1$ buy offers THEN
$p(g_q) \leftarrow 0$
$\alpha(g_1) \leftarrow (|q|-1)$highest price of all bids
END
ELSE
$p(g_q)$ is undefined
$\alpha(g) \leftarrow 0$
END
Select the $|q|-1$ highest buy bids, breaking ties in favor of earlier bids,
and set $\omega_a(g) \leftarrow 1$ for each respective buyer a
$S =$ a set of randomly chosen sell bidders such that
$|S| \leftarrow |\{\omega_a(g) = 1 \mid a \text{ is a buyer}\}|$
FOR EACH $a \in S$ DO $\omega_a(g) \leftarrow 1$
Report $p(g)$, $\alpha(g)$, and $\omega_a(g)$ to each bidder a

Fig. 11. The report price quote procedure for the MS-U auction mechanism.

PROCEDURE **MS-U Variable Agent Update Bids** by agent a_u
representing variable u
$r_T \leftarrow$ previously computed assignment cost for $t(u) = T$
$r_F \leftarrow$ previously computed assignment cost for $t(u) = F$
FOR EACH input good g of a DO
IF $\omega_a(g) = 1$ THEN $\hat{p}_{a_u}(g_q) \leftarrow p(g)$
ELSE $\hat{p}_{a_u}(g) \leftarrow \max(\alpha(g),\ \alpha(g) + \delta_b)$
END
$G_T \leftarrow \{g_q \in G \mid \mathrm{FTS}(q, u, T)\}$
$G_F \leftarrow \{g_q \in G \mid \mathrm{FTS}(q, u, F)\}$
$t(u) \leftarrow \arg\max_{\gamma \in \{T,F\}}(\sum_{g \in G_\gamma} \hat{p}_{a_u}(g),\ r_\gamma)$, breaking ties in favor of
previous assignment
FOR EACH $g \in G_{t(u)}$ DO
IF $\omega_{a_u}(g) = 0$ THEN increment bid for g by δ_b
END
FOR EACH $g \in G_{\overline{t(u)}}$ DO bid zero for g

Fig. 12. The variable agent bidding policy for the MS-U protocol.

In MS-U, each producer places an initial bid of zero for its clause and does not update its bids. A variable agent randomly chooses the initial assignment for its variable and places offers at price zero for the necessary licenses. When it subsequently receives price quotes, it increments its losing bids for the assignment that maximizes its perceived cost, under the assumption that the cost

of an assignment does not decrease in subsequent rounds. The variable agent submits offers at prize zero for goods needed for the alternative assignment. Figure 12 details the variable agent update bids policy in MS-U. Observe that if the market reaches quiescence in this way, then the agents' local assignments constitute a globally satisfying assignment.

5.2 Differential Pricing

In the ***MarketSAT protocol with differential pricing*** (MS-D), an auction allows an agent to place a bid to demand either either zero or one units of the license, without specifying a price. Agents may switch between zero and one quantity demands without constraint.

PROCEDURE **MS-D Report Price Quote** by the auction for good g_q,
a license to fail to satisfy clause q
d = the total demand expressed by bids in the auction
IF $d < |q| - 1$ THEN
Report $p_a(g_q) \leftarrow 0$ to each bidder a
Report $\omega_w(g_q) \leftarrow 1$ to each bidder w that bid for one unit
Report $\omega_l(g_q) \leftarrow 0$ to each bidder l that bid for zero units
END
ELSE IF $d = |q| - 1$ THEN
Report $p_w(g_q) \leftarrow 0$ and $\omega_w(g_q) \leftarrow 1$ to each bidder w
that bid for one unit
Report $p_l(g_q) \leftarrow \rho(g_q)$ and $\omega_w(g_q) \leftarrow 0$ to the bidder l
that bid for zero units
END
ELSE IF $d > |q| - 1$ THEN
$\rho(g_q) \leftarrow \rho(g_q) + 1$
Report $p_l(g_q) \leftarrow \rho(g_q)$ and $\omega_w(g_q) \leftarrow 0$ to one randomly
chosen bidder l
Report $p_w(g_q) \leftarrow 0$ and $\omega_w(g_q) \leftarrow 1$ to each w of the other bidders
END

Fig. 13. The report price quote procedure for the MS-D auction mechanism.

An auction for good g may report a different price quote $p_a(g)$ to each bidder a, depending on the demand for the license. An auction maintains a nondecreasing ***premium price***, $\rho(g)$ for its license. When demand is less than the supply of licenses, the auction reports a price of zero to all bidders. When demand equals supply, the auction reports the premium to the one agent that did not demand the good to discourage it from creating excess demand. When there is excess demand, the auction increases the premium price and reports it to one of the agents to encourage that agent to reduce its own demand. Figure 13 details the auction rules for the MS-D protocol.

PROCEDURE **MS-D Variable Agent Update Bids** by agent a_u,
representing variable u in MS-D
$G_T \leftarrow \{g_q \in G \mid \mathrm{FTS}(q, u, T)\}$
$G_F \leftarrow \{g_q \in G \mid \mathrm{FTS}(q, u, F)\}$
$t(u) \leftarrow \arg\max_{\gamma \in \{T,F\}} \sum_{g \in G_\gamma} p_{a_u}(g)$, breaking ties in favor of previous assignment
FOR EACH $g \in G_{t(u)}$ DO bid for one unit of g
FOR EACH $g \in G_{\overline{t(u)}}$ DO bid for zero units of g
END

Fig. 14. The variable agent bidding policy for the MS-D protocol.

An agent randomly chooses the initial assignment for its variable. When it subsequently receives price quotes, it chooses an assignment that minimizes the sum of the prices of licenses as reported by the last price quote, with preference for its current assignment when the costs are equal. An agent bids to demand quantity one for the licenses for its currently chosen assignment, and quantity zero for its unneeded licenses. Figure 14 details the variable agent update bids policy for MS-D. Observe that if the market reaches quiescence in this way, then the agents' local assignments constitute a globally satisfying assignment. Furthermore, in this case every agent pays zero for the licenses it receives.

6 Price-Guided Search

6.1 Intuition for Price-Guided Search in MarketSAT Protocols

Intuitively, market prices indicate the relative global value of licenses, and agents use the prices to guide their local decisions. The bidding process can be seen as a distributed search for prices that support a satisfying allocation. In particular, the prices of certain licenses must rise sufficiently high relative to other licenses to block out variable assignments that cannot be a part of the solution.

In the MarketSAT protocols, prices are closely analogous to the weights of nogoods (which correspond exactly to failing to satisfy a clause) in the breakout algorithm [25], presented in Figure 15. In breakout, weights are a measure of how often the nogoods appear in local minima visited by the algorithm. The weights bias the search away from expensive nogoods, hence away from local minima.

In contrast to breakout, which increases the costs of nogoods only when the global state is a local minimum, the MarketSAT protocols increase the prices

Breakout Algorithm

```
Initialize the weight of all nogoods to 1
UNTIL current state is solution DO
      IF current state is not a local minimum
      THEN make any local change that reduces
            the total weights of violated nogoods
      ELSE increase weights of all currently
            violated nogoods
END
```

Fig. 15. The breakout algorithm [Morris, 1993].

of licenses in response to local state. The price of a license increases whenever the corresponding clause is not satisfied for the current assignment, which can and does occur outside of local minima of the global state. This aspect of MarketSAT may actually be beneficial, for Frank [31] reported that increasing weights of nogoods not only at local minima, but also when the nogood is not satisfied, can improve the performance of GSAT.

The breakout algorithm sequentially flips variable assignments to reduce the global weight of the violated nogoods. The nogood weights are counted only for clauses that are not satisfied. In MS-O and MS-U, because bids cannot be withdrawn, once a license has an excess demand, it will always have excess demand, even if the current choices of variable assignments would indicate that the clause is currently satisfied. Thus the prices never decrease and agents attribute a cost to a license even if the clause is satisfied. In contrast, agents can reduce their bid demands to zero in MS-D. As as result, an agent will only attribute a positive cost to a license if either the license is over-demanded or if flipping its own assignment would make the license over-demanded (assuming no other agent would also flip). In this sense, the cost evaluation in MS-D is closer to the breakout algorithm than in MS-O and MS-U.

In breakout, current nogood costs are evaluated uniformly for all variables. In MS-O and MS-U, an agent distinguishes its cost evaluation for licenses depending on whether it is winning (in which case it uses the bid price) or losing (in which case it uses the ask price or the ask price plus δ_b), and assumes that its total cost over all licenses never decreases. This difference in perceived prices gives extra "friction" to the currently winning agents because they assume lower costs, hence are somewhat less likely to swap assignments (recall that in MS-U, assignment swapping is done directly by variable agents, while in MS-O, it is determined by the variable auctions). However, this friction is relatively small because agents increase their offers by δ_b only when they are losing, hence the bid and ask prices never differ by more than δ_b. In MS-D, agents can attribute widely varying costs to a license, with at most one agent attributing the premium cost and others attributing a zero cost.

Unlike breakout, variables can flip simultaneously in the MarketSAT protocols. This could be beneficial to performance when agents operate in parallel, if the right flips occur simultaneously. Yokoo and Hirayama [5] observed that it could be detrimental if ***neighbor*** variables—those that share a clause—flip simultaneously, and thus incorporated synchronizing steps into DB to prevent simultaneous neighbor flips.

Of particular concern are ***simultaneous, satisfying neighbor flips***—neighbors simultaneously flipping to satisfy a clause (e.g., variables x and y simultaneously flipping to T to satisfy clause $(x \vee y)$), which we should expect to be prevalent in the MarketSAT protocols. When neighbor variables simultaneously flip to satisfy the same clause, other clauses may become needlessly unsatisfied. Unlike DB, the MarketSAT protocols have no explicit mechanism to prevent simultaneous neighbor flips, yet we should expect the different protocols to differ in the number of simultaneous, satisfying neighbor flips. In MS-O and MS-U, because the bid/ask spread on any license is small, agents that desire common licenses are likely to have similar total cost evaluations for licenses. Hence, when the price of a shared license increases, they may be likely to simultaneously flip to satisfy the clause. In MS-D, because agents can attribute widely varying costs to licenses, the cost evaluations of neighbor agents are less likely to be close than in MS-O and MS-U. We should expect this would reduce the number of simultaneous, satisfying neighbor flips.

6.2 Incompleteness of MarketSAT Protocols

We can show that MS-U is incomplete using a close variant of the example that Morris [25] used to show that the (centralized) breakout algorithm is incomplete. We assume synchrony, noting that this implies the same for an asynchronous system. The CNF clauses are: $(\bar{x} \vee y)$, $(\bar{x} \vee z)$, $(\bar{x} \vee w)$, $(\bar{y} \vee x)$, $(\bar{y} \vee z)$, $(\bar{y} \vee w)$, $(\bar{z} \vee x)$, $(\bar{z} \vee y)$, $(\bar{z} \vee w)$, $(\bar{w} \vee x)$, $(\bar{w} \vee y)$, $(\bar{w} \vee z)$. Observe that the only solution for the problem assigns all variables T or all variables F. Consider the case when the initial random assignments are $t(x) = t(z) = T$ and $t(y) = t(w) = F$. Each agent submits offers at price zero for all the clauses it fails to satisfy in its current assignment. Assume that the random tie breaking occurs as follows in the first round: x wins $(\bar{x} \vee y)$, y wins $(\bar{z} \vee y)$, z wins $(\bar{z} \vee w)$, and w wins $(\bar{x} \vee w)$. Tie breaking is not necessary in the other auctions because they all have zero or one offer.

After the initial round, all variable agents have a cost of one for their current assignments and zero for their other assignments. Hence, all agents flip assignments in the second round and submit offers at price zero for the clauses they fail to satisfy with the new assignments. Assume that the random tie breaking

occurs as follows in the second round: y wins $(\bar{y} \vee x)$, x wins $(\bar{w} \vee x)$, z wins $(\bar{y} \vee z)$, and w wins $(z \vee \bar{w})$.

After round two, all agents have a cost of two for their current assignment and a cost of one for their other assignment, hence choose to flip assignments. In all subsequent rounds, agents have the same cost difference between their current and alternate assignments, auction tie breaking is deterministic (i.e., in favor of earlier bids), and agents continue to flip assignments simultaneously in the same rounds. Hence, the agents never converge to all equal assignments, and the protocol oscillates indefinitely.

The particular behavior in this example relies on the earliest-bid tie-breaking rule. Although we have not systematically explored alternate tie-breaking rules, we know that they would engender different behavior in the protocol. For instance, with random tie breaking, MS-U would not necessarily fall into the exact oscillations we describe above. However, because random tie breaking *could* break ties just like earliest-bid tie breaking with positive probability, MS-U remains incomplete with random tie breaking.

With the same SAT instance we used to show incompleteness of MS-U (but with different initial assignments and tie breakings), in our simulations we observed oscillations in flipping of variables, strongly suggesting to us that MS-O is not guaranteed to converge to a solution. We refrain from describing a trace of a non-converging run due to the greater complexity of the agent interactions in MS-O.

We can also show that MS-D is incomplete using Moris's exact example. The CNF clauses are: $(x \vee y \vee z \vee w)$, $(\bar{x} \vee y)$, $(\bar{x} \vee z)$, $(\bar{x} \vee w)$, $(\bar{y} \vee x)$, $(\bar{y} \vee z)$, $(\bar{y} \vee w)$, $(\bar{z} \vee x)$, $(\bar{z} \vee y)$, $(\bar{z} \vee w)$, $(\bar{w} \vee x)$, $(\bar{w} \vee y)$, $(\bar{w} \vee z)$. Observe that the only solution assigns all variables T. Consider the initial random assignment t, such that $t(x) = T$ and all other variables are F. The premium increases in each of the unsatisfied clauses so long as they remain unsatisfied, but the choice of which variable in the clauses pays the premium is chosen randomly. With positive probability, x can always be the only variable chosen to pay the premium, in which case it will flip indefinitely and no other variables will flip.

If the breakout algorithm reaches truth assignment t as described in the preceding, it *cannot* converge to the satisfying truth assignment [25]. But because MS-D has variable-specific pricing, it does not necessarily make a flip when it would be a global improvement, which can delay undesirable flips. In fact, MS-D can always converge, with positive probability, to some satisfying truth assignment t^*, from any state. Consider an alternate, non-satisfying truth assignment t_i at round i of the protocol and a set of variables U_i such that $t_i(u) \neq t^*(u)$ for each $u \in U_i$. The prices will increase only for licenses corresponding to unsatisfied clauses. For these licenses, the auction will decide

randomly which variable receives a premium price quote. With positive probability the auctions will choose variables from only U_i. Whenever this happens, each $u \in U_i$ will have a higher cost for assignment $t_i(u)$, hence will flip to $t^*(u)$ and U_i is strictly reduced. Since we can apply this reasoning to successive rounds, it follows that, with positive probability, MS-D will reach a round j in which $U_j = \emptyset$, which must be a satisfying truth assignment. Hence, MS-D can converge to a satisfying truth assignment (when one exists) from any other truth assignment, with positive probability. We have shown that this does not hold for the other MarketSAT protocols, although we do not know if small modifications, such as to tie-breaking rules, would give us this property.

7 Evaluation of MarketSAT Variants

SAT algorithms are typically evaluated on their runtime performance, hence we experimentally compare our MarketSAT protocols with each other and with DB and GSAT in Section 7.1. While important, speed is not the only attribute of interest in our study of the protocols. In Section 7.2 we discuss the decentralization characteristics of the MarketSAT protocols and DB, and in Section 7.3 we consider the degree to which our market-*inspired* protocols lend themselves to a more rigorous economic interpretation. We identify a tradeoff between performance and economic realism in the market protocols, and a tradeoff between performance and the degree of decentralization between the market protocols and DB.

7.1 Performance Comparison of MarketSAT Variants

In this section we describe the results of experiments comparing all of the MarketSAT protocols to DB and GSAT. We construct the experiments as described in Section 4.1. In addition to the "Uniform Random 3-SAT" problems at the phase transition, we ran experiments on other classes of SAT problems from the SATLIB benchmark library. Recall that the Uniform Random 3-SAT problems at the phase transition are generally considered the hardest for SAT solvers. To evaluate performance on more structured problems, we ran experiments on SAT-encoded "Flat Graph Coloring" problems that are very hard to solve on average for graph coloring algorithms like the Brelaz heuristic [32]. We also ran experiments on "Controlled Backbone Size" problems, for which the difficulty is parameterized by the number of clauses and the backbone size. The backbone size is the number of entailed literals. (A literal is entailed by a satisfiable SAT instance iff that literal must be true for the SAT instance to be satisfied.) Singer et al. [33] showed that for the local search algorithm WSAT/SKC, runtime increases with a smaller number of clauses and a larger

backbone. To determine whether the MarketSAT protocols behave similarly, we ran one set of experiments with a fixed number of variables and varying backbone sizes, and another set with a fixed backbone size and varying numbers of clauses. We ran experiments on only the first 100 instances of each Controlled Backbone Size problem set.

The performance of all the protocols is summarized in Tables 2, 3, 4, 5, and 6, again showing fraction of runs that successfully resulted in satisfying solutions (success ratio), and mean, median, and standard deviation of flips (for GSAT) or rounds (for all other protocols), as well as problem size measures appropriate for the particular benchmark. In each table, the problems increase in difficulty from top to bottom. Comparing the protocols from top to bottom, each subsequent protocol improves by about a factor of two to ten on mean and median performance measures. A notable exception is performance on the larger Flat Graph Coloring problems, for which DB performs much better than MS-D in success and median measures. Our experiments with the Controlled Backbone Size problems demonstrate results (Tables 4, 5, and 6) qualitatively consistent with Singer et al., with runtime increasing with more clauses and smaller backbone.

It appears that the improvement of MS-U over MS-O is due to the simplified network structure, since MS-U is otherwise essentially the same as MS-O. One plausible conjecture for the difference between MS-U, MS-D, and DB is differing numbers of simultaneous, satisfying neighbor flips. A corresponding prediction would be that an ordering of the protocols by increasing simultaneous, satisfying neighbor flips would accord with their performance rank.

To test this conjecture we measured the simultaneous, satisfying neighbor flips in both MS-D and MS-U on Uniform Random 3-SAT problems. Also, recognizing that the MarketSAT protocols are much like breakout, but with simultaneous flips, we modified the centralized breakout algorithm to allow simultaneous flips (with some ad hoc parameters to control the number of simultaneous flips and simultaneous, satisfying neighbor flips). To gain insight about the effects of simultaneous, satisfying neighbor flips, we tested breakout with simultaneous flips (SFB) with varying fractions of such neighbor flips. We also varied the number of generic simultaneous flips to help distinguish their contribution to performance from neighbor flips.

Table 7 shows the mean number of flips, flips per flip round (flips per round in which a flip actually occurred), and fraction of simultaneous, satisfying neighbor flips for MS-U, MS-D, DB, and SFB. As expected, MS-U performs substantially more simultaneous, satisfying neighbor flips absolutely, and as a fraction of the total flips, than does MS-D. However, the results from SFB do not support the conjecture that the neighbor flips contribute significantly to the performance difference. The performance of SFB does not appear to

n	Success Ratio	Mean	Median	σ
		Protocol: MS-O		
20	0.95	3.46×10^3	9.63×10^2	5.51×10^3
50	0.40	3.90×10^4	5.00×10^4	18.1×10^3
		Protocol: MS-U		
20	1.00	2.66×10^2	1.07×10^2	4.86×10^2
50	0.96	6.12×10^3	1.51×10^3	1.15×10^4
75	0.75	2.75×10^4	1.08×10^4	3.03×10^4
100	0.53	6.10×10^4	8.21×10^4	4.16×10^4
		Protocol: GSAT		
20	1.00	1.32×10^2	4.00×10^1	2.31×10^2
50	1.00	1.26×10^3	5.78×10^2	1.83×10^3
75	1.00	4.36×10^3	2.15×10^3	5.33×10^3
100	0.99	1.30×10^4	5.38×10^3	1.94×10^4
125	0.94	2.46×10^4	1.10×10^4	3.23×10^4
		Protocol: MS-D		
20	1.00	7.20×10^1	4.05×10^1	9.17×10^1
50	1.00	8.96×10^2	2.50×10^2	3.52×10^3
75	0.98	3.98×10^3	4.29×10^2	1.17×10^4
100	0.96	1.04×10^4	1.50×10^3	2.34×10^4
125	0.85	2.74×10^4	3.65×10^3	4.45×10^4
150	0.85	3.69×10^4	5.94×10^3	5.45×10^4
175	0.83	5.37×10^4	1.63×10^4	6.68×10^4
		Protocol: DB		
20	1.00	3.52×10^1	2.05×10^1	4.51×10^1
50	1.00	2.34×10^2	6.45×10^1	8.29×10^2
75	0.99	2.14×10^3	2.99×10^2	9.40×10^3
100	0.98	4.26×10^3	4.60×10^2	1.61×10^3
125	0.96	9.12×10^3	1.42×10^3	2.63×10^4
150	0.93	1.80×10^4	1.22×10^3	4.24×10^4
175	0.88	2.98×10^4	2.83×10^3	5.81×10^4

Table 2
Performance of protocols on Uniform Random 3-SAT problems at the phase transition with n variables. GSAT is measured by flips, with all other protocols measured by rounds.

Vertices	Edges	n	Success Ratio	Mean	Median	σ
			Protocol: MS-U			
30	60	90	1.00	1.73×10^3	1.03×10^3	2.66×10^3
50	115	150	0.92	3.80×10^4	1.70×10^4	4.54×10^4
			Protocol: GSAT			
30	60	90	1.00	1.19×10^3	5.83×10^2	1.92×10^3
50	115	150	1.00	1.33×10^4	9.09×10^3	1.54×10^4
			Protocol: MS-D			
30	60	90	1.00	4.40×10^2	2.81×10^2	5.23×10^2
50	115	150	0.99	3.91×10^3	8.75×10^2	1.62×10^4
75	180	225	0.97	1.65×10^4	2.88×10^3	4.16×10^4
100	239	300	0.89	6.02×10^4	1.14×10^4	9.71×10^4
125	301	375	0.77	6.63×10^4	2.82×10^4	1.56×10^5
			Protocol: DB			
30	60	90	1.00	1.17×10^2	7.65×10^1	1.46×10^2
50	115	150	1.00	9.47×10^2	2.09×10^2	3.99×10^3
75	180	225	1.00	3.82×10^3	5.72×10^2	1.51×10^4
100	239	300	0.98	1.36×10^4	8.95×10^2	5.02×10^4
125	301	375	0.95	3.51×10^4	5.16×10^2	9.23×10^4

Table 3
Performance of the protocols on SAT-encoded Flat Graph Coloring problems with n variables and the specified number of vertices and edges GSAT is measured by flips, and all other protocols are measured by rounds.

be sensitive to, or even monotonic in, the fraction of simultaneous, satisfying neighbor flips.

Table 7 suggests an alternate explanation of the performance difference between DB and MS-D. For 50-variable problems, MS-D requires nearly 3.8 times as many rounds as DB, but only 2.2 times as many flips. We also measured the number of rounds in which MS-D and DB performed flips and found that, in fact, the average number of rounds in which variables actually flip in MS-D is 316, only 35% of the average total rounds. The average number of rounds in which DB performed a flip is 73% of its average total rounds. Thus it seems that the poor performance of MS-D relative to DB is due to the extra rounds required to produce flips.

Clauses	Backbone	Success Ratio	Mean	Median	σ
		Protocol: MS-O			
449	10	0.27	9.03×10^4	1×10^5	2.24×10^4
		Protocol: MS-U			
449	10	0.99	9.19×10^3	2.98×10^3	1.60×10^4
449	30	0.81	3.79×10^4	1.77×10^4	3.85×10^4
449	50	0.73	4.53×10^4	2.39×10^4	4.09×10^4
449	70	0.67	5.24×10^4	4.99×10^4	3.94×10^4
		Protocol: GSAT			
449	10	1.00	1.70×10^3	1.14×10^3	1.19×10^3
449	30	1.00	4.27×10^3	2.32×10^3	5.19×10^3
449	50	1.00	5.87×10^3	3.09×10^3	7.35×10^3
449	70	1.00	7.72×10^3	4.74×10^3	8.60×10^3
		Protocol: MS-D			
449	10	1.00	7.36×10^2	4.94×10^2	8.26×10^2
449	30	1.00	3.10×10^3	7.92×10^2	1.03×10^4
449	50	1.00	3.60×10^3	9.68×10^2	8.26×10^3
449	70	0.97	5.81×10^3	1.28×10^3	1.72×10^4
449	90	0.91	1.88×10^4	5.05×10^3	3.06×10^4
		Protocol: DB			
449	10	1.00	2.38×10^2	1.07×10^2	4.41×10^2
449	30	1.00	6.68×10^2	2.20×10^2	1.44×10^3
449	50	1.00	6.29×10^2	2.97×10^2	1.20×10^3
449	70	1.00	6.05×10^2	3.13×10^2	9.06×10^2
449	90	0.98	5.34×10^3	9.93×10^2	1.64×10^4

Table 4
Performance of protocols on Controlled Backbone Size problems with 100 variables, 449 clauses, and varying backbone sizes. GSAT is measured by flips, and all other protocols are measured by rounds.

Clauses	Backbone	Success Ratio	Mean	Median	σ
		Protocol: MS-O			
449	30	0.13	9.66×10^4	1×10^5	1.66×10^4
		Protocol: MS-U			
449	30	0.81	3.79×10^4	1.77×10^4	8.65×10^4
411	30	0.85	3.16×10^4	1.28×10^4	3.60×10^4
435	30	0.81	3.85×10^4	2.46×10^4	3.79×10^4
429	30	0.65	4.91×10^4	2.88×10^4	4.28×10^4
423	30	0.66	5.23×10^4	4.60×10^4	4.12×10^4
418	30	0.58	5.58×10^4	5.16×10^4	4.30×10^4
411	30	0.65	5.14×10^4	3.99×10^4	4.19×10^4
403	30	0.51	6.03×10^4	8.23×10^4	4.20×10^4
		Protocol: GSAT			
449	30	1.00	4.27×10^3	2.33×10^3	5.19×10^3
441	30	1.00	4.56×10^3	2.64×10^3	5.93×10^3
435	30	1.00	5.41×10^3	2.79×10^3	7.80×10^3
429	30	1.00	5.10×10^3	2.71×10^3	6.15×10^3
423	30	1.00	6.05×10^3	2.81×10^3	1.14×10^4
418	30	1.00	7.31×10^3	4.31×10^3	9.38×10^3
411	30	1.00	7.24×10^3	4.72×10^3	8.75×10^3
403	30	1.00	8.80×10^3	5.56×10^3	1.12×10^4

Table 5
Performance of MS-O, MS-U, and GSAT on Controlled Backbone Size problems with 100 variables, backbone size 30, and varying numbers of clauses. GSAT is measured by flips, and all other protocols are measured by rounds.

We conjecture that these extra rounds in MS-D happen because auctions randomly reassign the premium price at each round. Thus the agents' costs fluctuate randomly, and do not directly progress every time the premium increases. To verify this conjecture, we tested a variation of MS-D whereby pricing is reported as the premium and a fraction b of the premium (rather than the premium or zero as in MS-D). With a positive b, an agent's costs would not fluctuate so heavily from random assignments of the premium cost. We tried $b = 0.1$ on 50-variable problems and found that it required only 509 rounds, 462 flips, and 231 flip rounds. The ratio of rounds to flip rounds is only 2.2

Clauses	Backbone	Success Ratio	Mean	Median	σ
		Protocol: MS-D			
449	30	1.00	3.10×10^3	7.92×10^2	1.03×10^4
411	30	1.00	1.56×10^3	6.24×10^2	3.57×10^3
435	30	0.99	4.16×10^3	9.69×10^2	1.30×10^4
429	30	1.00	3.51×10^3	1.24×10^3	7.47×10^3
423	30	1.00	4.45×10^3	1.15×10^3	1.10×10^4
418	30	0.99	4.87×10^3	1.50×10^3	1.19×10^4
411	30	0.97	5.40×10^3	7.79×10^2	1.74×10^4
403	30	0.96	7.67×10^3	1.21×10^3	2.01×10^4
		Protocol: DB			
449	30	1.00	6.86×10^2	2.20×10^2	1.44×10^3
441	30	1.00	1.20×10^3	2.16×10^2	7.66×10^3
435	30	1.00	5.89×10^2	2.04×10^2	2.10×10^3
429	30	1.00	8.25×10^2	2.72×10^2	1.77×10^3
423	30	1.00	9.64×10^2	3.32×10^2	1.91×10^3
418	30	0.99	1.76×10^3	4.61×10^2	9.93×10^3
411	30	1.00	1.32×10^3	3.71×10^2	4.31×10^3
403	30	1.00	2.17×10^3	4.27×10^2	6.25×10^3

Table 6
Performance of MS-D and DB on Controlled Backbone Size problems with 100 variables, backbone size 30, and varying numbers of clauses. The protocols are measured by rounds.

with $b = 0.1$, compared to 2.8 for MS-D. Furthermore, although the variant with $b = 0.1$ outperformed MS-D, the fraction of satisfying neighbor flips was 0.43—significantly more than in MS-D. This evidence suggests that difference in performance between MS-D and DB is largely due to the extra rounds required to produce flips in MS-D, rather than simultaneous, satisfying neighbor flips.

Appealing to extra, non-flipping rounds does not seem to explain the relative performance difference between MS-U and MS-D, as the ratio of rounds to flip rounds is only 1.3 for MS-U—significantly less than for MS-D. An alternate explanation we propose is that the relatively poor performance of MS-U is due to the fact that costs continue to be attributed to a license when its

Protocol	Rounds	Flips	Flips Per Flip Round	Neighbor Flips Ratio
MS-U	6.12×10^3	1.02×10^4	2.21	0.33
MS-D	896	473	2.02	0.23
DB	234	218	1.41	0
SFB	483	329	1.00	0
	403	374	1.66	0
	494	390	1.42	0.02
	474	413	1.53	0.10
	483	432	1.65	0.13
	410	340	1.66	0.23
	408	468	1.99	0.37

Table 7
Comparison of simultaneous, satisfying flips for 50-variable Uniform Random 3-SAT problems.

respective clause becomes satisfied (note that prices are *increased* only for unsatisfied clauses though). If the prices do not distinguish between satisfied and unsatisfied clauses, it would seem that MS-U pricing may not provide a effective indication of the relative difficulty of satisfying a clause. In contrast, recall that, in MS-D, when a clause is satisfied, the agents currently demanding the associated license receive price quotes of zero. Breakout attributes no cost to satisfied clauses. To test whether attributing costs to satisfied clauses can be detrimental, we modified the breakout algorithm so that it does just that and found that the algorithm rarely found satisfying assignments. Of course MS-U did not perform this poorly, but it does differ from breakout in other ways, as described in Section 6. Still, this test suggests that we have identified a significant cause of the relatively poor performance of MS-U.

7.2 On Decentralization

The MarketSAT protocols are highly decentralized in the sense that agents need only know about and communicate with auctions for their own licenses (which in turn requires knowledge about the clauses in which they are contained), and auctions need communicate only with the agents that participate in them. Agents need not communicate with, or even know the existence of agents for other variables. Similarly, auctions need not communicate with each

other. In DB, an agent must know in which clauses its variable is contained and must also communicate with all variables in those clauses. MarketSAT can operate fully asynchronously. In DB, variables must synchronize with their neighbors to detect quasi-local minima and to ensure that neighbor variables do not flip simultaneously.

Decentralization based on prices may offer particular advantages in some contexts. For instance, to extend the method from satisfaction to optimization problems, it is often helpful to quantify the value of relaxing constraints. Prices support tradeoff resolution by placing conflicting alternatives on a common scale. Markets provide a natural approach to derive prices in a decentralized manner, from the distributed interactions of elements responsible for distinct components of a problem.

The experiments with MS-D suggest that a highly decentralized market-inspired protocol can actually perform quite well. Indeed, it performed comparably to the centralized GSAT algorithm and its performance was within a factor of four worse than DB. Experiments with variants on MS-D suggest that even better performance can be obtained with the same degree of decentralization. Still, it is an open question whether decentralized approaches could perform as well as the best centralized SAT algorithms. Although we can presently meet the performance of GSAT, much improved centralized algorithms have been developed since the advent of GSAT. Centralized algorithms can utilize techniques such as restarts (which contributed significantly to the performance of GSAT and subsequent hill-climbing based algorithms) and random flips (e.g., as in WalkSAT [34]) to help reduce heavy tails in the performance distribution. We found that restarts can significantly improve the performance of MS-D. For example, with $10n$ max flips and 1000 max restarts, MS-D can solve all 175-variable problem instances within the bound, with average rounds 2.69×10^4, median 1.1×10^4, and $\sigma = 5.00 \times 10^4$. Although restarts could be implemented in a synchronous system with cooperating agents, it is not obvious how such techniques might be utilized in a distributed, asynchronous system. Moreover, we do not have any intuition for an economic interpretation of restarts. With respect to random flips, it would be straightforward for agents to independently flip randomly in some fraction of the bidding rounds, but again it is not clear how to interpret this in economic terms.

7.3 Economic Interpretation and Rational Behavior

The interpretation of the market-inspired protocols in economic terms requires a model of agent values under which the assumed behavior would be plausibly rational. In MS-O, we assume that the consumer places value v_c on achieving

a satisfying allocation, and is willing to pay money for the allocation. The producers and suppliers seek to acquire surplus profit for their participation.

In MS-U and MS-D, since there is no consumer, there is not a single source of value for obtaining a satisfying solution. Furthermore since variable agents do not provide outputs to other agents, they have no potential source of income to make a monetary profit. Rather, for a variable agent a_u to be willing to participate in these protocols, and hence be willing to pay money for their allocations, it must obtain some individual value v_{a_u} from participating in a satisfying solution. A variable agent obtains no value if it does not participate in a satisfying solution. A variable agent wishes to maximize its surplus value, which is the difference between the value it obtains and the total price it pays for the licenses it acquires.

In all protocols, the bidding policies are non-strategic in that agents do not account for the effects of their behavior on the prices or allocations. This assumption may be reasonable in large networks for which individual agents have little effect on the outcome, or in uncertain situations where their effect is unpredictable [35,36]. However, as noted above, a deeper strategic analysis would be required to establish fully rational behavior.

In MS-U and MS-D, the bidding policies can be considered myopic and best-response in that agents always bid to optimize their surplus given the current price quotes, without speculating about future price changes. In MS-O, the producers are myopically bidding to obtain at least zero surplus, based on current prices. The plausibility of these approaches depends on how accurately the price quotes indicate the prices agents may actually have to pay for their final allocations. For all protocols, the price quotes indicate what the agents would pay if the bidding stopped in the current state.

However, the protocols differ significantly in how price quotes signal future price movements. In MS-O and MS-U, the nondecreasing price quotes indicate lower bounds on the amount that agents would have to pay, providing a basis for agents' belief in the price quotes. Since prices do not decrease, in MS-U we would also expect that, in addition to the agent quiescence condition specified in Section 5.1, a rational variable agent would stop bidding when the total cost of an assignment exceeds v_{a_u}.

In MS-D, agents actually pay nothing for a globally satisfying allocation, and would have positive payments only if the protocol were to terminate with an unsatisfying allocation. Thus, unlike in MS-U, we would expect rational agents to stop bidding only when they reach a satisfying allocation (as specified in Section 5.2). But this calls into question the usefulness of the price quotes as meaningful future price indicators. The bidding policies in MS-D would be more plausibly rational if the agents had a reasonable expectation

that the protocol could terminate at any time, for instance if the auctions terminated negotiations based on a random signal. Indeed, such a mechanism may be useful both to encourage desirable behavior and to bound the length of negotiations.

8 Conclusions

We showed that supply chain formation is NP-complete by transforming SAT problems into task dependency networks. We also showed that we can then solve SAT problems using a distributed market protocol for supply chain formation. That a market protocol designed for other purposes can be fairly directly applied to SAT problems provides existential evidence for highly decentralized market-inspired solution methods to general classes of combinatorial problems. Moreover, it is intriguing that the original MarketSAT protocol succeeds without explicit search state or randomization. However, in practice, the performance of original MarketSAT, based directly on task dependency networks, is impractically slow.

We can improve the performance of the MarketSAT protocols by simplifying the market structure. The pricing method also significantly affects performance, with the differential pricing protocol roughly seven times better than uniform pricing, and comparable in performance to GSAT. However, the differential pricing protocol is less justifiable in terms of rational economic agent behavior. Although the variant MarketSAT protocols perform significantly better than the original MarketSAT, the less decentralized distributed breakout algorithm still outperforms the differential pricing MarketSAT by a factor of three to four. We found that the fraction of simultaneous, satisfying neighbor flips does not explain the difference in performance across MarketSAT protocols, as originally conjectured. However, the evidence suggests that MarketSAT with uniform pricing performs worse than with differential pricing because the former attributes costs to satisfied clauses while the latter does not. Evidence further suggests that distributed breakout solves problems faster than MarketSAT with differential pricing because the former requires fewer rounds to produce a flip.

An informal analysis suggests that the price-guided search of the MarketSAT protocols works because the protocols resemble the centralized breakout algorithm. We might cautiously apply this intuition to the operation of SAMP-SB in general task dependency networks, but further analysis is necessary to properly account for a broader class of structures.

We have identified tradeoffs in terms of runtime performance, decentralization, and the plausibility of assumed agent behaviors. Understanding these tradeoffs

is necessary to make informed engineering decisions about the appropriateness and applicability of alternate decentralized approaches to a particular problem environment.

The market approach has the benefit of providing a price-based interface for an agent to evaluate and direct its behavior in the context of its broader decision making. To better understand and further develop market approaches to complex coordination problems, we must explicitly incorporate a model of the agents' economic motivations in the context of the problem to be solved. Future work should also investigate extension of the protocols to optimization problems, and include a deeper analysis of strategic agent behavior.

Acknowledgments

The ideas for the MS-U and MS-D protocols were developed during a visit to the University of Michigan by the second author. The authors wish to thank NTT and Edmund H. Durfee for supporting the visit. Much of this work was performed while the first author was supported by a NASA/Jet Propulsion Laboratory Graduate Student Researcher Fellowship at the University of Michigan. This work was also supported in part by National Science Foundation Grant IIS-9988715.

References

[1] W. E. Walsh, M. P. Wellman, MarketSAT: An extremely decentralized (but really slow) algorithm for propositional satisfiability, in: Seventeenth National Conference on Artificial Intelligence, 2000, pp. 303–309.

[2] W. E. Walsh, M. Yokoo, K. Hirayama, M. P. Wellman, On market-inspired approaches to propositional satisfiability, in: Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 1152–1158.

[3] M. Yokoo, Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems, Springer, 2000.

[4] M. Yokoo, K. Hirayama, Algorithms for distributed constraint satisfaction: A review, Autonomous Agents and Multi-Agent Systems 3 (2) (2000) 198–212.

[5] M. Yokoo, K. Hirayama, Distributed breakout algorithm for solving distributed constraint satisfaction problems, in: Second International Conference on Multi-Agent Systems, 1996, pp. 401–408.

[6] F. Ygge, H. Akkermans, Decentralized markets versus central control: A comparative study, Journal of Artificial Intelligence Research 11 (1999) 301–333.

[7] F. Ygge, H. Akkermans, Power load management as a computational market, in: Second International Conference on Multi-Agent Systems, 1996, pp. 393–400.

[8] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, S. Stornetta, Spawn: A distributed computational economy, IEEE Transactions on Software Engineering (1992) 103–117.

[9] M. P. Wellman, A market-oriented programming environment and its application to distributed multicommodity flow problems, Journal of Artificial Intelligence Research 1 (1993) 1–23.

[10] D. M. Pennock, M. P. Wellman, Representing aggregate belief through the competitive equilibrium of a securities market, in: Thirteenth Conference on Uncertainty in Artificial Intelligence, 1997, pp. 392–400.

[11] Y. Shoham, M. Tennenholtz, On rational computability and communication complexity, Games and Economic Behavior 35 (2001) 197–211.

[12] P. A. Samuelson, Market mechanisms and maximization, Research memorandum, RAND, available in *The Collected Scientific Papers of Paul A. Samuelson*, Stiglitz, J. E., ed., 1966 (1949).

[13] J. Q. Cheng, M. P. Wellman, The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes, Computational Economics 12 (1998) 1–24.

[14] J. M. Crawford, L. D. Auton, Experimental results on the crossover point in random 3-SAT, Artificial Intelligence 81 (1996) 31–57.

[15] B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Tenth National Conference on Artificial Intelligence, 1992, pp. 440–446.

[16] W. E. Walsh, M. P. Wellman, A market protocol for decentralized task allocation, in: Third International Conference on Multi-Agent Systems, 1998, pp. 325–332.

[17] W. E. Walsh, M. P. Wellman, Efficiency and equilibrium in task allocation economies with hierarchical dependencies, in: Sixteenth International Joint Conference on Artificial Intelligence, 1999, pp. 520–526.

[18] W. E. Walsh, M. P. Wellman, F. Ygge, Combinatorial auctions for supply chain formation, in: Second ACM Conference on Electronic Commerce, 2000, pp. 260–269.

[19] W. E. Walsh, Market protocols for decentralized supply chain formation, Ph.D. thesis, University of Michigan (2001).

[20] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[21] M. A. Satterthwaite, S. R. Williams, Bilateral trade with the sealed bid k-double auction: Existence and efficiency, Journal of Economic Theory 48 (1989) 107–133.

[22] M. A. Satterthwaite, S. R. Williams, The Bayesian theory of the k-double auction, in: Friedman and Rust [37], Ch. 4, pp. 99–123.

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

[24] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, Journal of Finance 16 (1961) 8–37.

[25] P. Morris, The breakout method for escaping from local minima, in: Eleventh National Conference on Aritificial Intelligence, 1993, pp. 40–45.

[26] P. Cheeseman, B. Kanefsky, W. M. Taylor, Where the *really* hard problems are, in: Twelfth International Joint Conference on Artificial Intelligence, 1991, pp. 331–337.

[27] T. Hogg, B. A. Huberman, C. P. Williams, Phase transitions and the search problem, Artificial Intelligence 81 (1996) 1–15, editorial introduction for special issue on Frontiers in Problem Solving: Phase Trasitions and Complexity.

[28] D. Mitchell, B. Selman, H. Levesque, Hard and easy distributions of SAT problems, in: Tenth National Conference on Artificial Intelligence, 1992, pp. 459–465.

[29] D. Achlioptas, C. Gomes, H. Kautz, B. Selman, Generating satisfiable problem instances, in: Seventeenth National Conference on Artificial Intelligence, 2000, pp. 256–261.

[30] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Communications of the ACM 5 (1962) 394–397.

[31] J. Frank, Learning short-term weights for GSAT, in: Fifteenth International Joint Conference on Artificial Intelligence, 1997, pp. 384–389.

[32] T. Hogg, Refining the phase transition in combinatorial search, Artificial Intelligence 81 (1996) 127–154.

[33] J. Singer, I. Gent, A. Smaill, Backbone fragility and the local search cost peak, Journal of Artificial Intelligence Research (2000) 235–270.

[34] B. Selman, H. A. Kautz, B. Cohen, Noise strategies for improving local search, in: Twelfth National Conference on Artificial Intelligence, 1994, pp. 337–343.

[35] T. Sandholm, F. Ygge, On the gains and losses of speculation in equilibrium markets, in: Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997, pp. 632–638.

[36] M. P. Wellman, J. Hu, Conjectural equilibrium in multiagent learning, Machine Learning 33 (1998) 179–200.

[37] D. Friedman, J. Rust (Eds.), The Double Auction Market: Institutions, Theories, and Evidence, Addison-Wesley, 1993.