To appear in *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.

MarketSAT: An Extremely Decentralized (but Really Slow) Algorithm for Propositional Satisfiability

William E. Walsh   Michael P. Wellman
University of Michigan Artificial Intelligence Laboratory
1101 Beal Avenue, Ann Arbor, MI 48109-2110 USA
{wew, wellman}@umich.edu

Abstract

We describe MarketSAT, a highly decentralized, market-based algorithm for propositional satisfiability. The approach 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. MarketSAT employs a market protocol for general supply chain problems, which we show to be expressively equivalent to 3SAT. Experiments suggest that MarketSAT reliably converges to market allocations corresponding to satisfiable truth assignments. We experimentally compare the computational performance with GSAT, a centralized local search algorithm.

Introduction

Decentralization comprises constraints on the distribution of information and authority among participants in a distributed system. 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, and apparently of the system as a whole.

Because computational environments are increasingly decentralized in some respects (e.g., multiagent systems, where agents represent distinct individuals or organizations with diverse information and interests), it is important to understand the computational properties of decentralized systems. To do so, we require an appropriate model of decentralized computation.

Markets provide one 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 (Fujishima, Leyton-Brown, & Shoham 1999; Sandholm 1999; Wurman, Wellman, & Walsh 1998; Ygge & Akkermans 1998).

Shoham and Tennenholtz (to appear) 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 (1949) 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 (Cheng & Wellman 1998). However, none of these lines of analysis provide answers with respect to the sort of combinatorial optimization problems of most interest in AI research.

To address this gap, we examine here the possibility of using markets to solve propositional satisfiability (SAT) problems in a decentralized manner. As the original NP-complete problem, SAT (or its equivalently difficult special case, 3SAT) 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 (Crawford & Auton 1996), and steady improvements in centralized algorithms, starting with the success of GSAT (Selman, Levesque, & Mitchell 1992).

Our market approach to satisfiability employs a market protocol we found to be effective in decentralized supply-chain formation problems (Walsh & Wellman 1998; 1999). We formulate satisfiability (3SAT) problems in terms of a supply chain, and investigate the effectiveness of the market in solving them. The market protocol applied to these problems is what we refer to as ***MarketSAT***.

We describe the supply chain formation problem in the next section. We show how to transform 3SAT to supply chain formation in the "Supply Chain Formation is NP-Complete" section. We discuss the role of prices in guiding decentralized supply chain formation in the "Decentralization and Prices" section. Next we describe a distributed approach to supply chain formation in the "Market Protocol" section. We compare MarketSAT to GSAT in "MarketSAT

Copyright © 2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Experiments". Finally, we discuss work on distributed constraint satisfaction and our conclusions.

Supply Chain Formation

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 (Walsh & Wellman 1999). The term "good" refers to any discrete resource or task for which the results cannot be shared between agents.

More precisely, we (1999) formulate the problem as follows. 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 $A = C \cup \Pi \cup S$ is the set of agents, comprised of consumers C, producers Π, and suppliers S. 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. A ***supplier*** can supply a set of goods, up to some maximum quantity for each, without requiring any input goods.

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 and suppliers 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.

Definition 1 (supply chain formation problem) *(*SUPP-CHAIN*).*

Instance: *A task dependency network, (V,E), with agents, goods, and edges as described above.*

Question: *Is there a solution $(V',E') \subseteq (V,E)$?*

Supply Chain Formation is NP-Complete

To develop our market approach to solving satisfiability problems, we show that SUPP-CHAIN is NP-complete by a reduction transformation from 3SAT. The transformation combined with the protocol described in the next section provide the desired solution method.

Definition 2 (3satisfiability) *(*3SAT*)*

Instance: *Set U of variables, and collection Q of clauses, where each $q \in Q$ is a set of literals over U, and $|q| = 3$.*

Question: *Is there a truth assignment $t : U \to \{T,F\}$ that satisfies each $q \in Q$?*

Theorem 1 SUPP-CHAIN *is NP-complete.*

Proof concept. We say that a variable u ***fails to satisfy*** a clause q under truth assignment t, iff either: (1) $t(u) = T$, $u \notin q$, and $\overline{u} \in q$, *or*, (2) $t(u) = F$, $\overline{u} \notin q$, and $u \in q$. The key observation behind our reduction is that in a satisfying truth assignment, at least one variable must satisfy any given clause, hence at *most* two variables can *fail to satisfy* any given clause. 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 two are available per clause) to be allocated.

The task dependency network corresponding to a 3SAT 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: supplier of licenses to fail to satisfy q
- π_u: producer of a positive assignment to u
- $\pi_{\overline{u}}$: 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 3SAT to an instance of SUPP-CHAIN.

1. For each $q \in Q$, add g_q to G, add s_q to S, and add $\langle s_q,g_q \rangle_1$ and $\langle s_q,g_q \rangle_2$ 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 $\overline{u} \in q$, add $\langle g_q,\pi_u \rangle$ to E,
   - add $\pi_{\overline{u}}$ to Π, add $\langle \pi_u,g_u \rangle$ to E, and for each $q \in Q$ such that $\overline{u} \notin q$ and $u \in q$, add $\langle g_q,\pi_{\overline{u}} \rangle$ to E.
3. Add c to C, $\langle g_c,c \rangle$ to E, g_c to G, 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.

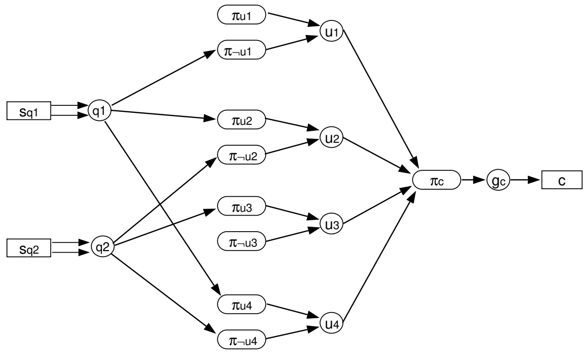


Figure 1: The transformation for $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

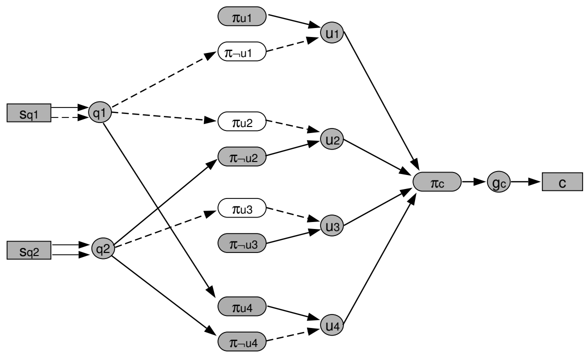


Figure 2: A solution that corresponds to the satisfying truth assignment $t(u_1) = T$, $t(u_2) = F$, $t(u_3) = F$, $t(u_4) = T$, for the problem in Figure 1. Shaded vertices and solid edges are in the solution.

Figure 1 shows the transformation for an example with $Q = \{q_1, q_2\}$, $q_1 = \{u_1, \bar{u}_2, \bar{u}_4\}$, $q_2 = \{u_2, \bar{u}_3, u_4\}$.

A "yes" configuration for 3SAT *is a "yes" configuration for* SUPP-CHAIN. If t is a satisfying truth assignment in the instance of 3SAT, 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'.

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 s_q to V' and a new $\langle s_q, g_q \rangle_i$ 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 two variables that fail to satisfy q, hence at most two producers require g_q as input in (V', E'). Hence, (V', E') is feasible. It is also a solution because $\langle \pi_c, g_c \rangle \in E'$.

Figure 2 shows a feasible allocation that corresponds to the satisfying truth assignment $t(u_1) = T$, $t(u_2) = F$, $t(u_3) = F$, $t(u_4) = T$, for the problem in Figure 1.

A "yes" configuration for SUPP-CHAIN *is a "yes" configuration for* 3SAT. 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 two units of any g_q available, each of which is in material balance, there are at most two 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 two variables in $q \in Q$ that fail to satisfy q, and hence at least one variable that does satisfy q under t. Thus, t is a satisfying truth assignment.

The transformation runs in polynomial time. The number of goods, agents, and edges incident thereof is polynomial in the 3SAT size. □

Decentralization and Prices

Given full knowledge of a 3SAT problem, we could employ a centralized algorithm known to be effective (e.g. GSAT). But in a decentralized system, agents need incentives to participate. In the market framework we model this in terms of recovering costs and acquiring value. Generally, supplier s has some ***opportunity cost*** $oc_s(\{g\})$ for supplying one unit $\langle s, g \rangle$ of good g. We assume $oc_s(\{g\}) = 0$ for all $s \in S$ and $\langle s, g \rangle \in E$. The consumer c obtains ***value*** $v_c(\{g_c\})$ for obtaining a single unit of good g_c.

We further assume the problem is decentralized in that each agent has knowledge only of its goods of interest and its valuations or costs thereof. In the market approach, we posit a ***price system*** p, which assigns to each good g a non-negative number $p(g)$ as its ***price***. Intuitively, prices indicate the relative global value of the goods. Therefore, agents may use the prices as a guide to their local decision making. We assume that each agent wishes to maximize its ***surplus***, that is the difference between its values (for consumers, from goods obtained, or for producers, from the price of goods sold) and costs incurred (for suppliers, opportunity costs of goods provided, or for producers, total price of inputs acquired), while maintaining feasibility.

We allow the existence of mediators for each good to facilitate indirect communication between agents, via prices. Thus an agent is constrained to exchange messages with mediators for its goods of interest based on its own valuations or costs and the history of price messages received from the mediators.

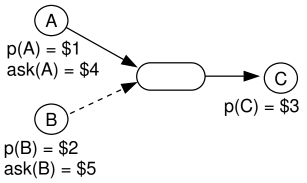
We describe particular protocol for forming supply chains, subject to the decentralization constraints, in the next section.

Market Protocol

In previous work (1999) we defined a market protocol for the supply chain formation problem. MarketSAT is simply this protocol applied to 3SAT 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. Here we assume reliable synchronous message passing, although all results pertaining to general task dependency networks apply to the asynchronous case as well (1998; 1999).

Auction Mechanism

The task allocation market includes a separate auction for each good of interest. Agents submit bids for goods they wish to buy or sell. A ***bid*** specifies the price below/above



Good	Current Bid	Next Bid
A	$2	hold at $2
B	$1	$2
C	$4	$6

Figure 3: Example of how a producer updates its bids in a state where it is winning A and C, with $\delta = \$1$.

which the agent is willing to buy/sell. Auctions respond with ***price quotes*** specifying the current going price and the number of units the recipient would trade at what price, given the current bid state. Agents may in turn respond with further bids. Each auction requires that an agent's successive bids increase by no less than some (typically small) positive increment δ.

This process proceeds synchronously in rounds. That is, all agents submit bids within a round, and then receive price quotes before the next round. When the market reaches ***quiescence***—a state in which no new bids or price quotes are issued—the auctions ***clear***. Each bidder is notified of the final prices and how many units it transacted in each good.

According to the the (M+1)st-price rules (Satterthwaite & Williams 1989; Wurman, Walsh, & Wellman 1998), an auction balances reported supply and demand at a uniform price. If there are M units of a good offered for sale, based on the current bids, then the price of a good is the $M+1$st price of *all* bids in the auction. Winners include all buyers/sellers strictly above/below the price, and, to maximize the benefits from trade, some agents at the clearing price. Ties are broken in favor of bids received in earlier rounds, and randomly among tied bids within a round.

When issuing price quotes, the auction reports both the price, $p(g)$, and the ***ask price***, $ask(g)$ 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.

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.

Let the current going prices be p. Supplier s places a one-time bid of $oc_s(\{g\})$, for each unit $\langle s,g\rangle \in E$ it can supply. When consumer c is not winning g_c it increases its bid for g_c to $p(g_c)+\delta$ if $v_c(\{g_c\}) - p(g_c) - \delta \geq 0$, otherwise it stops bidding.

A producer π initially bids zero for each input g, and increments its bid on an g whenever it is winning its output but losing g. π initially bids zero for its output g_π, and then changes the bid in an attempt to recover the ***perceived costs*** of its inputs. If π is currently winning an input g, its perceived cost, $\hat{p}_\pi(g)$ of g is simply $p(g)$. When π is not currently winning g, $\hat{p}_\pi(g) = ask(g)$ if $ask(g) > p(g)$ and $\hat{p}_\pi(g) = ask(g)+\delta$ if $ask(g) = p(g)$ (in the latter case, the producer must bid strictly above $ask(g)$ to win the good). When the prices of its inputs change, π bids $\sum_{\langle g,\pi\rangle\in E} \hat{p}_\pi(g)$ for g_π. Figure 3 shows an example of how a producer updates its bids.

Properties

A solution in which all consumers, suppliers and *active* producers have nonnegative surplus is called a ***valid solution***. The quiescent state of the protocol is a valid solution iff some consumers acquire desired goods (which implies that no *in*valid solutions are reached) (Walsh & Wellman 1998).

Because bids are strictly ascending and the consumer has an upper bound on its value for goods, the market protocol is guaranteed to reach quiescence. Moreover, the runtime is bounded by a polynomial function of the network size and the size of the consumer value.

Theorem 2 *In the worst case, MarketSAT reaches quiescence after a number of bidding rounds polynomial in the network size and the size of the consumer value.*

Proof. (See the appendix for Lemmas.) There is one initial bid for each edge. By Lemma 4 and the fact that agents increase buy bids by at least δ, the number of subsequent buy bids is polynomial. Lemmas 5 and 6 prove the same for sell bids. At least one bid is placed in each round. □

Extensive simulations on generic, random task dependency networks (with a slightly different version of the protocol) led us to conjecture that the protocol always converges to a valid solution if one exists and if some consumer has a sufficiently high value for a good (1998). As described in the next section, experiments support this conjecture for MarketSAT. It turns out that the size of $v_c(\{g_c\})$ can effect a tradeoff between worst-case runtime and the probability of successfully finding a satisfying assignment.

Corollary 3 (to Theorem 2) *The minimum consumer value necessary to ensure solution convergence in MarketSAT cannot be bounded by a polynomial function of the* 3SAT *instance problem size (assuming P $\neq$ NP).*

Although we do not yet have a proof of solution convergence, we have some intuition for how it works. The bidding process can be seen as a distributed search for a set of prices such that the agents in a solution would choose to be in the solution while other agents would choose to be out of the solution. In particular, the prices of certain goods of type g_q must rise sufficiently high relative to other such goods to block out variable assignments that cannot be a part of the solution.

Recall that a producer increases an input bid only when it is winning its output but losing that input. In this way, a variable assignment producer has the opportunity to affect the price space search to support its tentative inclusion in the solution. But as the prices on its inputs rise, it will raise its output bid in response. If these prices rise too high, the opposite assignment may be tentatively included in the solution instead, reflecting the collective market evaluation that the opposite assignment is currently the better for which to search.

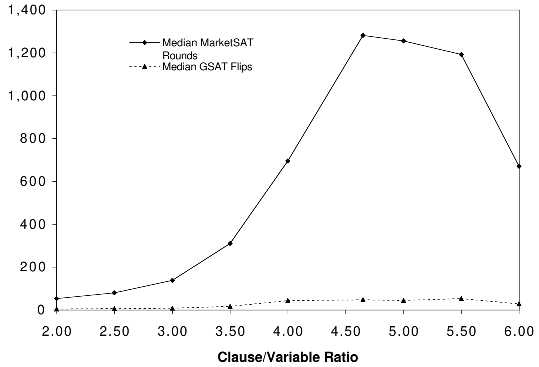


Figure 4: Median runtimes for 20-variable problems.

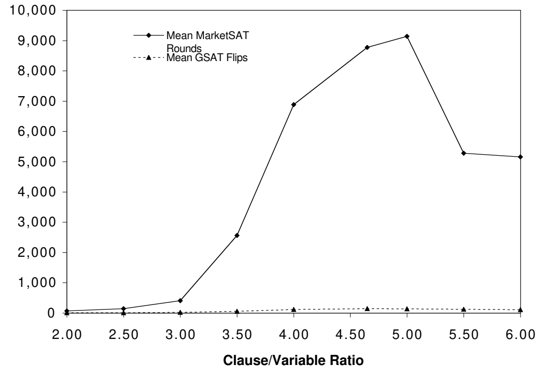


Figure 5: Mean values for 20-variable problems.

	Standard Deviation	Mean/ Median	Max/ Median
MarketSAT	$49,694$	741	6.8
GSAT	231	41	3

Table 1: Comparison of MarketSAT and GSAT for 20-variable problems at the crossover point.

MarketSAT Experiments

In this section we describe the construction and results of a battery of experiments comparing MarketSAT to GSAT (Selman, Levesque, & Mitchell 1992), a centralized random-restart hill-climbing algorithm.

Construction

We ran the two algorithms on a set of problems chosen according to the random 3SAT model (Mitchell, Selman, & Levesque 1992). Given a set of variables U, a clause contains three variables chosen uniformly and independently from U, each negated with probability .5. We eliminated unsatisfiable problems.

We tried problems with 15 and 20 variables, and varying numbers of clauses. Mitchell et al. (1992) observed that the hardest problems occur at a crossover point in the clause/variable ratio where 50% of the problems are satisfiable. Crawford and Auton (1996) found experimentally that the transition occurs when the number of clauses is approximately $4.258v + 58.26v^{-2/3}$, where v is the number of variables. In order to understand the behavior of MarketSAT on the hardest problems, as well as a broader class, we examined problems with numbers of clauses computed with the Crawford-Auton function (ratios of 4.87 and 4.65 for 15 and 20 variables, respectively), as well with clause/variable ratios of 2, 2.5, 3, 3.5, 4, 5, 5.5, and 6. We ran 500 trials for each number of variables and clauses, for a total of 9000.

Results

Given that agents and auctions run in parallel, a plausible measure of runtime 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 (Wurman, Walsh, & Wellman 1998)).

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 MarketSAT, zero or multiple such "flips" can take place. However, our experiments show that the number of MarketSAT flips is highly linear in the number of rounds (regression result: $\mathit{flips} = 0.5\mathit{rounds} - 11.9$, $R^2 = .99$) hence we choose to compare MarketSAT bidding rounds to GSAT flips as a rough estimate of their relative performance.

Both algorithms found solutions to all problems, and we have been unable to construct a problem that MarketSAT could not solve, given enough time and a high enough $v_c(\{g_c\})$ (2×10^7 was sufficient for all our experiments). GSAT nearly dominated MarketSAT, performing better in all but 3.3% of the instances. Moreover, GSAT ran significantly faster. In the median case, MarketSAT required 17.6 times as many rounds as GSAT required flips. Figure 4 shows the median runtime of each algorithm over the range of clause/variable ratios in 20-variable problems. Not only did MarketSAT fare poorly on median measures, but also had a heavy tail of slow runs. As shown in Figure 5, the mean runtimes of MarketSAT are significantly higher than the medians. Table 1 shows that, for 20-variable problems at the crossover point, MarketSAT has significantly higher standard deviation, mean/median ratio, and maximum/median ratio. We obtained qualitatively similar results for 15-variable problems.

We found ourselves constrained in the size of problems we could practically solve with MarketSAT. While GSAT can solve 500-variable problems on 1992 computers (Selman, Levesque, & Mitchell 1992) MarketSAT can be prohibitively slow on much smaller problems. In our 20-variable trials, the worst MarketSAT run required 1.48×10^6 rounds, compared with only $2,017$ flips for the worst GSAT run. Some 25-variable problems we have tried require tens of millions of MarketSAT rounds and hours of single-CPU computation. Despite the somewhat limited range of problem sizes explored, we believe they are sufficient to establish the qualitative performance of MarketSAT relative to GSAT[1].

Distributed Constraint Satisfaction

Yokoo et al. (Yokoo *et al.* 1998) provide a general formulation of distributed constraint satisfaction problems (CSPs). Since CSP is polynomially equivalent to 3SAT, we could solve 3SAT problems with distributed constraint satisfaction algorithms. However, the distributed CSP formulation imposes much weaker decentralization constraints than MarketSAT.

In Yokoo et al.'s model, each agent represents one or more variables. An agent proposes values for its variables and evaluates constraints on those variables. Agents send and receive (partial) solutions as well as lists of nogood partial solutions. This mapping of agents to the underlying problem is admittedly more natural than in MarketSAT. The algorithms presented by Yokoo et al. can be viewed as adaptations of centralized CSP solution techniques to a multiagent environment, performing systematic search with respect to the (implicit) global state. The primary adaptation is *asynchronous backtracking*, which allows agents to propose partial solutions in parallel. Since they do not ultimately restrict the dissemination of information, their distributed framework is able to provide performance comparable to (and, with effective parallelism, potentially better than) centralized algorithms.

In contrast, MarketSAT appears to perform substantially worse, but adheres to much stricter decentralization constraints. If we assume that, for a given 3SAT variable u, the literal producers π_u and $\pi_{\bar{u}}$ are in fact a single producer with separate policies for g_u and $g_{\bar{u}}$, then the task dependence network can be constructed with only the following knowledge of the original 3SAT problem: (1) a producer representing a variable knows in which clauses it is contained, and (2) π_c knows all the variables. No further knowledge of the structure nor partial solutions of the 3SAT problem is later transmitted, except indirectly and compactly via prices. In contrast, the distributed CSP algorithms require that agents have more knowledge of the initial problem and the partial solutions. Any agent involved in a constraint must be able to evaluate the constraint, and agents may potentially form links with all other agents to transmit partial solutions or nogoods.

Conclusions

Our experiments show that, in terms of runtime, GSAT dominates MarketSAT by several orders of magnitude. The ability of centralized algorithms like GSAT to evaluate global state (and similarly for distributed formulations that allow arbitrary information sharing) provide a decided advantage in combinatorial search. In contrast, MarketSAT makes decisions about individual variables entirely based on local price information.

Nevertheless, that a market *can* reliably solve satisfiability problems, however slowly, in a distributed fashion, provides existential evidence for highly decentralized solution methods to general classes of combinatorial problems. Moreover, it is intriguing that MarketSAT succeeds without explicit search state or randomization. At this time we do not fully understand how prices guide MarketSAT to a solution, hence we can only conjecture that MarketSAT is complete. We regard the present work as evidence of the versatility of the market model of computation, and a starting point for further studies of decentralizing combinatorial optimization.

[1] In preliminary experiments, CPLEX, a commercial mixed integer programming package, solves much larger SUPP-CHAIN encoded 3SAT problems much faster than MarketSAT. Hence we believe that MarketSAT's performance is mainly attributable to decentralization, rather than the encoding.

Appendix: Lemmas

Lemma 4 *No agent places a buy bid above* $v_c(\{g_c\}) + 4\delta$.

Proof. First we show that, if $\beta(g)$ is the maximum buy bid that any agent ever places for good g, then a producer with output g will never bid above $\beta(g) + 2\delta$ for any of its inputs. Assume that π will raise its bid for input g_i from α to α', where $\beta(g) < \alpha \leq \beta(g) + \delta < \alpha' \leq \beta(g) + 2\delta$. π must bid α' for g_i before bidding above $\beta(g) + 2\delta$. It must also be that π is losing its current bid α for g_i, otherwise it would not raise that input bid. But then the current price quote for g_i is greater than $\beta(g)$. Thus, π will bid greater than $\beta(g)$ for its output g. Because bids are nondecreasing, it will never again win its output bid, and hence will never raise its bid for input g_i above $\beta(g) + 2\delta$.

The consumer never bids above $v_c(\{g_c\})$ for g_c. Given the relationship between output good bids and input good bids shown above, it follows that π_c never bids above $v_c(\{g_c\}) + 2\delta$ for any of its inputs, and hence no variable assignment producer π_u or $\pi_{\bar{u}}$ ever bids above $v_c(\{g_c\}) + 4\delta$ for any of its inputs. □

Lemma 5 *A producer of an assignment to a variable updates its sell bid at most* $2|Q|(v_c(\{g_c\}) + 4\delta)/\delta$ *times.*

Proof. Consider producer π of the specified type (e.g. either π_u or $\pi_{\bar{u}}$ for some variable u). After its initial bid, π changes its output bid when either the price or bid price for an input changes, or it switches from winning to losing an input (in which case it switches from using the price to the ask price in its output bid calculation). Thus, the number of times π changes its output bid is bounded by the total number of times the price and ask price change for each input.

The price quote of an input good g_q of π changes only in response to buy bids from the variable assignment producers. By Lemma 4, producers never place a buy bid above $v_c(\{g_c\}) + 4\delta$. Because the buy bids start at zero and go up by δ, the price and ask price of g_q can each change at most $(v_c(\{g_c\}) + 4\delta)/\delta$ times. Since π has at most $|Q|$ inputs, we have proved the lemma. □

Lemma 6 *The producer* π_c *updates is sell bid at most* $|U|2[v_c(\{g_c\}) + 4\delta + 4|Q|(v_c(\{g_c\}) + 4\delta)]/\delta$ *times.*

Proof. As in the proof for Lemma 5, the number of times π_c changes its output bid is bounded by the total number of times the price and ask price change for each input. The price quote on an input good g_u of π_c is affected by sell bids from producers π_u and $\pi_{\bar{u}}$ as well as by the buy bids by π_c. By Lemma 5, and the fact that there are two potential sellers of g_u, no more than $4|Q|(v_c(\{g_c\})+4\delta)/\delta$ bid updates are placed for g_u. By Lemma 4, π_c bids no higher than $v_c(\{g_c\})+4\delta$ for g_u, and by its policy, does so in increments of δ. Thus the price and ask price of g_u *each* increase no more than $[v_c(\{g_c\})+4\delta+4|Q|(v_c(\{g_c\})+4\delta)]/\delta$ times. This holds for all $|U|$ input goods of π_c. $\square$

Acknowledgments We thank the anonymous reviewers for helpful comments. This work was supported by a NASA/Jet Propulsion Laboratory Graduate Student Researcher fellowship and DARPA grant F30602-97-1-0228 from the Information Survivability program.

References

Cheng, J. Q., and Wellman, M. P. 1998. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics* 12:1–24.

Crawford, J. M., and Auton, L. D. 1996. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence* 81:31–57.

Fujishima, Y.; Leyton-Brown, K.; and Shoham, Y. 1999. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Sixteenth International Joint Conference on Artificial Intelligence*, 548–553.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Tenth National Conference on Artificial Intelligence*, 459–465.

Samuelson, P. A. 1949. Market mechanisms and maximization. Research memorandum, RAND. Available in *The Collected Scientific Papers of Paul A. Samuelson*, Stiglitz, J. E., ed., 1966.

Sandholm, T. 1999. An algorithm for optimal winner determination in combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, 542–547.

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

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Tenth National Conference on Artificial Intelligence*, 440–446.

Shoham, Y., and Tennenholtz, M. to appear. Rational computability and communication complexity. *Games and Economic Behavior*.

Walsh, W. E., and Wellman, M. P. 1998. A market protocol for decentralized task allocation. In *Third International Conference on Multi-Agent Systems*, 325–332.

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

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

Wurman, P. R.; Wellman, M. P.; and Walsh, W. E. 1998. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, 301–308.

Ygge, F., and Akkermans, H. 1998. On resource-oriented multi-commodity market computations. In *Third International Conference on Multi-Agent Systems*, 365–371.

Yokoo, M.; Durfee, E. H.; Ishida, T.; and Kuwabara, K. 1998. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering* 10(5):673–685.