Generalized English Auctions by Relaxation in Dynamic Distributed CSPs with Private Constraints[1]

Marius-Călin Silaghi, Djamila Sam-Haroud, Monique Calisti, and Boi Faltings
Swiss Federal Institute of Technology Lausanne
1015 Ecublens, Switzerland
{silaghi,haroud,calisti,faltings}@lia.di.epfl.ch

ABSTRACT

Certain classes of negotiation problems lend themselves to strategies ensuring that no agent can gain by lying. Truth incentive protocols, among which Generalized Vickrey Auction (GVA) is one of the most famous, can then be used to centrally compute fair and efficient solutions. However, for problems that allow no truth incentive protocols (e.g. problems with false name bids), English Auctions are prefered to GVA. In this paper we show how the framework of Distributed Constraint Satisfaction (DisCSP) with private constraints can be extended for modeling and solving negotiation problems such as English Auctions with multiple-items where bids can correspond to complex actions (selling, buying, or both).

1. INTRODUCTION

Having agents represent the interests of their owners is desirable in many practical applications. One advantage of using software agents consists in their speed of response. Automated negotiation is a process whereby a distributed network of software agents agree on decisions on behalf of their owners. Agents negotiate on *resources* and their decisions are conditioned by *constraints* (e.g. costs, existence,...). When the available information is suboptimally used, local decisions can lead to losses for some parties involved in negotiations. Bad decisions can also result in a decrease of the social welfare by inefficient resource allocation. There is consequently a demand for automated negotiation techniques that are fair and acceptable to each of the involved parties.

In the automated multi-agent setting the work described in [21] has brought a new and revolutionary idea, based on concepts from Game Theory. It proves that certain problems from the class called Task Oriented Domains can be solved by *truth incentive* protocols[2]. A protocol is truth incentive if any participant cannot gain more than by telling the whole truth about its problem. Additional problems were shown to allow truth incentive protocols and the best known examples are the one item auctions. They can be solved with the Vickrey protocol [16]. An extension of this protocol, Generalized Vickrey Auctions (GVA) [15], has also been proposed for multiple-items auctions, namely auctions where individual pricing for items is different from grouped pricing. Truth incentive protocols naturally allow automatic centralized resolution and this is a big success of AI in general. Unfortunately, even if the GVA protocol [15] guarantees a certain degree of equity for many multiple-items auctions, it is not always truth incentive [19]. The outcome for this complication, illustrated in [20], is that with public constraints, the social welfare is sub-optimally managed. General auctions as well as other types of negotiations may be truth incentive even if resources and parties are involved in other known negotiations. However, if there exist *unknown connections* with future negotiations, revealing the truth presents a risk for involved parties. The unknown connections of a given problem P consist of all future negotiations for which not all details are known and that share resources with P. In particular, truth incentive-ness is penalized by the following property, related to the theorem 7.1 presented in [7]:

Property 1. *If a particular constraint on a resource x of an agent can ever be involved in an unknown future problem that allows no truth incentive protocol, then no truth incentive protocol can be safely used for any problem requiring x.*

This property does not mean that no truth incentive mechanism exists for the known part of the problem. Rather it states that involved parties might prefer not to reveal their constraints due to external unknown conditions. We therefore introduce the next definition.

Definition 1 (Globally truth incentive). *Let P be a problem allowing a truth incentive mechanism. P is* globally truth incentive *if it does not have any unknown connection. The corresponding truth incentive mechanism is then* globally truth incentive.

For example, a multi-provider bandwidth reservation problem may not be truth incentive with respect to the structure of the internal networks (e.g. due to the mechanisms currently used on the corresponding market). The negotiation for buying cables for the providers may be truth incentive with respect to the same structures of the internal networks.[3] Therefore, if the auctioneer of the cable negotiation cannot be a trusted party for the routing negotiation,

[1]An initial version of this article has been previously distributed under the title: *Negotiation by Relaxation in Dynamic Distributed CSPs with Private Constraints*

[2]Also referred as *incentive compatible mechanisms.*

[3]Actually it may not be truth incentive when the reservation price of the auctioneer for future auctions can be changed.

then the cable negotiation is not globally truth incentive. This property has implications in most problems. Making abstraction of it, even if optimal in the present, may be fatal for the near future. For simplicity, in the remaining part of this paper we refer problems that do not allow global truth incentive mechanisms due to unknown connections as being non-truth incentive problems.

For this kind of problems and for problems with false name bids, English Auctions are prefered to GVA since they do not require the agents to reveal everything.

DEFINITION 2 (GEA). *English Auctions for multi-items auction problems where bids can correspond to complex actions (selling, buying, or combinations) are referred as Generalized English Auctions (GEA).*

In this paper we present how the framework of Distributed Constraint Satisfaction (DisCSPs) can be extended to model GEA.

In the next section we introduce background definitions and define formally our goal. The section 4 gives a global view of the GEA and comments on the rational strategies available to the agents. A new framework is introduced in section 5 and a parallel is drawn with the well-known notions from English Auctions. The main technical contribution of the paper is concentrated in section 6 where a family of distributed algorithm is adapted for generalized english auctions.

2. PROBLEM STATEMENT

The English Auction negotiation mechanism is a good candidate for non-globally truth incentive problems since it offers a certain degree of privacy. For example, an agent may win the auction without revealing the highest price it can pay. Contrary to GVA, English Auctions are inherently distributed. The one item English Auctions mechanism is well understood and widely used in practice. Due to the complexity of the English Auctions for multiple-items auctions, GVA has been the most used solving mechanism even when it leads to less suitable solutions.

We show in this paper how English Auctions can be automated when multiple-items negotiation problems are modeled using an extension of the Distributed Constraint Satisfaction framework.

Distributed Constraint Satisfaction (DisCSPs) as defined in Artificial Intelligence provides a flexible framework for representing static distributed combinatorial problems.

DEFINITION 3 (DISCSPS). *A DisCSP is composed of:*

($d1$) *A set of agents $A = \{A_1, A_2, ...A_n\}$.*

($d2$) *A set of k variables $V = \{v_1, v_2, ...v_k\}$, each of them under the control of the agents interested in it. The variables in V are called external variables.*

($d3$) *A set of external variables $V_i = \{v_{i1}, v_{i2}, ..., v_{im}\}$, $V_i \subseteq V$, and a set of constraints $C_i = \{c_{i1}, c_{i2}, ...c_{ik_i}\}$ for each agent A_i, such that any external variable constrained by a constraint in C_i is also contained in V_i. The domain of a variable v_i is D_i. All the variables x_j constrained by constraints in C_i, and such that $x_j \notin V_i$ are said to be internal.*

Solving a DisCSP amounts to assigning values to both external and internal variables so that the constraints of all the agents are satisfied. The agents therefore need to coordinate their decisions on the external variables.

Powerful complete algorithms for solving DisCSPs have been proposed recently [2, 14, 18, 11]. In particular, the framework of Asynchronous Aggregation Search (AAS) [11] allows for a natural modeling and a highly parallelized solving of general problems with private constraints, which is adapted to non globally truth incentive negotiation problems. In AAS, each agent is interested in enforcing a set of (private) constraints. Each agent can *assign* values to the variables involved in its constraints. The information exchanged via different types of messages coordinate assignments on shared variables.

2.1 Negotiation using DisCSPs

We propose to implement automated negotiation using an extension of the Asynchronous Aggregation Search with Asynchronous Reordering AASR (also denoted $\mathrm{MAS}_{(+,-,+)}$) [13]. AASR has been chosen for its flexibility and generality. However, as presently defined, it cannot model some important aspects of negotiation:

- *Dynamism*: In negotiation, the existence of the constraints, as well as the participation of the agents are conditioned by time and environment.
- *Evaluation of alternatives*: A fundamental element of negotiation is the ability to associate (ask) *prices* to alternatives. In our case, an alternative corresponds to an assignment that can be agreed by an agent. Each agent must be able to ask a price for any of its possible agreements. In practice, an agent may also have to pay something for each alternative it chooses. This corresponds to the *cost*, which is often a hidden information. Moreover, two alternatives with the same price can be discriminated using *preferences*.
- *Relaxation*: Due to new information acquired during resolution, the agents will accept to relax their constraints. By constraint relaxation we mean that an agent can renounce to parts of its constraints (e.g. can reduce prices).

We show how prices, preferences and constraint relaxation can be integrated in DisCSPs in order to provide the necessary framework for multiple-items English Auctions.

2.2 Fairness

The quality of an automated negotiation protocol mainly depends on its ability to compute fair solutions. In the following we give a set of definitions for characterizing solutions in problems with hidden costs. These definitions mainly adapt the commonly used ones to our framework.

The cost of a solution is given by the sum of the costs of the agents. Note that since requested prices are negative costs, the sum of all the costs paid by all the agents in A is equal to the sum paid by the agents in A to factors outside A. Any negotiation is started by a subset of A called *initiators*. We assume that the initiators are self-interested. The sum of the costs paid by the initiators to some agent is called price.

DEFINITION 4 (SOLUTION COST). *The cost of a solution is given by the sum of the prices asked by the agents to* *the initiators for agreeing on the alternatives composing the solution.*

Definition 5 (Utility). *The utility of an agent is defined as the difference between the price it asks and the cost it pays for the chosen alternative.*

A rational agent prefers to offer alternatives that increase its utility[4]. Therefore, even if the utilities are hidden, it is usually beneficial for the agents to reveal the order of their preferences.

Definition 6 (Pareto-optimal solution). *A solution is pareto-optimal if any other solution is either equally preferred for all agents, or worse for at least one agent, given the order defined by the utilities of each agent on solutions.*

We call declared-pareto-optimal solution, a pareto optimal solution computed for the DisCSP declared by the agents.

Definition 7 (Declared-Pareto-optimal). *A solution is declared-pareto-optimal if any other solution is either equally preferred for all agents, or worse for at least one agent, given the order defined by prices and declared preferences.*

Definition 8 (Estimated Social Welfare). *An estimated social welfare solution (ESW) is a declared-pareto-optimal solution with minimal Solution Cost.*

Guaranteeing that a solution is ESW is possible with complete search techniques. We also want the ESW solution to be chosen impartially (fairness).

Definition 9 (Fairness). *When several ESWs are candidate, fairness consists in giving them equal probability to be chosen.*

Real Social Welfare

Definition 10 (Equivalent Solutions). *A problem with equivalent solutions is a problem where the difference between the quality (value) of its solutions is equal to the difference between the cost or the respective solutions (the solutions are equally good).*

It is worth mentioning that for problems with equivalent solutions an ESW gives the best possible estimation of the real Social Welfare (SW). This is the case of a bandwidth allocation problem where any two paths in the network are equally good as long as it has the required bandwidth and quality of service. One of our goals is to help in reaching a social welfare solution.

Definition 11 (Social Welfare Solution). *For problems with equivalent solutions, a social welfare solution (SW) for a set of agents A is defined as a solution minimizing the sum of all the costs paid by all the agents in A for agreeing on the alternatives composing the solution.*

[4]Alternatively, the notion of *worth* [21] can be similarly used.

3. RELATED WORK

Researchers have already related *negotiation* and *Distributed CSPs* from both sides. On one side, the *negotiation* is seen as technique for solving distributed CSPs. The authors of [6] propose a min-conflict heuristic technique called *negotiation search* as a means of converging towards a solution in a distributed problem with heterogeneous components. On the other side agents have also been proposed for solving by negotiation over-constrained resource allocation problems in [3, 5]. Frameworks for over-constrained distributed problems with public constraints are presented in [17, 4]. Our approach shares common concepts with the framework proposed in [9] for resource allocation. An overview of known types of auctions was given in [8].

4. THE NEGOTIATION PROTOCOL

A negotiation is viewed as a multi-criteria optimization problem where the agents have to find a solution maximizing their utilities while respecting their constraint on resources. Auctions are special case of negotiations where the negotiation ends when a subset of the agents (auctioneers) cannot improve any longer their utility. In GEA such problems are solved by iterative improvement of ESW solutions according to the following protocol.

- $a1$ Compute the best solutions (ESW) satisfying the constraints so far imposed by the agents and retain one of them.
- $a2$ If any solution was found at $a1$, publish the ESW as an any-time solution.
- $a3$ If any agent wants to relax the constraints it imposes, go to $a1$.
- $a4$ If any solution was found at $a1$, return the estimated ESW and stop.
- $a5$ Return failure and stop.

The solution of a GEA is a global optima (i.e. no better solution can be constructed by the agents).

If the prices are modified with a minimal increment and the set of alternatives is finite, the previous protocol is safe to converge in finite time as long as the agents are stable in the order on their preferences and commit to their agreements (monotonicity in finite domains).

A solution S of a distributed problem may not need the agreement of some particular agent A_i. In that case we say that A_i is *inactive* for S. Conversely, we say that A_i is *active* for S if its agreement is necessary for choosing S. This provides a means to model a facet of *competition* useful for ameliorating the ESW. An agent, *inactive* for the current solution, may indeed want to make concessions to become *active*.

Rational Strategies

If any agent A has means of estimating the *will to risk* of all the agents, the following relaxation strategy can be proposed.

- A wants to propose its best alternative first. If it is not satisfied with the current solution, S, and has the lowest will to risk by accepting S, A will do a minimal concession.

This strategy is not in equilibrium when A knows well the strategies and data of the others. In that case, A's rational strategy can be:

- If A knows that some other agent will make a minimal concession, getting involved with A in the best ESW solution, then A makes no concession.

Commitment of initiators

If the initiators are requested to commit to their currently imposed constraints, meaning that the first chosen ESW solution is the final one, the rational negotiation strategies change.

- If A knows that no solution can be found where A is *inactive*, then if A has the lowest will to risk, it will make a minimal concession, otherwise A will wait.
- If a solution, S, can be found without A in *active* state, A proposes all its acceptable alternatives.

If A knows well the strategies and data of the others, A's rational strategy would rather be:

- If A knows that no solution can be found where A is *inactive*, then if A has the lowest will to risk, A will make a minimal concession, otherwise A will wait. If this is the last round, and others will make some next relaxation leading to a solution, A waits for them.
- If a solution can be found with A in *inactive* state, since A can compute a quality of the solution S that can be obtained without A, then A proposes the alternative being the minimal concession leading to a global solution better than S (if any exists).

5. EXTENDING DisCSPs

In order to model practical negotiation problems, we introduce a formalism that enriches the DisCSP framework with dynamism, preferences and constraint relaxation. The extended framework builds on the notion of Valued CSPs [10]. First we describe the problem of an agent, A_u, as a Negotiation Valued CSP, (NVCSP$_u$). NVCSP$_u$ consists of:

- A minimal increment, ε.
- A set of external variables, V(u). The domain of each variable contains a value F[5] meaning *unchanged* and *indifferent*.
- An ordered set of global constraints c_1(u),...,c_{n_u}(u).
- Each pair (valuation v, constraint c_i(u)) has associated a tuple:

$$T_i^v(u) = (feasible_i^v(u), price_i^v(u), preference_i^v(u)).$$

$T_i^v(u)$ is such that for each constraint $c_i(u)$, price$_i^v$(u)$\geq$cost^v(u) and if n_u≥i>j>0 then:

- for any valuation v, $feasible_j^v(u) \rightarrow feasible_i^v(u)$ and $price_i^v(u) \leq price_j^v(u)$,
- there exists a valuation v such that either $feasible_i^v(u) \neq feasible_j^v(u)$, or

$$feasible_i^v(u) = feasible_j^v(u) = T$$
$$and$$
$$price_i^v(u) + \varepsilon \leq price_j^v(u)$$

[5] or a set of values.

A Dynamic DisCSP[6] (DyDisCSP) is defined by:

- A set of agents A_0,...,A_n. $A_k, k \in [0, h), n \geq h > 0$, are h agents called *initiators*.
- Each agent A_j owns a NVCSP, NVCSP$_j$.
- Each agent A_j is interested in a set of external variables V(j).
- $cost^v(j)$ is private to the agent A_j for any valuation v.

Given a valuation v for all the external variables, $S(v)$ is the set of agents owning a variable not instantiated in v to F. By convention, the *initiators* always belong to $S(v)$.

Definition 12 (Acceptable valuation). *A valuation v is acceptable if each agent in $S(v)$ proposes for v a feasible associated tuple, ($feasible_{k_i}^v(i) = T$).*

By $\check{v}$ we denote a valuation obtained from the valuation v by reassigning a subset of variables to F such that $v \neq \check{v}$.

Definition 13 (Stable valuation). *v is stable if there exists no acceptable $\check{v}$.*

Intuitively, a *stable valuation* is minimal in the sense that it corresponds to a agreement of the agents in $S(v)$, and by eliminating any subset of agents from $S(v)$, no agreement can be obtained with the *initiators*.

When v is stable we say that all the agents in $S(v)$ are defined by v as **active**.

Definition 14 (Solution). *A solution of a DyDisCSP is a stable valuation v of all the external variables such that if each agent A_i in $S(v)$ proposes for v an associated tuple $(T, price_{k_i}^v(i), preference_{k_i}^v(i))$, $k_i \leq n_i$, and if*

$$A = \{b |\ b = \underset{a}{\text{argmin}}(\sum_{A_i \in S(v), i \geq h} price_{k_i}^a(i))\},$$

then $v \in A$, v is pareto-optimal for $S(v)$ over A. and no agent A_i, $i>0$, wants to reveal a constraint c_j, $j>k_i$.

The feasibility condition is $\sum_{A_i \in S(v)} price_{k_i}^a(i) \leq 0$.

The feasibility condition verifies that the solution is acceptable to the *initiators*. If v is a solution of a DyDisCSP, then $S(v)$ is the *solver set* for v.

In our framework, the step $a1$ of GEA amounts to solving a DyDisCSP where the k_i of any A_i is fixed.

[6] We propose to call DyDisCSP a DisCSP where the participation of agents is dynamic. An alternative is to call this framevork dynamic distributed valued CSP (DyDisVCSP). A DyDisCSP where the agents own dynamic CSPs can then be called dynamic distributed dynamic CSP (DyDisDyCSP).

5.1 Relation with existing negotiation frameworks

In [8] was presented a framework for Multi-Unit Combinatorial Exchanges. The characteristic of these auctions is that in one bid a bidder can be selling some items and buying other items simultaneously. The multi-unit combinatorial exchange winner determination problem (MUCEWDP) is to label the bids as winning or losing so as to maximize surplus under the constraint that demand does not exceed supply. In our framework, MUCEWDP are modeled by having exactly one *initiator* that owns no constraint.

The auctions enabled by our approach to GEA (DyDisCSP) are an extension of MUCEWDP where the final solution has to get the agreement of a predefined (sub)set of agents (the *initiators*). We therefore call such auction problems *Multi-Unit Supervised Combinatorial Exchanges (MUSCEWDP).*

DEFINITION 15 (MUSCEWDP). *MUSCEWDP are Multi-Unit Combinatorial Exchanges winner determination problems where the solution needs the agreement of a predefined set of agents.*

The other existing types of auctions are suggested in [8] to be instances of MUCEWDP. Therefore, they can also be modeled as MUSCEWDP.

5.2 Modeling English Auctions

The dynamism enabled by DyDisCSPs can be used to model English Auctions. We draw now a parallel between the introduced framework and typical English Auctions. The equivalence of notions is:

- external variable ⇔ transaction for the allocation of a good to an agent, other than the current owner.

  Each variable is in the NVCSPs of the current owner and of the target owner of the good.

  values $\in \{T, F\}$, showing if the transaction is chosen.

- initiator ⇔ auctioneer.

- price ⇔ minus of the bids for a combination of allocations

  The cost in DyDisCSPs is the minus of the worth in English Auctions.

- price − cost ⇔ utility or worth.

The initiator launches the search in the space of allocations. At each step, an agent A_u imposes the constraint $c_{k_u}(u), k_u \leq n_u$. A relaxation of the imposed constraints corresponds to increasing k_u.

6. EXTENDING AASR

In this section we introduce an algorithm called Secure Asynchronous Search (SAS) which is an adaptation of AASR to the DyDisCSP framework. SAS can be used to find all ESW at step a1 of GEA. First we recall the basic elements of the AAS [11] protocols. The *system agent* is a special agent that receives the subscriptions of the agents for the search. It decides an initial order of the agents and announces the termination of the search. If the agent A_i is ordered before A_j, then we say that A_i is an *ancestor* of A_j. We denote with A^i the agent that has the position $i, i \geq 0$.

DEFINITION 16 (ASSIGNMENT). *An assignment is a triplet (v, set, h) where v is a variable, set a set of values for v and h a history of the pair (v, set).*

A history h for an assignment $a = (v, set, h)$ proposed by an agent A^k takes the form of a list of pairs $|i : l|$ where i is the index of an ancestor of h that has made a proposal on v and l is the value of a counter. These pairs are ordered in h according to the ascending value of i. The last pair in h has the form $|k : l_{k^j}|$. An order $\propto$ is defined on pairs such that $|i_1 : l_1| \propto |i_2 : l_2|$ means either $i_1 < i_2$, or $i_1 = i_2$ and $l_1 > l_2$. An assignment requests higher priority agents to comply with a proposal, therefore it defines by itself of a nogood. All the values that do not comply with the assignment are nogoods. Such nogoods are called **nogoods entailed by the view**.

DEFINITION 17 (NEWER HISTORY). *A history h_1 is newer than a history h_2 if a string-like comparison on them, using the order $\propto$ on pairs, decides that h_1 precedes h_2.*

An assignment with history h^j_x built by A^j for a variable x is **valid** for an agent A^m, $m \geq j$ if no other history known by A^m and built by agents A^k, $k \leq j$ for some assignment of x, is more recent than h^j_x. A nogood is valid if all the assignments contained in its premise are valid.

DEFINITION 18 (EXPLICIT NOGOOD). *An explicit nogood has the form $\neg V$, where V is a list of assignments.*

DEFINITION 19 (ORDERING). *An ordering is a sequence of agent names.*

The agents communicate, using channels without message loss, via:

- **ok** messages, sent from agent A_j to agent A_i, and having as parameter a list of assignments for variables in which A_i is interested.

- **nogood** messages which have as parameter an explicit nogood.

- **add-link**(vars) messages, sent from agent A_j to agent A_i, informing A_i that A_j is interested in the variables *vars*. They are always answered.

- **reorder** messages which have as parameter an ordering.

- **heuristic** messages which have as parameter data for computing heuristics.

In AAS, each agent is allowed to keep its constraints secret and has to declare from start the external variables for which it has constraints.

AASR [13], allows the agents to asynchronously propose new orders among themselves. An order is represented by a sequence of agents and is tagged by a history. An agent can only propose new orders within a bounded delay after having received a new proposal. We consider here the case where an agent does not reorder agents having lower positions than itself.

When used in competitional situations (e.g. within negotiation), the AASR technique is no longer appropriate. The reason is that a solution of such problems does not need to be an acceptable solution for all the agents, as long as some of them are not **active** in the solution and do not gain anything[7].

6.1 Secure Asynchronous Search

In AASR, both **ok** and **nogood** messages transport some kind of nogoods. These are the nogoods entailed by the view, respectively the explicit nogoods. In order to allow the agents detect messages that are potentially harmful for the quality of the computed solution, we introduce the notions of legal nogood and legal assignment. We want to prevent the agents from disturbing the search by generating illegal messages. A message (containing a nogood $\neg N$) is illegal if it is generated by an agent that can be inactive in a valuation extending the partial valuation in the Cartesian-product defined by N. SAS requests agents to build messages in such a way that their lawfulness can be proved.

Definition 20 (Legal explicit nogood). *Any legal explicit nogood generated by an agent A_i, where A_i is not an* initiator, *must contain at least one* assignment *of a variable v_j from $V(i)$ such that v_j does not contain F.*

Definition 21 (Justification). *Each assignment I_i generated by an agent A_i that is not* initiator *needs a* justification. *The* justification *of the assignment I_i consists of a pair (v,h) built from an assignment (v,s,h) that activates A_i.*

The *justification* of an *assignment*, a, corresponds to a relaxation of the nogood entailed by the view given by a and is stored in the history of the *assignment*, attached to the pair corresponding to the agent that has generated a. A history has now the form $|i_1,l_1,j_1|i_2,l_2,j_2|...$ where i_k is the index of an agent, l_k is the value of an instantiation counter and j_k is the justification of the corresponding instantiation.

Property 2. *The space needed by an agent to store all the assignments is O(nv), where n is the number of agents and v is the number of variables.*

Proof. The number of possible simultaneous valid assignments is nv since each agent can generate at most v valid assignments at a time (one per variable). All assignments and justifications can be represented as a directed graph having the valid assignments as nodes. The maximum number of arcs in this graph is $2nv$ since there cannot be more than 2 arcs getting out of a node. □

Corolary 1. *The size of an assignment is O(nv).*

Property 3. *SAS has polynomial space complexity in each agent.*

Proof. AASR requires polynomial space and the only additional structures required by SAS consist of the new assignments in justifications. For all the references to assignments in the structures of AAS, Corolary 1 shows that a polynomial mapping exists to the new form of the assignments. □

Besides generating illegal nogoods, the agents can also generate illegal assignments against their competitors.

[7]And have to reduce their preferences.

Definition 22 (Legal assignment). *An assignment is legal if its justification is valid and the variable in the justification does not contain F in its instantiation. By convention, any assignment generated by an initiator is legal.*

No assignment (v,s,h) generated in SAS may aggregate in s both the value F and some other values.

6.2 The SAS protocol

In order to enable agents to make proposals, they must be given the opportunity to know when they are activated. The active/inactive state of an agent is know when either one of its external variables is instantiated outside F (active), or when all its external variables are instantiated with F (inactive). For the security of the search, we want to involve on low positions in search only agents that are known to be active.

Rule 1 (Initiator first). *The agent A^1 has to be an* initiator.

Rule 2 (Active first). *Whenever possible, each agent proposes orders to make sure that the agent on the next position is known to be active.*

In order to let agents know which of the next agents are active, active agents must announce all their instantiations for external variables to all their successors.

Since in SAS the messages must prove that their sender is active, agents must generate only legal nogoods. Any other nogood would be discarded. The next rule shows how legal nogoods can be obtained.

Rule 3 (Nogood generation). *Whenever an agent A_i computes an explicit nogood N that is not legal, and the set in the newest assignment it has received for some variable v_j from $V(i)$ does not contain F, it should add the newest assignment of v_j to N.*[8] *If this is not possible, it means that A_i is inactive and it should refrain from sending N to other agents. This rule does not apply to initiators.*

Coalitions can still be created in SAS. In fact any agent that does not check if a receiving message is valid makes a (temporal) coalition with the sender.

Rule 4 (Checking). *The receiver of an explicit nogood N should check that N is legal. Also the receiver of any* assignment, *(when an* **ok** *message is received), should check that the new assignment is legal and the assignment is not empty (the search cannot be voluntarily blocked).*

If one of these conditions is not respected, the messages must be discarded.

In order to ensure completeness and termination of SAS, the management of justifications has to be coherent. The *justifications* trigger **add-link** messages in the same conditions as the assignments received in an explicit nogood in AAS. Moreover, justified nogoods should not be delivered to the receiving agent and integrated in the other structures inherited from AASR before the answer to eventual **add-link** messages is received.

[8]When illegal nogoods are made legal, they are in fact relaxed. Agents that must relax nogoods can use heuristics for choosing the variable v_j from $V(i)$. (e.g. choosing the variable for which the known assignment was generated by an agent with the lowest priority.)

Rule 5 (Justification change). *Whenever the* justification *of an agent A_i is modified, A_i has to send again all its assignments.*

Only one assignment is used as justification by an agent at a time.

Rule 6 (Stored agent-view). *Each agent stores all the valid assignments it receives.*

Rule 7 (Justification invalidation). *Whenever the* justification *J of a stored* assignment *a_1 in A_i is invalidated by some incoming new* assignment *a_2, A_i has to invalidate a_1 and has to apply again this* invalidation rule *as if a new* assignment *of the variable in a_1 would have been received.*

Lemma 1. *The information maintained by agents in Secure Asynchronous Search is consistent with their predecessors.*

Proof. The rules for nogoods generation and nogood checking are allowed by the AAS framework and the proof in [11] remains valid. Sending any (finite) number of **add-link** messages when justifications are received fits AAS as well. It remains to prove that the properties are maintained when the invalidation, of a *justification*, j, triggers the invalidation of an *assignment*, x, that it justifies.

The role of a stored *assignment* in AAS is to define a valid *nogood entailed by the view* of the agent. Therefore x defines a *nogood entailed by the view.* In the Secure Asynchronous Search, the agent activated by j has to invalidate and recompute its instantiation (due to the justification change rule). Therefore x, becomes indeed invalid. Consequently, in all cases the information maintained by the agents will eventually be consistent with their predecessors. □

Definition 23 (Quiescence). *By quiescence of a group of agents we mean that none of them will receive or generate any legal valid nogoods, new legal valid assignments, reorder messages or add-link messages.*

Lemma 2. *At quiescence for the agents $A^1, ..., A^i$, an agent A^i, knows whether there exist any active agent placed after it or not.*

Proof. If any initiator is placed after A^i, this is known to A^i since the last ordering was received.
Otherwise, A^i knows all assignments outside F done by itself or predecessors for external variables. A^i also knows the variables that can activate its successors since they are announced as initial data. A^i therefore knows when any successor is activated by the agents $A^1, ..., A^i$.
If no successor is activated by $A^1, ..., A^i$ and no successor is initiator then, since an inactive agent cannot activate another agent, no successor of A^i is active.

Lemma 3. *In Secure Asynchronous Search, let a set of k agents reach quiescence in finite time t^k and at quiescence they take positions $1, ..., k$ where each of them agrees with its presecessors. Then, if some of the other agents are active, either failure is detected or an active agent will reach quiescence on position $k+1$ in finite time after t^k having an instantiation that agrees with its predecessors.*

Proof. The number of possible `add-link` messages is bounded so we do not need to consider them.

Let τ be the maximum delay for delivering a message. After $t^k+\tau$, all the agents know the instantiations of $A^i, ..., A^k$. Within time $t^k_h = t^k + \tau + t_h + \tau$, no **heuristic** message is received any longer by the agents $R^0, ..., R^k$. The final order that decides the agent taking the position A^{k+1} is therefore issued before $t^k_h + t_r$ and if any successor is active, an active agent becomes A^{k+1}. After $t^k_h + t_r + \tau$, the identity of A^{k+1} no longer changes and its view remains stable after $t_o, t_o < t^k_h + t_r + \tau$. Since the domains are finite, in finite time after t_o either some proposal of A^{k+1} is no longer refused, or all its proposals are refused and A^{k+1} infers at time $t_n, t_n > t_o$ a legal valid nogood $\neg N$.
Since $\neg N$ is valid and legal, either:

I. N it is empty and the search fails, or

II. when $t_o < t_k$, its receiver changes its instantiation for a variable, v, and either it or its predecessors announce A^{k+1} of the new assignment for v. But such an announcement arrives at A^{k+1} after $t_n + 2\tau > t_o$. This contradicts the hypothesis that the view is stable after t_o and that $t_o < t_n$.

III. The predecessor receiving $\neg N$ detects failure and the search ends.

Therefore in all possible cases the lema holds.

Lemma 4. *In Secure Asynchronous Search, the active agents reach quiescence on consecutive positions 1 to k, and all inactive agents are placed after A^k.*

Proof. Due to the lemma 1, each agent knows when some successor is active and according to the rule 2 (Active first), the active agents are ordered consecutively.

The solutions computed by Secure Asynchronous Search correspond to an agreement among the tuples revealed by the active agents. This agreement consists in the instantiation of the shared external variables to the same value. The optimality of the prices has to be checked separately by the broker.

Proposition 1. *The Secure Asynchronous Search is complete, correct, and terminates. The solutions it computes correspond to an agreement on the tuples revealed by the active agents.*

Proof. Quiescence of the active agents: is a result of lemma 3. In bounded time, the active agents find a solution and reach quiescence letting the broker to detect it, or a failure is detected.

Completeness: All the nogoods are generated by inference and therefore no empty nogood is generated if a solution exists. An erroneous inference made by an agent A_i cannot eliminate a solution that does not instantiate outside F a variable activating the agent A_i. Therefore those solutions needed the acceptance of A_i.

Correctness: The assignments that activate the active agents at quiescence and the other variables set by these agents to values that do not contain F is a Cartesian-product C. Each element in C defines a **stable valuation** v when it is expanded by instantiating any other external variables to F. The active agents correspond to $S(v)$. They all know the instantiations of their predecessors and they can generate legal valid nogoods since they are active. Since at quiescence the active agents agree with their predecessors (Lemma 3), then all the agents in $S(v)$ agree on v.

For the Secure Asynchronous Search, the termination detection algorithms from AAS can no longer be used since the agents are not trusted. The messages for announcing the solution may be sent via inactive agents and therefore they can be corrupted. For the general case, the *solution* can be detected by the *system agent*. Each agent has to send its solution to the *system agent*. The *system agent* has to find the *solver-sets*, namely all the subsets of agents agreeing with the initiators that are consistent and together define a *solution*. This is a problem with polynomial complexity. For some real problems, alternative techniques for *solution* detection can be used.

PROPERTY 4. *ESW solutions can be computed using the Secure Asynchronous Search.*

Proof. The completeness of the Secure Asynchronous Search ensures that all ESW solutions are computed together with other solutions. They are filtered out during computation according to the definition of ESW, yielding the step a1.

7. PRACTICAL CONSIDERATIONS

In order to enhance the scalability of Secure Asynchronous Search, special care must be devoted to managing the size of the problem. Since the total number of agents in the world, ready to be involved in a negotiation can be huge, we want to maintain the set of involved agents to a minimum, without losing completeness. The current techniques can be adapted to this new requirement. In order to reduce computation effort, the agents have the interest to use acceptable value ordering heuristics for guiding the search.

DEFINITION 24 (MIS). *An agent belongs to the* minimal interesting set *if, during search, at least one partial solution, better than the lowest bound has been found, which requires the participation of this agent.*

DEFINITION 25 (INVOLVED AGENT). *An agent is involved in a current computation if it has been sent at least one proposal for that computation.*

An agent is involved only if it belongs to the *minimum interesting set*. Usually, agents can be added in a natural manner to a computation. In several existing protocols, it is less easy to eliminate agents. We have introduced the notions of *minimum interesting set* and *involved agent* to guarantee that an agent is involved only when this is needed for completeness. For example, in Figure 1 the active agents at node n_4 are A and F. The involved agents are A, B, C, D, F, the MIS also contains H and L. G has not been in MIS.

Let us consider that the agents with right to control each external variable can be found in some *yellow pages*. The *system agent* maintains a set of involved agents, empty in the beginning. Whenever a set of *initiator* agents pose a new problem, they are inserted in the set of involved agents.

Each time that the involved agents reach quiescence, but instantiated variables in the partial solution correspond to the activation of some agents not yet involved in the search, the *system agent* checks that the partial solution built so far has a lower value than the best solution found so far. If this condition is not respected, a nogood is generated for the current valuation. Otherwise, the *system agent* sends them a

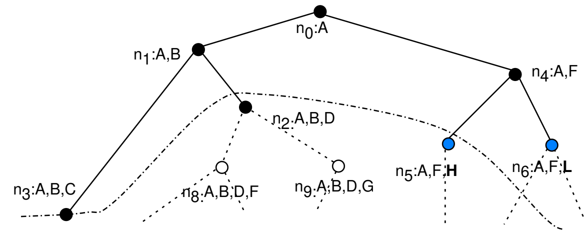

Figure 1: The solid circles correspond to visited nodes, up to node n_4. The list of agents active at each of them is shown under the node. The curved dashed line show the lower bound for ESW.

request to get involved (an **involve** message). The **involve** takes as parameter the current priority of the receiving agent in the search, the set of *initiators*, and the priorities (and information on external variables) for all the other agents involved in the search.

When new agents accept to get involved in the computation, all the existing agents should receive information on the priority and the external variables of the new agents via an **involved** message. They will have to send their proposals to the new agents.

If some agents cannot be contacted or refuse to get involved in the search, the *system agent* announces it to the already involved agents by an **update-yellow-pages** message mentioning the name of the removed agents and the variables they control. The alternatives on variables controlled by removed agents have to be disabled from all agents.

The nogoods can also be reused between rounds by using the technique proposed in [12]. That technique consists of explaining inferences with references to constraints (CR). The CRs do not necessarily stand for a given constraint, but provides a way to signal when due to relaxations, a nogood is invalidated. The algorithms remain polynomial space only if the number of CRs in use is bounded. The reuse of CRs can be enabled by attaching to them counters.

Branch and Bound can also be used dynamically during the search. However, in order to use it, the AAS protocol has to be run on the dual representation of the problem where all the constraints of each agent are represented by a variable. The values (prices and preferences) for all the tuples in the sent aggregate have to be attached to the values in the aggregate. However, the initial variables in the aggregate no longer need separate histories, rather only one history is needed for the instantiation of each agent. If aggregations are done in such a way that all values in an assignment have equal values, then this value needs to be sent only once. The trimming of Branch and Bound is more informed if each agent sends its proposed aggregate to all the agents with higher priority. The trimming takes into account only active agents.

8. EXAMPLE OF APPLICATION

The problem of Multi-Provider Interactions as described in [1] consists of reserving requested bandwidth across a heterogeneous network of self-interested providers that have secret constraints. The problem of the existence of solutions has been modeled as a Distributed CSP [1].

Since providers are anxious about revealing details concerning their network links, costs and capacity, this problem is a natural candidate for solving with GEA. The problem is not globally truth incentive. The initiators consist in the clients requesting some bandwidth reservation across the network. The minimal involved set expands itself as long as the sum of the costs does not reach the best found solution. For the Multi-Provider Interactions problem, the preferences can be used for dealing with previsions of future traffic or with internal maintenance scheduling.

9. CONCLUSIONS

We present an approach to negotiation for problems where no globally truth incentive mechanism is available. The concept of Dynamic Distributed Constraint Satisfaction is proposed and we show how it allows for modeling complex characteristics of such problems. In particular we show that it is sufficient to add a simple set of rules to asynchronous constraint satisfaction protocols (e.g. AASR) in order to meet new requirements on security, dynamism and evaluation of alternatives (SAS).

We illustrate how Dynamic DisCSPs can be used in general negotiation problems (e.g. GEA) to provide fair environments. The problems that can be modeled by Dynamic DisCSPs, MUSCEWDPs, are identified as being a generalization of Multi-Unit Combinatorial Exchanges (MUCEWDP). The presented framework inherits from Constraint Reasoning generality and flexibility in modeling.

As shown in [13] for MAS, usually asynchronous search algorithms are generalizations of synchronous versions and behave like the last ones for:

- certain delays in message delivery,
- certain strategies of the agents concerning:
  - aggregation choices and
  - delays in answering to messages
- and when the channels can transit by intermediary agents.

Therefore, the SAS algorithm presented here has such a synchronous equivalent, Secure Synchronous Search (SSS).

Acknowledgments

This research is supported by the Swiss National Science Foundation. The choice of the name for the new framework was inspired by Christian Bessiere that has mentioned it in a discussion on conflicts between acronyms for DisCSPs and Dynamic CSPs at the DCS'2000 workshop in Singapore.

10. REFERENCES

[1] M. Calisti, C. Frei, and B. Faltings. A distributed approach for QoS-based multi-domain routing. *AiDIN'99, AAAI-Workshop on Artificial Intelligence for Distributed Information Networking*, 1999.

[2] Z. Collin, R. Dechter, and S. Katz. Self-stabilizing distributed constraint satisfaction. *Chicago Journal of Theoretical Computer Science*, 2000. UCI Technical Report, TR 92-67.

[3] S. E. Conry, K. Kuwabara, and V. R. Lesser. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on systems, man, and cybernetics*, 21(6):1462–1477, Nov/Dec 91.

[4] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the Conference on Constraint Processing (CP-97),LNCS 1330*, pages 222–236, 97.

[5] T. Khedro and M. R. Genesereth. Modeling multiagent cooperation as distributed constraint satisfaction problem solving. In *Proceedings of ECAI'94*, pages 249–253, 94.

[6] S. Lander and V. R. Lesser. Understanding the role of negotiation in distributed search among hetergeneous agents. In *Proceedings of IJCAI-93*, pages 438–444, Chambéry, France, 93.

[7] T. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of ICMAS-96*, pages 299–306, 1996.

[8] T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of AAAI'00*, pages 90–97, 2000.

[9] A. Sathi and M. Fox. *Distributed Artificial Intelligence*, volume 2, chapter Constraint-Directed Negotiations of Resource Reallocations, pages 163–193. Morgan Kaufmann, California, 1989.

[10] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Procs. IJCAI'95*, pages 631–637, 95.

[11] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proc. of AAAI2000*, pages 917–922, 2000.

[12] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Maintaining hierarchical distributed consistency. In *Proceedings of the International Workshop on Distributed Constraint Satisfaction*. CP'00, 2000.

[13] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous consistency maintenance with reordering. Technical Report #01/360, EPFL, March 2001.

[14] G. Solotorevsky, E. Gudes, and A. Meisels. Distributed Constraint Satisfaction Problems - a model and application. Preprint: http://www.cs.bgu.ac.il/˜am/papers.html, Oct 97.

[15] H. R. Varian. Economic mechanism design for computerized agents. In *Proceedings of the First Usenix Workshop on Electronic Commerce*, 95.

[16] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 61.

[17] M. Yokoo. Constraint relaxation in distributed constraint satisfaction problem. In *ICDCS'93*, pages 56–63, June 93.

[18] M. Yokoo. Asynchronous weak-commitment search for solving large-scale distributed constraint satisfaction problems. In *Proc. of 1st ICMAS*, pages 467–318, 95.

[19] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name declarations in mechanism design: Towards collective decision making on the internet. In *Proceedings of IDCDS-2000*, 2000.

[20] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust combinatorial auction protocol against false-name bids. In *Proc. of AAAI2000*, pages 110–115, 2000.

[21] G. Zlotkin and J. S. Rosenschein. A domain theory for

task oriented negotiation. In *TR 92-13, Hebrew University, Jerusalem*, 92.