Incentive Auctions and Stable Marriages Problems Solved with $\lfloor n/2 \rfloor$-Privacy of Human Preferences

Marius C. Silaghi
Florida Institute of Technology

July 8, 2004

Technical Report CS-2004-11

Abstract

Incentive auctions let several participants to cooperate for clearing a set of offers and requests, ensuring that each participant cannot do better than by inputing his true utility. This increases the social welfare by efficient allocations, and is proven to have similar outcomes as the traditional English Auctions. The desk-mates (stable matchings) problem comes from the need of placing students in pairs of two for working in projects or seating in two-seats desks. The stable marriages problem consists of finding matches of a man and a woman out of two sets of men, respectively women. Each of the persons in the previous two problems has a (hopefully stable) secret preference between every two possible partners. The participants want to find an allocation satisfying their secret preferences and without leaking any of these secret preferences, except for what a participant can infer from the identity of the partner/spouse that was recommended to her/him.

We use a distributed weighted constraint satisfaction (DisWCSP) framework where the actual constraints are secrets that are not known by any agent. They are defined by a set of functions on some secret inputs from all agents. The solution is also kept secret and each agent learns just the result of applying an agreed function on the solution. The new framework is shown to improve the efficiency in modeling the aforementioned problems. We show how to extend our previous techniques to solve securely problems modeled with the new formalism, and exemplify with the two problems in the title.

1 Introduction

Incentive auctions let several participants to cooperate for clearing a set of offers and requests, ensuring that each participant cannot do better than by inputing his true utility. This increases the social welfare by efficient allocations, and is proven to have similar outcomes as the traditional English Auctions.

The desk-mates (or stable matchings) problem consists in grouping a set of students in stable working teams of two, such that whenever one person wants to change her partner for a third one, the third one prefers her current partner to the change. The students have a secret preference between any pair of potential partners, and between working with each partner or working alone.

The stable marriages problem consists of matching pairs out of two distinct sets of participants [GS62]. One member of the pair should belong to the first set and the second member should belong to the second set. Whenever one participant wants to change her partner for a third one, the third participant prefers her current partner to the change. The participants have a secret preference between any pair of potential partners.

Versions of these problems, without privacy requirements, have been long known and studied. Techniques for the stable marriages problem are used in US to assign hospitals to medical interns [Ski90]. It is an example of constraint satisfaction problem (CSP). A CSP is modeled as a set of variables and a set of constraints on the possible values of those variables. The CSP problem consists in finding assignments for those variables with values from their domains such that all constraints are satisfied. The CSP techniques require every eventual participant to reveal its preferences (e.g. to a trusted server), to compute the solution. Therefore, they apply only when the participants accept to reveal their preferences to the trusted party.

There exist frameworks and techniques to model and solve distributed CSPs (DisCSPs) with privacy requirements, namely when the domains of the variables are private to agents [YDIK98, MJ00], or when the constraints are private to agents [SSHF00a, Sil03b, SR04].

However, incentive auctions cannot be fully modeled with existing DisCSP frameworks. Also, the desk-mates and the stable marriages problems seem not to be modeled efficiently (i.e. with a reduced search space) with any of the two known types of distributed CSP frameworks. In this article we propose a new framework for the distributed constraint satisfaction problems. It can model naturally existing distributed constraint satisfaction problems, and also the desk-mates, stable marriages problems, and all necessary steps for incentive auctions. The new framework assumes that the constraints are not known to absolutely any agent but they are computable from secret inputs, with known functions. These functions use

x1 \ x2	T	W
P	0	1
Q	1	0

Figure 1: A constraint between two variables, place (x_1) $x_1 \in \{Paris(P), Quebec(Q)\}$, and time ($x_2$) $x_2 \in \{Tuesday(T), Wednesday(W)\}$. The 0s mark rejected tuples. I.e. this constraint allows only the pairs (P,W) and (Q,T), and can be written $\{(P,W),(Q,T)\}$.

secret inputs provided securely by the different participants. Similarly, the final assignments are secret and each agent can retrieve just the result of applying some agreed function on the secret solution.

We also show how secure multi-party computation techniques that we have recently developed for solving DisCSPs with private constraints can be extended to solve problems described in the new framework. We start introducing formally the CSP problem.

CSP. A *constraint satisfaction problem* (CSP) is defined by three sets: (X, D, C). $X = \{x_1, ..., x_m\}$ is a set of variables and $D = \{D_1, ..., D_m\}$ is a set of finite domains such that x_i can take values only from $D_i = \{v_1^i, ..., v_{d_i}^i\}$. $C = \{\phi_1, ..., \phi_c\}$ is a set of constraints. A constraint ϕ_i limits the legality of each combination of assignments to the variables of an ordered subset X_i of the variables in X, $X_i \subseteq X$. An assignment is a pair $\langle x_i, v_k^i \rangle$ meaning that the variable x_i is assigned the value v_k^i.

A tuple is an ordered set. The projection of a tuple ϵ of assignments over a tuple of variables X_i is denoted $\epsilon_{|X_i}$. A solution of a CSP (X,D,C) is a tuple of assignments, $\epsilon *$, with one assignment for each variable in X such that each $\phi_i \in C$ is satisfied by $\epsilon *_{|X_i}$. The search space of a CSP is the Cartesian product of the domains of its variables.

Example 1 *In a CSP, one has to find a place (x_1) and time (x_2) for meeting. x_1 is either Paris (P) or Quebec (Q), i.e. $D_1 = \{P, Q\}$. x_2 is either Tuesday (T) or Wednesday (W), i.e. $D_2 = \{T, W\}$. There are two constraints: $\phi_1 = \{(P,W),(Q,T)\}$, and $\phi_2 = \{(P,W),(Q,T),(Q,W)\}$. ϕ_1 is depicted in Figure 1. The problem is to find values for x_1 and x_2 satisfying both ϕ_1 and ϕ_2.*

We consider that a set of participants are the source of such CSPs and one has to find agreements for a solution, from the set of possible alternatives, that satisfies a set of (secret) requirements of the participants. This view suggests a concept of a distributed CSP. Several frameworks were proposed so far for Distributed Constraint Satisfaction [ZM91, CDK91, YSH02a, MJ00]. Some versions consider that each agent owns a constraint of the CSP [ZM91, SGM96]. This constraint could model the private information of the agent [SSHF00a]. Other versions consider that each agent owns the domain of a variable while the constraints are shared [YDIK98]. The secret domains can also model some private constraints of the agent.

None of the two approaches, namely private variables or private domains, can model efficiently the desk-mates or stable marriages problems. This is because their private data does not *directly* constrain the allocation of the natural shared resources. An indirect relation exist with such a constraint. Redundant variables would need to be introduced in the system. A new framework will be introduced in this article to avoid the redundant variables.

2 Background

Our techniques here apply only to problems whose constraints and outputs can be represented as first order logic expressions, or as arithmetic circuits on inputs. Actually, we propose a procedure to translate first order logic definitions of constraints/outputs into arithmetic circuits. In the following we introduce arithmetic circuits and a short overview of the literature and techniques that made them relevant.

2.1 Secure Arithmetic Circuit Evaluation

Secure multi-party computations can simulate any arithmetic circuit [BOGW88] or boolean circuit [Kil88, Gol04] evaluation. An *arithmetic circuit* can be intuitively imagined as a directed graph without cycles where each node is described either by an addition/subtraction or by a multiplication operator (see Figure 2). Each leaf is a constant. In a secure arithmetic circuit evaluation, a set of participants perform the operations of an arithmetic circuit over some inputs, each input being either public or an (encrypted/shared) secret of one of them. The result of the arithmetic circuit are the values of some predefined nodes. The protocol can be designed to reveal the result to only a subset of the agents, while none of them learns anything about intermediary values. One says that the multi-party computation *simulates* the evaluation of the arithmetic circuit. A *boolean circuit* is similar, just that the leafs are boolean truth values, false or true, often represented as 0 and 1. The rest

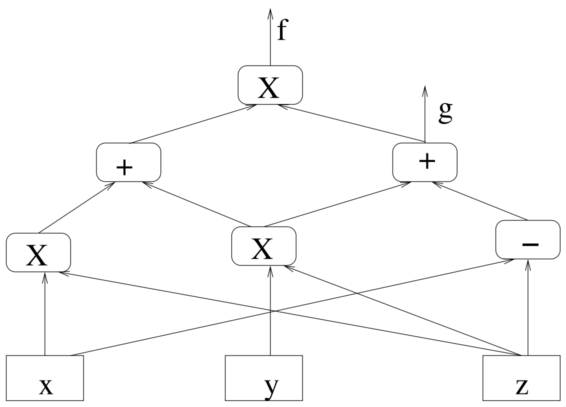

Figure 2: An arithmetic circuit, $g = yz + (x - z)$ and f=$(xz + yz)g$. Each input can be the secret of some participant. The output may not be revealed to all participants. All intermediary values remain secret to everybody.

of the nodes are boolean operators like AND or XOR. A function does not have to be represented in this form to be solvable using general secure arithmetic circuit evaluation. It only needs to have such an equivalent representation. For example, the operation $\sum_{i=B}^{E} f(i)$ is an arithmetic circuit if B and E are public constants and $f(i)$ is an arithmetic circuit. The same is true about $\prod_{i=B}^{E} f(i)$. Such constructs are useful when designing arithmetic circuits.

Secure simulation of arithmetic circuit evaluation. The secure multi-party simulation of arithmetic circuit evaluation proposed in [BOGW88] exploits Shamir's secret sharing [Sha79]. This sharing is based on the fact that a polynomial $f(x)$ of degree $t{-}1$ with unknown parameters can be reconstructed given the evaluation of f in at least t distinct values of x, using Lagrange interpolation. Absolutely no information is given about the value of $f(0)$ by revealing the valuation of f in any at most $t{-}1$ non-zero values of x. Therefore, in order to share a secret number s to n participants $A_1, ..., A_n$, one first selects $t{-}1$ random numbers $a_1, ..., a_{t-1}$ that will define the polynomial $f(x) = s + \sum_{i=1}^{t-1}(a_i x^i)$. A distinct non-zero number τ_i is assigned to each participant A_i. The value of the pair $(\tau_i, f(\tau_i))$ is sent over a secure channel (e.g. encrypted) to each participant A_i. This is called a (t, n)-threshold scheme. Once secret numbers are shared with a (t, n)-threshold scheme, evaluation of an arbitrary arithmetic circuit can be performed over the shared secrets, in such a way that all results remain shared secrets with the same security properties (the number of supported colluders, $t{-}1$) [BOGW88, Yao82].

For [Sha79]'s technique, one knows to perform additions and multiplications when $t \leq (n-1)/2$. Since any $\lfloor n/2 \rfloor$ participants cannot find anything secret by colluding, such a technique is called $\lfloor n/2 \rfloor$-private [BOGW88].

2.2 Overview of MPC-DisCSP1

In [Sil03b] we have proposed a multi-party computation technique, called MPC-DisCSP1, that extracts a random solution of a distributed CSP. MPC-DisCSP1 uses general multi-party computation building blocks. General multi-party computation techniques can solve securely certain functions, one of the most general classes of solved problems being the arithmetic circuits (see Section 2.1). A distributed CSP is not a function. A DisCSP can have several solutions for an input problem, or can even have no solution. Two of the three reformulations of DisCSPs as a function (see [SR04]) are relevant for MPC-DisCSP1:

i A function DisCSP^1() returning the first solution in lexicographic order, respectively an invalid valuation τ when there is no solution.

ii A probabilistic function DisCSP() which picks randomly a solution if it exists, respectively returns τ when there is no solution.

For privacy purposes only the 2^{nd} alternative is satisfactory. DisCSP() only reveals what we usually expect to get from a DisCSP, namely *some* solution. DisCSP^1() intrinsically reveals more [SR04]. MPC-DisCSP1 implements DisCSP() in five phases:

1. Share the secret parameters of the input DisCSP using Shamir's secret sharing.

2. The shared DisCSP problem is shuffled in a cooperative way, reordering values (and eventually variables), by composing secret permutations from each participant agent, such that no agent can retrieve the total domains permutation by comparing the initial with the obtained DisCSP.

3. A version of DisCSP^1() where the operations performed by agents are independent of the input secrets (to avoid leaking the secrets), is executed by simulating arithmetic circuits evaluation with the technique in [BOGW88].

4. The solution returned by DisCSP^1() at Step 3 is translated into the initial problem formulation using a transformation that is inverse of the shuffling at Step 2.

5. Construct the solution from its secret shares.

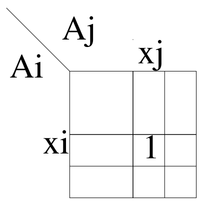

Figure 3: A constraint between x_i and x_j for the desk-mates and stable-marriages problems.

It is also possible and very simple to find all solutions [HCN$^+$01]. However, when only a single solution is needed, this leaks a lot of information. At Step 3, MPC-DisCSP1 requires a version of the DisCSP^1() function whose cost is independent of the input, since otherwise the users can learn things like: *The returned solution is the only one, being found after unsuccessfully checking all other tuples, all other tuples being infeasible.* Since the used DisCSP^1() has to be independent of the problem details, its cost is exponential (at least as long as nobody proves P=NP).

3 Distributed CSPs with constraints secret to everybody

In this article we redefine the distributed CSP framework, aiming to model efficiently (i.e. with a reduced search space) the distribution of some famous CSP problems, namely the incentive auctions, desk-mates and the stable marriages problems.

Desk-mates The desk-mates problem consists in placing a set of persons $A = \{A_1, ..., A_m\}$ in a set of two-seats desks, such that if any person A_i prefers a person A_j to the desk-mate selected for her, then A_j prefers her current desk-mate to A_i.

A way of modeling the desk-mates problem as a CSP is to have one variable x_i for each person A_i specifying the index of the desk-mate assigned to her by the solution, or specifying i, the index of A_i itself, if she remains alone. The constraints are obtained by preprocessing the input from participants about their preferences. The fact that a person A_i prefers A_u to A_v is specified by the first order logic predicate $P_{A_i}(u, v)$. There is a constraint ϕ^{ij} between every pair of distinct variables x_i and x_j. In first order logic notation, the constraint between each two variables x_i and x_j is:

$$\forall x_i, x_j : \phi^{ij}(x_i, x_j) \overset{\text{def}}{=} (P_{A_i}(x_j, x_i) \Rightarrow P_{A_{x_j}}(j, i)) \wedge (P_{A_j}(x_i, x_j) \Rightarrow P_{A_{x_i}}(i, j)) \wedge ((x_i = j) \Leftrightarrow (x_j = i)) \qquad (1)$$

Note that this model subsumes the constraints: $\forall i, j : x_i \neq x_j$.

Stable Marriages The stable marriages problem is the problem of finding a set of matches between a set of females, $A_1, ..., A_m$, and a set of males, $B_1, ..., B_m$, such that if any person from the set of females, A_i, prefers some male, B_j, to the partner selected for her, then B_j prefers his current partner to A_i. If any male, B_i, prefers some female, A_j, to the partner selected for him, then A_j prefers her current partner to B_i.

The stable marriages problem is an instance of the desk-mates problem, that can be modeled with a lower search space. A way of modeling the stable marriages problem as a CSP is to have one variable x_i for each female[1], specifying the index of the male assigned to her by the solution. The constraints are obtained by preprocessing the input of participants about their preferences. The fact that a person A_i prefers B_u to B_v is specified by the first order logic predicate $P_{A_i}(u, v)$. The fact that B_i prefers A_u to A_v is specified by the first order logic predicate $P_{B_i}(u, v)$. There is a constraint ϕ^{ij} between every pair of variables x_i and x_j. In first order logic notation, the constraint between each two variables x_i and x_j is:

$$\forall x_i, x_j : \phi^{ij}(x_i, x_j) \overset{\text{def}}{=} (P_{A_i}(x_j, x_i) \Rightarrow P_{B_{x_j}}(j, i)) \wedge (P_{A_j}(x_i, x_j) \Rightarrow P_{B_{x_i}}(i, j)) \wedge (x_i \neq x_j) \qquad (2)$$

In this formulation, the preferences of an agent do not necessarily require a total order on the possible spouses. Note that a total order is part of the common definition of the stable marriages problem [GS62, Ski90].

It is possible to extend the stable marriages problem to the case with an unequal number of males and females. In this case, it can be modeled either:

- as a usual instance of the desk-mates problem, with one variable for the partner of each participant, each participant publicly preferring to work alone rather then with somebody of the same type, or

- with variables only for females (or males), where the variables have an additional value, 0, for specifying that the participant remains single.

[1]Or male. Then, everything is defined symmetrically.

For the second case, there is a global constraint:

$$\begin{aligned}
\forall\epsilon, \phi(\epsilon) \quad &\overset{\text{def}}{=} \quad (\forall i, k : ((k \neq \epsilon_{|\{x_i\}}) \wedge P_{A_i}(k, \epsilon_{|\{x_i\}})) \Rightarrow && (3)\\
&\qquad\qquad ((k \neq 0) \wedge (\exists j : (\epsilon_{|\{x_j\}} = k) \wedge P_{B_k}(j, i)))) \wedge \\
&\qquad (\forall q, t : \epsilon_{|\{x_q\}} \neq \epsilon_{|\{x_t\}}) && (4)
\end{aligned}$$

The main complication with this kind of CSPs is that the constraints are functions of secrets that cannot be easily elicited from the participants. Distributed CSP frameworks are meant to address such problems.

Modeling the desk-mates and stable marriages problem with DisCSPs with secret constraints that are known to some agents. One can model the desk-mates problem with secret constraints known to some agents [ZM91, SSHF00b] by choosing as variables, $x_1, ..., x_m$, the index of the partner associated to each agent (that has to be computed) and using one additional boolean variable for each secret preference, $P_{A_i}(u, v)$. The total number of boolean variables is m^3, m^2 of them being actually fixed by public constraints (e.g. $P_{A_i}(u, u) = 0$). However, also taking into account the variables $x_1, ..., x_m$, the total search space becomes $O(m^m 2^{m^3})$. This is $O(2^{m^3})$ times worse than the centralized CSP formalization whose search space is only O(m^m).

We propose now a distributed constraint satisfaction framework that allows to model these problems with the same search space size as the CSP framework, O(m^m).

3.1 Redefining the Distributed Constraint Satisfaction Framework

In the previous part of this section we have exemplified CSP models for the desk-mates and stable marriage problem. We have seen that it is difficult to model efficiently these problems using existing private variable-, or private constraint-oriented distributed constraint satisfaction frameworks.

Let us propose a framework for modeling distributed CSPs, where a constraint is not (necessarily) a secret known to an agent, or public, but can also be a secret unknown to all agents.

Definition 1 *A Distributed CSP (DisCSP) is defined by six sets (A, X, D, C, I, O). A=$\{A_1, ..., A_n\}$ is a set of agents. X, D, and the solution are defined like for CSPs.*

I=$\{I_1, ..., I_n\}$ is a set of secret inputs. I_i is a tuple of α_i secret inputs (defined on a set F) from the agent A_i. Each input I_i belongs to F^{α_i}.

Like for CSPs, C is a set of constraints. There may exist a public constraint in C, ϕ_0, defined by a predicate $\phi_0(\epsilon)$ on tuples of assignments ϵ, known to everybody. However, each constraint $\phi_i, i>0$, in C is defined as a set of known predicates $\phi_i(\epsilon, I)$ over the secret inputs I, and the tuples ϵ of assignments to all the variables in a set of variables X_i, $X_i \subseteq X$.

$O=\{o_1, ..., o_n\}$ is the set of outputs to the different agents. $o_i : D_1\times...\times D_m \rightarrow F^{\omega_i}$ is a function receiving as parameter a solution and returning ω_i secret outputs (from F) that will be revealed only to the agent A_i.

Theorem 1 *The framework in the Definition 1 can model any distributed constraint satisfaction problems with private constraints [SSHF00b].*

Proof. The new DisCSP framework can be used to model any of the DisCSP problems with constraints private to agents, by defining I_i as the extensional representation of the private constraint of A_i (assuming the simple but sufficient case of one constraint per agent). $\phi_i(\epsilon, I)$ is then given by the corresponding value for ϵ in I_i. The outputs are going to be $o_i(\epsilon) = \epsilon$ for all i. q.e.d. □

Theorem 2 *The framework in the Definition 1 can model distributed constraint satisfaction problems with private domains [YDIK98].*

Proof. A private domain of an agent can also be modeled as a private unary constraint, in a DisCSP where each domain is the maximum possible domain for the variable. Then, the Theorem 1 applies. q.e.d. □

We do not claim that the new framework is more general than the existing frameworks. It enables us to model naturally and efficiently the desk-mate and stable marriages problems. One can also model these problems with the old frameworks, but they seem to yield much larger search spaces, and therefore less efficient solutions. Let us now exemplify how this framework can model the new problems.

Modeling the desk-mates problem as a DisCSP. A way of modeling the desk-mates problem as a DisCSP is to have one agent, A_i, and one variable, x_i, for each participant in the problem description. x_i specifies the index of the desk-mate assigned to A_i by the solution, or specifies i if she remains alone. The inputs I_i of each agent are given by the set of preferences $P_{A_i}(u, v)$, specifying whether A_i prefers A_u to A_v, for each u and v. The set F, to which belong the inputs and the outputs, is $\{true, false\}$.

There is a constraint ϕ^{ij} between every pair of variables x_i and x_j, defined as in Equation 1. The output functions are defined as: $o_i(\epsilon) \overset{\text{def}}{=} \epsilon_{|\{x_i\}}$. Namely, each agent learns only the name of her desk-mate. There is a public constraint:

$$\phi_0 \stackrel{\text{def}}{=} \forall i, j, ((x_i = j) \Leftrightarrow (x_j = i)) \wedge (x_i \neq x_j) \tag{5}$$

Modeling the stable marriages problem as a DisCSP. To model the stable marriages problem as a DisCSP, one can also have one agent, A_i, for each female participant A_i in the problem description. Each participant B_j is mapped to an agent A_{m+j}. One has m variables, $x_1, ..., x_m$, modeling the partners of the agents $A_1, ..., A_m$. x_i specifies the index of the spouse assigned to A_i by the solution, or specifies 0, if she remains alone. The inputs I_i of each agent are given by the set of preferences $P_{A_i}(u, v)$ and $P_{B_i}(u, v)$, specifying whether A_i prefers B_u to B_v, respectively whether B_i prefers A_u to A_v, for each u and v. The set F for inputs and outputs is $\{true, false\}$.

The constraint ϕ^{ij} between every pair of variables x_i and x_j, is defined as in Equation 2. The output functions for $i \in [1..m]$ are defined as: $o_i(\epsilon) \stackrel{\text{def}}{=} \epsilon_{|\{x_i\}}$. Namely, each female learns only the index of the husband proposed to her. To return to each male A_{m+i} the identity of the spouse proposed the him, the corresponding output is $o_{m+i} \stackrel{\text{def}}{=} \{k | x_k = i\}$.

There is a public constraint:

$$\phi_0 \stackrel{\text{def}}{=} \forall i, j, x_i \neq x_j \tag{6}$$

3.2 Distributed Weighted Constraint Satisfaction Problems

Definition 2 *A distributed constraint satisfaction problem (DisWCSP) is defined by six sets (A, X, D, C, I, O) and a set of acceptable solution qualities B, that can be often represented as an interval $[B_1, B_2]$. A=$\{A_1, ..., A_n\}$ is a set of agents. $X = \{x_1, ..., x_m\}$ is a set of variables and $D = \{D_1, ..., D_m\}$ is a set of finite domains such that x_i can take values only from $D_i = \{v_1^i, ..., v_{d_i}^i\}$. An assignment is a pair $\langle x_i, v_k^i \rangle$ meaning that the variable x_i is assigned the value v_k^i. A tuple is an ordered set. I=$\{I_1, ..., I_n\}$ is a set of secret inputs. I_i is a tuple of α_i secret inputs (defined on a set F) from the agent A_i. Each input I_i belongs to F^{α_i}. $C = \{\phi_0, ..., \phi_c\}$ is a set of constraints. A constraint ϕ_i weights the legality of each combination of assignments to the variables of an ordered subset X_i of the variables in X, $X_i \subseteq X$. ϕ_0 is a public constraint defined by a function $\phi_0(\epsilon)$ on tuples of assignments ϵ, known to everybody. Each constraint ϕ_i, i>0, in C is defined as a known function $\phi_i(\epsilon, I)$ over the secret inputs I, and the tuples ϵ of assignments to all the variables in a set of variables X_i, $X_i \subseteq X$. The projection of a tuple ϵ of assignments over a tuple of variables X_i is denoted $\epsilon_{|X_i}$. A solution is*

$\epsilon* = \underset{\epsilon \in D_1 \times ... \times D_n}{\text{argmin}} \sum_{i=1}^{c} \phi_i(\epsilon_{|X_i})$, *if* $\sum_{i=1}^{c} \phi_i(\epsilon*_{|X_i}) \in [B_1...B_2]$. O=$\{o_1, ..., o_n\}$ *is the set of outputs to the different agents.* $o_i : I \times D_1 \times ... \times D_m \rightarrow F^{\omega_i}$ *is a function receiving as parameter the inputs and a solution, and returning* ω_i *secret outputs (from* F*) that will be revealed only to the agent* A_i*.*

Solvers developed in our previous work require that the functions in sets O and C are input either in first order logic form, or in the form of arithmetic circuits.

The public constraint ϕ_0 can be input into the system using a set of constraints $\{\phi_0^1, \phi_0^2, ...\}$, and the tuples of assignments accepted by ϕ_0 can be obtained separately by each agent, when needed, using any systematic search technique that finds all solutions of a CSP, e.g. backtracking or lookahead algorithms (BT, BM, CBJ, FC, MAC, EMAC, etc.).

4 Adapting existing secure solvers to the new DisCSP framework

There exist a large set of algorithms addressing distributed CSPs with privacy of constraints [Sil02, HCN$^+$01, FMW01, WS04, YSH02b, Sil03b]. The ones that we succeed to extend to the new framework are:

- Finding the set of all solutions of a distributed constraint problem with secret constraints [HCN$^+$01].
- Finding the first solution in a lexicographic order for a distributed constraint satisfaction problem with secret constraints that are known to some agents [Sil03a].
- Finding a random solution for a DisCSP with secret constraints that are known to some agents [Sil03b].

When a solution is returned to the desk-mates problem, each agent A_i can infer that: *any agent* A_k *preferred by* A_i *to her current desk-mate* A_j*, prefers her current partner to* A_i. If only one solution is returned (picked randomly among the existing solutions), then no other secret preference can be inferred with certainty.

Theorem 3 *The desk-mates problem and the stable marriages problem can have several solutions.*

Proof. Consider a case with three agents, A_1, A_2, A_3 where $P_{A_1}(2,3)$, $P_{A_2}(3,1)$, $P_{A_3}(1,2)$. This is a loop of preferences, and has three stable solutions, the sets of teams

$\{(A_1, A_2), (A_3)\}$, $\{(A_2, A_3), (A_1)\}$, $\{(A_3, A_1), (A_2)\}$. Such an example can be constructed out of any similar loop of preferences, of any size.

A similar construct can show that the stable marriages problem can have several solutions. Namely take four agents with $P_{A_1}(1, 2)$, $P_{A_2}(2, 1)$, $P_{B_1}(2, 1)$, $P_{B_2}(1, 2)$. q.e.d. $\square$

If there exist several solutions, the agents will prefer not to reveal more then one of them. The remaining solutions would only reveal more secret preferences:

- Typically there is no other fair way, except randomness, to break the tie between several solutions.
- If the single solution that is returned is selected as the first one in some given lexicographic order on the variables and domains of the problem, then additional information is leaked concerning the fact that tuples placed lexicographically before the suggested solution do not satisfy the constraints [Sil03b].

As it follows, if it is known that a certain problem has only one solution, then any technique is acceptable among either:

- Finding and returning all solutions using the technique in [HCN$^+$01], or
- Returning only the first solution (e.g. by sequentially checking each tuple in lexicographical order until a solution is found).

Otherwise, strong privacy requirements make techniques returning a random solution [Sil03b] desirable, despite their potential of having a lower efficiency.

4.1 General Scheme

We will note that the main difference between the new DisCSP framework, and the one with secret constraints that are known to some agents, is that now the constraints need to be computed dynamically from secrets inputs.

The techniques solving DisCSPs with private constraints can be used as a black box, except for the equivalent of the Step 1 and Step 5 in MPC-DisCSP1 (see Section 2). Namely, instead of simply sending encrypted Shamir shares of one's constraint, those shares of the constraints have to be computed from the secret inputs of the agents. We therefore propose to replace the equivalents of Step 1 ad Step 5 with simulations of arithmetic circuit evaluation which will compute each $\phi_k(\epsilon, I)$ for each tuple ϵ and for the actual inputs I. This step is called *preprocessing*.

Similarly, instead of just reconstructing the assignments to variables in a solution ϵ at Step 5, one will have to design and execute secure computations of the functions $o_k(\epsilon)$. This step is called *post-processing*. We show that in our cases this can also be done using simulations of arithmetic circuit evaluations.

Assume $\mathcal{A}$ is some algorithm using Shamir's secret sharing for securely finding a solution of a distributed CSP (with secret constraints known to some agents). The generic extension of the algorithm $\mathcal{A}$ to solve the DisCSP in the new framework is:

- **Preprocessing:** Share the secrets in I with Shamir's secret sharing scheme. Compute each $\phi_k(\epsilon_{|X_k}, I)$ for each tuple $\epsilon_{|X_k}$ and for the actual inputs I by designing it as an arithmetic circuit and simulating securely its evaluation. The public constraint ϕ_0 can be shared by any agent.

- Run the algorithm $\mathcal{A}$ as a black-box, for finding a solution $\epsilon*$ shared with Shamir's secret sharing scheme, for a DisCSP with parameters (i.e. constraints) shared with Shamir's secret sharing scheme.

- **Post-processing:** Compute each $o_i(\epsilon*)$ by designing it as an arithmetic circuit and simulating securely its evaluation. Reveal the result of $o_i(\epsilon*)$ only to A_i.

4.2 Pre- and post- processing for desk-mate and stable marriages problems

In the remaining part of the article we will prove that it is possible to design the needed preprocessing and post-processing to solve our two examples of DisCSPs: desk-mates and stable marriages, using the general scheme defined above.

Preprocessing for the desk-mates problem. We assume the same choice of variables, as for the CSP formalization of this problem in Section 3. Let us now show how simple arithmetic circuits can implement the required preprocessing.

Each variable x_i specifies the index of the desk-mate associated to A_i. The input of each agent A_i is a preference value $P_{A_i}(j,k)$ for each ordered pair of agents (A_j, A_k), and specifying whether A_i prefers A_j to A_k. $P_{A_i}(j,k)$=1 if and only if A_i prefers A_j to A_k. Otherwise $P_{A_i}(j,k)$=0. A constraint ϕ^{ij} is defined between each two variables, x_i and x_j. I.e. $\phi^{ij}[u,v]$ is the acceptance value of the pair of matches: $(A_i, A_u), (A_j, A_v)$. One synthesizes $m(m-1)/2$ such constraints:

$$\phi^{i,j}[u,v] \quad = \quad \begin{cases} 0 & \text{when } u = v \\ \\ (1 - P_{A_i}(v,u) * (1 - P_{A_v}(j,i)))* & \\ \qquad\qquad (1 - P_{A_j}(u,v) * (1 - P_{A_u}(i,j))) & \text{when } u \neq v \end{cases}$$

The public constraint ϕ_0 (same as in Equation 5) restricts each pair of assignments:

$$\forall\epsilon, \epsilon{=}(\langle x_i, u\rangle, \langle x_j, v\rangle) : \phi_0(\epsilon) \overset{\text{def}}{=} ((u{=}j) \Leftrightarrow (v{=}i)) \wedge (u \neq v)$$

ϕ_0 is known by everybody, and therefore there is no need to compute it with arithmetic circuits. The complexity of this preprocessing is $O(m^4)$ multiplications of secrets (for m^2 binary constraints with m^2 tuples each).

The desk-mates problem does not require any arithmetic circuit evaluation for the post-processing, as each agent A_i learns a value existing in the solution, $o_i(\epsilon) = \epsilon_{|\{x_i\}}$. The participants just reveal to A_i their shares of x_i in the solution.

Preprocessing for the stable marriages problem. Some simple arithmetic circuits can implement the preprocessing for the stable marriages problem, too.

Each variable x_i specifies the index of the male associated to the female A_i. The input of each female A_i specifies a preference value $P_{A_i}(j,k)$ for each pair of males, (B_j, B_k). Each male B_i specifies a preference value $P_{B_i}(j,k)$ for each pair of females (A_j, A_k). $P_{A_i}(j,k){=}1$ if and only if A_i prefers B_j to B_k. Otherwise $P_{A_i}(j,k){=}0$. $P_{B_i}(j,k){=}1$ if and only if B_i prefers A_j to A_k. Otherwise $P_{B_i}(j,k){=}0$. A constraint ϕ^{ij} is defined between each two variables x_i and x_j. $\phi^{ij}[u,v]$ is the acceptance value of the pair of matches: $(A_i, B_u), (A_j, B_v)$. One synthesizes $m(m-1)/2$ constraints:

$$\phi^{i,j}[u,v] \quad = \quad \begin{cases} 0 & \text{when } u = v \\ & \\ (1 - P_{A_i}(v,u) * (1 - P_{B_v}(j,i)))* & \\ \qquad\qquad (1 - P_{A_j}(u,v) * (1 - P_{B_u}(i,j))) & \text{when } u \neq v \end{cases}$$

The public constraint ϕ_0 (same as in Equation 6) restricts each pair of assignments:

$$\forall\epsilon, \epsilon{=}(\langle x_i, u\rangle, \langle x_j, v\rangle) : \phi_0(\epsilon) \overset{\text{def}}{=} (u \neq v)$$

specifying that it is not possible for two persons to be associated to the same spouse in a solution.

Post-processing for the stable marriages problem. The stable marriages problem requires a post-processing phase to compute and reveal to each male the spouse proposed to him. Remember that the variables specify only the spouse for each female. The function $o_{m+i} \overset{\text{def}}{=} \{k | x_k = i\}$ can be computed with the following arithmetic circuit.

$$o_{m+i} = \frac{1}{(i-1)!(m-i)!} \sum_{j=1}^{m} j \prod_{k=1}^{i-1} (x_j - k) \prod_{k=i+1}^{m} (k - x_j)$$

5 Incentive Auctions

To clear a combinatorial auction according to the 1-st price when several allocations may be optimal. :

- The participants select as public parameters of the problem a set of variables X where there is a distinct variable for each item to be sold, and the domain of each variable is the set of participants that may own the item at the end of the auction (by winning it or by not selling it). There is a function $\phi_k(\epsilon, I)$ for each participant A_k, which associates to each possible tuple ϵ of assignments of the variables in X, an element of I_k (the bid of A_k for ϵ). The maximum and minimum value of the sum of the bids B_1, B_2, are also enforced by allocating ranges of possible bids to each participant.

- Each participant decides its secret inputs I_k (bids) for each tuple defining an allocation, by taking into account both the items she acquires and the reservation price of the items she cedes in that allocation.

- The secret inputs are shared with Shamir's scheme. Each agent encrypts her share with her own public key and the agents of the participants form a mix-net shuffling these shares, and randomizing them at each permutation. The shares can also be sent directly from their creator to the mix-net, encrypted with the public key of the destination participant.

- A solution of the DisWCSP is computed with a secure protocol (e.g. MPC-DisWCSPx).

- The chosen allocation and its total price is revealed by revealing the shares.

- To only reveal the winner allocation to the participants involved in it, the participants must define the functions o_k such that each participant learns the items that she receives. Also, each participant receives the shares of the variables for items that she is selling, to learn whom to give them. Namely, o_k returns an array such that with m items and n participants, $\forall i, 0 < i \leq m$, if x_i models an item of A_k then $o_k[i] = x_i$, otherwise $o_k[i] = (x_i = k)$. These first order logic predicates are translated in arithmetic circuits as shown later: $(x_i = k)$ becomes $\frac{1}{(k-1)!(n-k)!} \prod_{i=1}^{k-1}(x_i - i) \prod_{i=k+1}^{n}(i - x_i)$.

- The exact price p_u to be paid by each agent A_u in this case is the bid of the agent A_u for the solution, and can be made known to a participant A_i with an output $o'_i[u] = \phi_u(\epsilon*, I)$, by the arithmetic circuit: $\sum_k \lambda(\sum_{i=1}^{n} (w_i \prod_{j=1}^{i-1} d_i), \prod_{i=1}^{n} d_i)[k]\phi_u(\epsilon_k, I)$,
where w_i is the shared secret specifying the index of x_i in the solution and $\lambda(s, d)$ is a function translating a shared secret s into a vector of shared secrets with dimesion $d{+}1$ having a one at index s and 0 elsewhere (see function value2unaryconstraintX in [Sil03b]).

For incentive auctions (using Clarke tax), everything is similar with the case of 1st price auctions, except for:

- At the end of the last step, the price p_k to be paid by each agent A_k is not revealed but it is subtracted from w_0 (the shared secret representing the total weight of the solution to the DisWCSP, as returned by the used MPC-DisWCSPx) obtaining a shared secret $w'_0[k]$.
- The computation is run n more times, each k-th time by not considering the bids of the agent A_k, and recording the w_0 as $w_0[k]$. The price (Clarke tax) to be paid by each agent is given by $w_0[k] - w'_0[k]$.

6 Transforming first order logic in arithmetic circuits

Based on the experience with the examples analyzed so far, we conclude that with the new DisCSP framework it is useful to have a mechanism for automatic translation of first order logic sentences about secrets, into arithmetic circuits.

The main constructs in first order logic whose translation to arithmetic circuits will be given here are: $\forall i \in [1..n]P(i)$, $\exists i \in [1..n]P(i)$, $P \wedge Q$, $P \vee Q$, $\neg P$, $\min_{P(i)}(i)$, and $f = k$, where P and Q are predicates with a true (1) or false (0) value, f is a secret integer in a given interval, [1..n], i is a quantified variable that can take integer values in a given interval, [1..n], and k is a constant. They can also apply to variables and secrets from any finite set of numbers, $S = \{a_1, ..., a_n\}$. $\min_{P(i)} i$ is the function returning the minimum i such that $P(i)$ holds. The equivalent arithmetic circuits are shown in Table 1.

Let us exemplify how this translation applies to the first order logic predicate defining the global constraint for stable marriages with n males and m females, given in Equation 4. Consider that publicly $P_{A_i}(u, u)$=0 for all u and i. By applying the transformations for quantifiers, implication and conjunction, treating separately the case k=0 as well as the enforcement of distinct assignments, we

First Order Logic Sentence	Equivalent Arithmetic Circuit
P	$\overline{P}$
$\forall i \in [1..n], P(i)$	$\prod_{i=1}^{n} \overline{P(i)}$
$\forall a \in S, P(a)$	$\prod_{i=1}^{n} \overline{P(a_i)}$
$\exists i \in [1..n], P(i)$	$\sum_{i=1}^{n} [\overline{P(i)} \prod_{j=1}^{i-1} (1 - \overline{P(j)})]$
$\exists a \in S, P(a)$	$\sum_{i=1}^{n} [\overline{P(a_i)} \prod_{j=1}^{i-1} (1 - \overline{P(a_j)})]$
$P \wedge Q$	$\overline{P} * \overline{Q}$
$P \vee Q$	$\overline{P} + (1 - \overline{P})\overline{Q}$
$P \Rightarrow Q$	$1 - \overline{P}(1 - \overline{Q})$
$\neg P$	$1 - \overline{P}$
$f = k, (f, k \in [1..n])$	$\frac{1}{(k-1)!(n-k)!} \prod_{i=1}^{k-1} (f - i) \prod_{i=k+1}^{n} (i - f)$
$f = a_k, (f \in S, k \in [1..n])$	$\frac{\prod_{a_i \in S, i \neq k} (f - a_i)}{\prod_{a_i \in S, i \neq k} (a_k - a_i)}$
$\min_{P(i), i \in [1..n]} i$	$\sum_{i=1}^{n} [i\overline{P(i)} \prod_{j=1}^{i-1} (1 - \overline{P(j)})]$

Table 1: Equivalences between first order logic constructs and arithmetic circuits. P and Q are predicates and $\overline{P}$ and $\overline{Q}$ are their equivalent arithmetic circuits. $S = \{a_1, ..., a_n\}$.

obtain:

$$\phi \quad (\langle x_{\epsilon_1}, u_1 \rangle, ..., \langle x_{\epsilon_n}, u_n \rangle) =$$
$$\begin{cases} 0, & \text{when } u_i = u_j \text{ for different i,j, and } u_i \neq 0 \\ \prod_{i=1}^{n} ((1 - P_{A_i}(0, u_i)) * \\ \quad \prod_{k=1, k \neq u_i}^{m} (1 - P_{A_i}(k, u_i) * \\ \quad \quad (1 - \sum_{j=1, u_j = k}^{n} (P_{B_k}(j, i) \prod_{t=1, u_t = k}^{j-1} (1 - P_{B_k}(t, i)))))) & \text{otherwise} \end{cases}$$

The previous arithmetic circuit is obtained by simply applying the transformations in Table 1 to the first order logic definition in Equation 4. It can be noted that since the variables are constrained to take distinct values, the arithmetic circuit can be written in a simpler equivalent form:

$$\phi \quad (\langle x_{\epsilon_1}, u_1 \rangle, ..., \langle x_{\epsilon_n}, u_n \rangle) =$$
$$\begin{cases} 0, & \text{when } u_i = u_j \text{ for different i,j, and } u_i \neq 0\text{; otherwise} \\ \prod_{i=1}^{n} ((1 - P_{A_i}(0, u_i)) \prod_{k=1, k \neq u_i}^{m} (1 - P_{A_i}(k, u_i)(1 - \sum_{j=1, u_j = k}^{n} (P_{B_k}(j, i))))) & \end{cases}$$

The total number of multiplications needed to construct this global constraint is $O(mn^{m+1})$, namely mn multiplications for each of the n^m tuples. A public constraint for this problem is:

$$\phi_0(\langle x_{\epsilon_1}, u_1\rangle, ..., \langle x_{\epsilon_n}, u_n\rangle) \overset{\text{def}}{=} \begin{cases} 0, & \text{when } u_i{=}u_j \text{ for different i,j and } u_i \neq 0 \\ 1, & \text{otherwise} \end{cases}$$

6.1 Complexity

For a problem with size of the search space Θ and c constraints, the number of messages for finding all solutions with secure techniques similar to the one in [HCN$^+$01] is given by $(c-1)\Theta$ multiplications of shared secrets ($n(n{-}1)$ messages for each such multiplication). For the desk-mates problem modeled with the new framework, $\Theta{=}m^m$ and $c{=}1$ for the version with a single global constraint, or $c{=}m^2/2$ for the version with binary constraints. For the case with binary constraints, it yields a complexity of $O(m^{m+2})$. As mentioned before, the preprocessing has complexity $O(m^4)$ multiplications between shared secrets, resulting in a total complexity $O(m^2(m^m+m^2))$.

Solving the same problem with the same algorithm but modeled with the old DisCSP framework with private constraints, $\Theta = m^m 2^{m^3}$ and $c = m$, for one global constraint from each agent. There is no preprocessing, but the total complexity is $O(m^{m+1}2^{m^3})$. The new framework behaves better since $m << 2^{m^3}$. The comparison is similar for other secure algorithms, like MPC-DisCSP1 (see Section 2.2) whose complexity is given by $O(dm(c{+}m)\Theta)$ multiplications between shared secrets.

7 Conclusions

DisCSPs [BMM01, SGM96, LV97, Ham99, MR99, ZWW02, BD97, FBKG02, MTSY04] are a very active research area. Privacy has been recently stressed in [MJ00, BB01, FMW01, WF02, FMG02, YSH02b] as an important goal in designing algorithms for solving DisCSPs.

In this article we have investigated how versions of old and famous problems, incentive auctions and the stable marriages problems [GS62, Ski90], can be solved such that the privacy of the participants is guaranteed except for what is leaked by the selected solution. Incentive auctions are a very intense field of research and application of agents and economic theories. Techniques for stable marriages problems are currently applied to college admissions and medical interns assignments in US. Our technique uses secure simulations of arithmetic circuit evaluations and is therefore robust whenever no majority of the participants colludes to find the secret of the others, and when all agents follow the protocol.

We note that the desk-mates and the stable marriages problems cannot be efficiently modeled (at least not in an obvious way) with existing distributed constraint satisfaction frameworks. We have therefore introduced a new distributed constraint satisfaction framework that can model such problems with the same search space size as the classic centralized CSP models. We have shown how some techniques for the existing frameworks can be adapted to problems modeled with the new DisCSPs, and we exemplify the model with the desk-mates, stable marriages, and incentive auctions problems. For m participants in the desk-mates problem, the size of the search space in the DisCSP model achieved with the new framework is $O(m^m)$ while the previous framework with private constraints yields DisCSP instances with a size of the search space of $O(m^m 2^{m^3})$. A similar ratio is obtained for the stable marriages problem. In existing secure algorithms for solving DisCSPs, the number of exchanged messages is fix and directly proportional to the search space size, making this property of a problem instance particularly relevant.

References

[BB01] G. Bella and S. Bistarelli. Soft constraints for security protocol ananysis: Confidentiality. In *Third International Symposium on Practical Aspects of Declarative Languages*, number 1990 in LNCS, 2001.

[BD97] B. Baudot and Y. Deville. Analysis of distributed arc-consistency algorithms. Technical Report RR-97-07, U. Catholique Louvain, 1997.

[BMM01] C. Bessière, A. Maestre, and P. Meseguer. Distributed dynamic backtracking. In *CP*, page 772, 2001.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computating. In *STOC*, pages 1–10, 1988.

[CDK91] Z. Collin, R. Dechter, and S. Katz. On the feasibility of distributed constraint satisfaction. In *Proceedings of IJCAI 1991*, pages 318–324, 1991.

[FBKG02] C. Fernàndez, R. Béjar, B. Krishnamachari, and C. Gomes. Communication and computation in dis. CSP algorithms. In *CP*, pages 664–679, 2002.

[FMG02] B. Faltings and S. Macho-Gonzalez. Open constraint satisfaction. In *CP*, 2002.

[FMW01] E.C. Freuder, M. Minca, and R.J. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *Proc. IJCAI DCR*, pages 63–72, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge, 2004.

[GS62] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematics Monthly*, 69:9–14, 1962.

[Ham99] Youssef Hamadi. *Traitement des problèmes de satisfaction de contraintes distribués*. PhD thesis, Université Montpellier II, Juillet 1999.

[HCN$^+$01] T Herlea, J. Claessens, G. Neven, F. Piessens, B. Preneel, and B. Decker. On securely scheduling a meeting. In *Proc. of IFIP SEC*, pages 183–198, 2001.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[LV97] Michel Lemaître and Gérard Verfaillie. An incomplete method for solving distributed valued constraint satisfaction problems, 1997.

[MJ00] P. Meseguer and M. Jiménez. Distributed forward checking. In *DCS*, 2000.

[MR99] Eric Monfroy and Jean-Hugues Rety. Chaotic iteration for distributed constraint propagation. In *SAC*, pages 19–24, 1999.

[MTSY04] P.J. Modi, M. Tambe, W.-M. Shen, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 2004.

[SGM96] G. Solotorevsky, E. Gudes, and A. Meisels. Algorithms for solving distributed constraint satisfaction problems (DCSPs). In *AIPS96*, 1996.

[Sha79] A. Shamir. How to share a secret. *Comm. of the ACM*, 22:612–613, 1979.

[Sil02] Marius-Călin Silaghi. *Asynchronously Solving Distributed Problems with Privacy Requirements*. PhD Thesis 2601, (EPFL), June 27, 2002. http://www.cs.fit.edu/˜msilaghi/teza.

[Sil03a] M.C. Silaghi. Arithmetic circuit for the first solution of distributed CSPs with cryptographic multi-party computations. In *IAT*, Halifax, 2003.

[Sil03b] M.C. Silaghi. Solving a distributed CSP with cryptographic multi-party computations, without revealing constraints and without involving trusted servers. In *IJCAI-DCR*, 2003.

[Ski90] S. Skiena. *Stable Marriages*, chapter 6.4.4, pages 245–246. AW, 1990.

[SR04] M. Silaghi and V. Rajeshirke. The effect of policies for selecting the solution of a DisCSP on privacy loss. In *AAMAS*, 2004.

[SSHF00a] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proc. of AAAI2000*, pages 917–922, Austin, August 2000.

[SSHF00b] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with private constraints. In *Proc. of AA2000*, pages 177–178, Barcelona, June 2000.

[WF02] R.J. Wallace and E.C. Freuder. Constraint-based multi-agent meeting scheduling: Effects of agent heterogeneity on performance and privacy loss. In *DCR*, pages 176–182, 2002.

[WS04] R. Wallace and M.C. Silaghi. Using privacy loss to guide decisions in distributed CSP search. In *FLAIRS'04*, 2004.

[Yao82] A. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.

[YDIK98] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE TKDE*, 10(5):673–685, 1998.

[YSH02a] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *Proc. of the AAMAS-02 DCR Workshop*, Bologna, July 2002.

[YSH02b] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP*, 2002.

[ZM91] Y. Zhang and A. K. Mackworth. Parallel and distributed algorithms for finite constraint satisfaction problems. In *Proc. of Third IEEE Symposium on Parallel and Distributed Processing*, pages 394–397, 1991.

[ZWW02] W. Zhang, G. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *PAS*, 2002.