Hiding absence of solution for a Distributed Constraint Satisfaction Problem

Marius-Călin Silaghi
Florida Institute of Technology

Abstract

A distributed constraint satisfaction problem (DisCSP) is defined by a set of agents, trying to find assignments for a set of variables with known domains and subject to secret constraints. They can model applications like auctions, distributed team-making, scheduling, and configuration.

Secure multiparty computations (MPC) can pick and reveal randomly one of the solutions of any distributed constraint satisfaction and optimization problem. Our previous algorithms are based on an arithmetic circuit for selecting a random element out of the elements with a given value in a secret array (Silaghi 2004a). Here we show an improvement based on a very simple and elegant (optimized) version of the involved functions and on the usage of CSP solvers to exploit public constraints. The technique can hide the absence of solutions.

The algorithm proposed here is called MPC-DisCSP4. Compared to previous techniques, MPC-DisCSP4 is faster and can hide the absence of solutions.

Definition 1 *A Distributed CSP is defined by five sets (A, X, D, C, O). 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 domains such that x_i can take values only from $D_i = \{v^i_1, ..., v^i_{d_i}\}$. $C = \phi_0 \cup \{\phi_1, ..., \phi_c\}$ is a set of constraints. ϕ_i involves an ordered subset $X_i = \{x_{i_1}, ..., x_{i_{k_i}}\}$ of the variables in X, $X_i \subseteq X$, and constrains the legality of each combination of assignments to the variables in X_i. An assignment is a pair $\langle x_i, v^i_k\rangle$ meaning that variable x_i is assigned the value v^i_k. Each constraint ϕ_i, $i > 0$, is known only by one agent, being the secret of that agent. There may exist c_0 public constraints in C, ϕ_0=$\{\phi^1_0, ..., \phi^{c_0}_0\}$.*

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 DisCSP (X,D,C) is a tuple of assignments ϵ with one assignment for each variable in X such that each $\phi_i \in C$ is satisfied by $\epsilon_{|_{X_i}}$ (written $\phi_i(\epsilon_{|_{X_i}})$).

MPC-DisCSP4. Figure 1 gives an overview of MPC-DisCSP4, also shown in Algorithm 1. MPC-DisCSP4 is composed of a CSP solver, a mix-net, and a set of functions

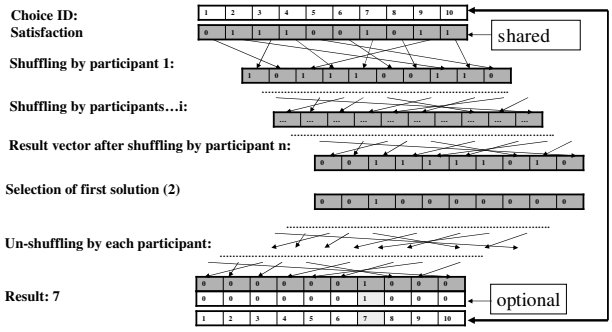

Figure 1: Pictorial view for MPC-DisCSP4. The reconstructions of secrets at the end is optional, as it will not be performed when the solver is integrated in other techniques (optimization, auctions). Dark gray show shared secrets.

(arithmetic circuits) that are much more elegant and simple than those used in previous versions.

MPC-DisCSP4 starts by sharing the encoded constraints with the Shamir secret sharing scheme. A complete CSP solver (e.g., backtracking) is used to generate the vector: $S'' = \{\epsilon | \phi_0(\epsilon)\}$. I.e. S'' contains all tuples ϵ that satisfy ϕ_0. Each tuple of assignments candidating as solution to the input problem is refered by an index (e.g. see the first array in Figure 1, with elements from 1 to 10). The agents share the secret values of their constraints. Let ϵ_k denote the k^{th} tuple in the lexicographic order. They compute for each such tuple ϵ_k, a shared secret whose unknown value is 1 when the tuple satisfies all agents and 0 otherwise. This is done by using (Ben-Or, Goldwasser, & Widgerson 1988) to securely simulate the evaluation of the arithmetic circuit: $p(\epsilon_k) = \prod_{i=1}^{c} \phi_i(\epsilon_k)$. The results form the second array of the figure, S', $S'[i] = p(S''[i])$ (each participant obtains a distinct vector with a share for each element of S'). S' is now shuffled and the shares are randomized with a mix-net whose details are given later. We define:

$$\begin{aligned} h_1(P) &= 1 \\ h_i(P) &= h_{i-1}(P) * (1 - S'[i-1]) \end{aligned}$$

The lexicographically first solution can be isolated:

$$S[i] = S'[i] * h_i(P) \quad (1)$$

A vector, S, is computed with Equation 1. It is then decoded by traversing the mix-net in the inverse direction and

procedure *MPC-DisCSP4* **do**

1. Shares the {0,1} encoded secret constraints.
2. Compute all tuples ϵ returned by the CSP solver for ϕ_0. Place the results lexicographically in vector S''.
3. Compute a vector S', $S'[k] = p(S''[k])$.
4. The mix-net shuffles the vectors of shares, S', randomizing the shares at each permutation (by adding shares of 0) exploiting homomorphic encryption.
5. Compute vector S with Equation (1).
6. The mix-net decodes S randomizing the shares and inversing permutations of Step 5.
7. Compute Equations 2.
8. Reveal assignments to the interested agents.

Algorithm 1: MPC-DisCSP4

with the inverse permutations, randomizing the shares as at shuffling. The solution is the tuple of S'' situated at the same index as the only element of S that is different from 0.

The solution can also be transformed into a set of indexes of values of the variables by the following method. Assume the value of the u^{th} variable in the t^{th} tuple of the search space is denoted $\eta_u(t)$. The values in the solution are computed with the arithmetic circuits in Equation (2).

$$f_i(P) \quad = \quad \sum_{t=1}^{|S''|} (\eta_i(t) + 1) * S'[t-1] \qquad (2)$$

Each variable x_i is assigned in the solution to the value in D_i at index given by the functions f_i, and can be revealed.

MPC-DisCSP4's mix-net for reordering vectors of shared secrets. Each agent A_i chooses a random secret permutation π_i, picked with a uniform distribution over the set of possible permutations: $\pi_i : [1..|S''|] \rightarrow [1..|S''|]$. Each agent chooses a pair of keys for a $(+, \times)$-homomorphic public encryption scheme and publishes the public key. The secret shares, of the values computed in the vector S', are encrypted by each A_i with her public key and are serialized. The serialized encrypted vectors are sent to A_1. A_1 shuffles the serialized vectors according to her permutation π_1, then passes them to A_2 which applies π_2, etc., until the agent A_n which applies π_n. A_n sends each vector to its owner.

To avoid that agents get a chance to learn the final permutation by matching final shares with the ones that they encrypted, a randomization step is also applied at each shuffling. Each agent applies a randomization step on the set of shares for each element of S', by adding corresponding shares of zero. Since operands are encrypted, to perform this summation we exploit the $(+, \times)$-homomorphic property. For each secret in S', a new set of 0's Shamir shares are generated, and $\forall i, i \leq n$, the $0's$ i^{th} share is encrypted with the public key of A_i, then it is multiplied to the corresponding A_i's encrypted share of the secret (resulting in resharing the secret). This assumes $\mu > \nu(n+1)$, for the decryption to be correct in $\mathbb{Z}_\nu$. $(\mathbb{Z}_\mu, +)$ is the group with possible plaintexts for the homomorphic encryption. $(\mathbb{Z}_\nu, +, \times)$ is the field in which that arithmetic circuits are evaluated.

Example 1 *Let us see an example of how MPC-DisCSP4 is applied to the Example 2.* $p(P, W)$ *is not computed (*ϕ_0*).*
$p(P, T)$=1, $p(Q, T)$=0, $p(Q, W)$=1.
S"=((P,T),(Q,T),(Q,W)) S'=(1,0,1)
Shuffle (1,0,1), (assume it remains unchanged)
$h_1(P)$=1, $h_2(P)$=0, $h_3(P)$=0.
With Equation 1 we get vector S={*1,0,0*}.
Unshuffle S=(1,0,0): S=(1,0,0)
Using Equation 2: $\eta_1(1)$=0, $\eta_1(2)$=1, $\eta_1(3)$=1. $\eta_2(1)$=0, $\eta_2(2)$=0, $\eta_2(3)$=1. $f_1(P)$=1, $f_2(P)$=1.
Therefore the chosen solution is x_1=*Paris,* x_2=*Tuesday.*

Besides the mixnet, the most expensive operation — requiring exchanges of messages — is the muliplication of two shared secrets. MPC-DisCSP4 has $2(|S''|-1)$ such multiplications. $|S''|-1$ multiplications to compute the h series, and $|S''|-1$ multiplications to obtain S. The number of such multiplications in MPC-DisCSP3 is $5(|S''|-1)$. This corresponds to a 2.5 times improvement. Preparing the vectors S' has a cost proportional to $c|S''|$.

In contrast to previous techniques, the most expensive computation in MPC-DisCSP4 is due to the mix-nets. Their efficiency was improved by the use of the CSP solver to prune tuples that are inconsistent with public constraints.

Hiding the (in)existance of a solution. To hide the fact that a solution does not exist for a problem, the participants may prefer to miss an existing solution with some probability p. Then, the fact that no solution is found does not prove that no solution exist, and certain leaks of secrets induced by the lack of a solution are avoided. This can be achieved by generating unknown shared number(s) K that with probability p equal 0, and otherwise equal 1 (Silaghi 2004b).

Now, each of the secret elements of the vector S (or all the secret values returned by f_i) are multiplied with such a number K, losing the solution with probability p.

Conclusions. MPC-DisCSP4 can offer a privacy that is stronger than what is offered by the previous methods for DisCSPs. The solution is picked randomly over the set of existing solutions, hiding statistical information about whether other tuples are solutions or not. If needed, one may choose to hide the information about whether a solution exists at all, by probabilistically discarding the solution.

The computation cost on shuffled problems is 2.5 times faster with the new technique than with MPC-DisCSP3, which has the closest performance in privacy. As with MPC-DisCSP3, by taking advantage of public constraints, the efficiency of the most expensive operations (public key encryptions), is reduced proportionally with their tightness.

References

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

Silaghi, M.-C. 2004a. Meeting scheduling system guaranteeing n/2-privacy and resistant to statistical analysis (applicable to any DisCSP). In *IC on Web Intel.*, 711–715.

Silaghi, M. C. 2004b. Secure multi-party computation for selecting a solution according to a uniform distribution over all solutions of a general combinatorial problem. Cryptology e-print Archive 2004/333.