Distributed Constraint Satisfaction – A secure and private approach

Yogesh Piolet T

Abstract— A Constraint Satisfaction Problem (CSP) consists of a finite number of variables, each of which can take values only from a specified domain and a set of constraints on their values. A CSP is said to be solved when we arrive at a consistent assignment of values to the variables such that none of these constraints are violated. The Distributed Constraint Satisfaction Problem (DisCSP) is a CSP in which variables and constraints are distributed among multiple agents. It consists of a set of agents. Each of the agents has a CSP that is solved consistently with solutions to the CSP's of other agents. Each of these CSP's has a set of variables that belong to a particular agent. Also, they consist of intra-agent constraints (between the variables of a particular agent) and inter-agent constraints (between the variables belonging to different agents). A solution to the DisCSP is a set of solutions to all of the agents' CSPs, i.e., a state where all of the agents find sets of assignments of values for their local variables whereby no intra/inter-agent constraint is violated. One of the trivial methods of solving a DisCSP is the centralized method. In this method, the agents select a leader among themselves. Each of the agents would then send their variable domains and constraints to the leader. The leader then solves the problem alone and sends the result back to the agents.But, the drawback of this approach is that it incurs huge amount of communication overhead. That apart, this method is not desirable for ***security and privacy*** reasons. It is important to note that some agents may not be willing to share their information with others. The major motivation of this paper is to find a solution to provide some level of security to the agents. It could be possible to allow the agents to send some information and not all of them. This major aim of this paper is to develop a much more secure distributed constraint satisfaction algorithm. It is possible to prevent the agents from leaking information to other agents. A few of the standard encryption algorithms could be used to solve the problem.

Index Terms— homomorphic encryption, privacy, secure distributed constraint satisfaction problem, security.

---

1 INTRODUCTION

A Constraint Satisfaction Problem (CSP) is a problem in which there exists a set of n number of variables $x_1, x_2, \ldots, x_n$ in which each of these variable x_i can take its value from only a specific domain D_i bounded by a set of constraints on their values. A constraint is said to be a *nogood*, i.e., a set of values for some variables that are prohibited for the variables. A nogood is violated when its corresponding variables actually take the values appearing in the nogood. A solution to the CSP is an assignment of values for all of the variables whereby no nogood is violated. The problem of finding a solution to the CSP is known to be NP-complete.

The Distributed Constraint Satisfaction Problem (DisCSP) is a CSP in which variables and constraints are distributed among multiple agents. It consists of the following:

- a set of agents, 1, 2, . . .,k
  where each agent could be a process, processor or computer that is autonomous and interconnected by an underlying network. In general, agent is a producer of an effect.
- a set of CSPs, $P_1, P_2, \ldots, P_k$, such that P_i belongs to agent i and consists of
  - a set of local variables whose values are controlled by agent i ,
  - a set of intra-agent constraints, each of which is defined over agent i's local variables,
  - a set of inter-agent constraints, each of which is defined over agent i's local variables and other agents' local variables.

A solution to the DisCSP is a set of solutions to all of the agents' CSPs, i.e., a state where all of the agents find sets of assignments of values for their local variables whereby no intra/inter-agent constraint is violated. Obviously, the problem of finding a solution to the DisCSP is NP-complete.

1.1 Terminology

- Agent - A producer of an effect. It could be a process, processor, computer, human being or objects.
- Public key encryption - A cryptographic system that uses two keys -- a public key known to everyone and a private or secret key known only to the recipient of the message.

1.2 Literature survey

Over the past few years, researchers have been constantly focussing on solving the DisCSP optimally. The researchers have been successful in bringing out a solution to the DisCSP mostly by utilizing the concept of backtracking. Some of the initial algorithms that were

• *Yogesh Piolet T is with the School of Information Technology, Indian Institute of Technology Kharagpur, Kharagpur, India-721302. E-mail: yogeshpiolet@yahoo.co.in*

proposed to solve the DisCSP based on backtracking are Synchronous Backtracking [2], Asynchronous Backtracking [2], Asynchronous Weak Commitment Backtracking [2], Distributed Breakout Algorithm [4,5] and a few others.

Now that the researchers have successfully found out a solution to the DisCSP, they have started focusing on providing security while arriving at the solution to the assignment of values to the variables in each of the agent. Security is becoming an increasing concern in this domain of research.

(Yokoo et al., 1998) proposed a centralized approach of solving the distributed problem. It is a very trivial approach. It begins with trying to select a leader agent among all agents, and gather all information about the variables, their domains, and their constraints, into the leader agent. The leader then solves the CSP alone using normal centralized constraint satisfaction algorithms. But, there are serious drawbacks to this approach .Some of them are mentioned below: The cost of collecting all information about a problem can be prohibitively high. Moreover, gathering all information to one agent is undesirable or impossible for security/privacy reasons. The process of collecting all information about a problem requires not only the communication costs but also the costs of translating one's knowledge into an exchangeable format. For example, a constraint might be stored as a very complicated specialized internal function within an agent. In order to communicate the knowledge of this constraint to other agent, which might be implemented on different computer architecture, the agent must translate the knowledge into an exchangeable format, such as a table of allowed (or not allowed) combinations of variable values. These costs of centralizing all information to one agent could be prohibitively high.

An illustrating example for this problem is the meeting scheduling problem. In this problem, there is a group of people who want to meet some other people in the same group. Hence, each of these people is an agent. But, a person cannot attend multiple meetings at the same time. This forms the constraint.

The above problem can be modelled as a DisCSP as follows-

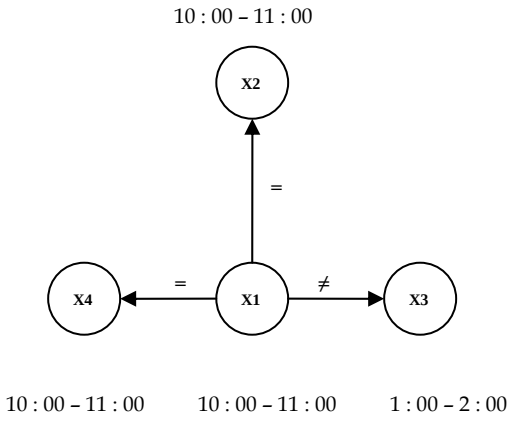

Fig 1 . Example of DisCSP

In the above example, x_1 , x_2 , x_3 , x_4 are the agents (persons). x_1 meets x_2 and x_4 in the same meeting. x_1 cannot meet x_3 in the same session. Thus, the problem is modeled using the variables x_1 , x_2 , x_3 , x_4 . The domains of each of these variables are the time slots and date. The constraints can be inequality (a person cannot attend multiple meeting at the same time) or equality (they must meet at the same day/time). One possible solution is shown in Fig 1.

The drawback of this approach is very clear. Each agent would make proposal on the date and time that he is willing to meet other people. This would reveal the fact that the particular agent does not have any personal schedule/meeting on that particular day. Suppose a person declines a date, it would easily indicate that he is busy on that date. This is the manner in which the private information about each agent is leaked to other agent.

Thus, our goal is to develop an algorithm that does not leak any private information about individual agents.

2. Problem Definition

Based on the survey done, it can be observed that security is a major concern in the area of distributed systems, particularly in the area of DisCSPs. It is necessary to formalize the problem as follows: An algorithm is said to be not leaking any private information if an agent cannot obtain any additional information on the assignment of values to variables that belong to other agents. Thus, with reference to the example shown in Fig 1, each person must not be able to know the scheduled dates of the meetings he/she will not attend. Also, it is necessary to ensure that he/she cannot obtain any information on the private schedules of other participants. Thus, each agent can only know the assignment of values to its own variables and cannot obtain any additional information on the assignment of values to variables that belong to other agents.

3. Existing Solution

A solution given by (Yokoo et al., 2005) is explained here.Their approach used the following strategy. (Yokoo et al., 2005) assumed that each agent had exactly one variable. They considered only binary constraints between all pairs of variables. These assumptions were done without loss of generality as it can be observed that if an agent has multiple variables, it can be easily treated as a single combined variable whose domain is the product of the domains of original variables.

(Yokoo et al., 2005) developed a secure DisCSP algorithm using computational servers called search controller and decryptors. Atleast two decryptors must be used in the distributed system. The decryptors performed decryption of the messages sent between the agents using Elgamal encryption [7]. Each decryptor consists of a part of the secret key and a public key. Only one search controller is sufficient for a distributed system.

The procedure is as follows :
The agents encode the unary/binary constraints and pass the information to the servers. Then, the servers solve the DisCSP and return the assignment of values to each agent.

The flowchart of the algorithm is proposed by (Yokoo et al., 2005) is shown in Fig. 2

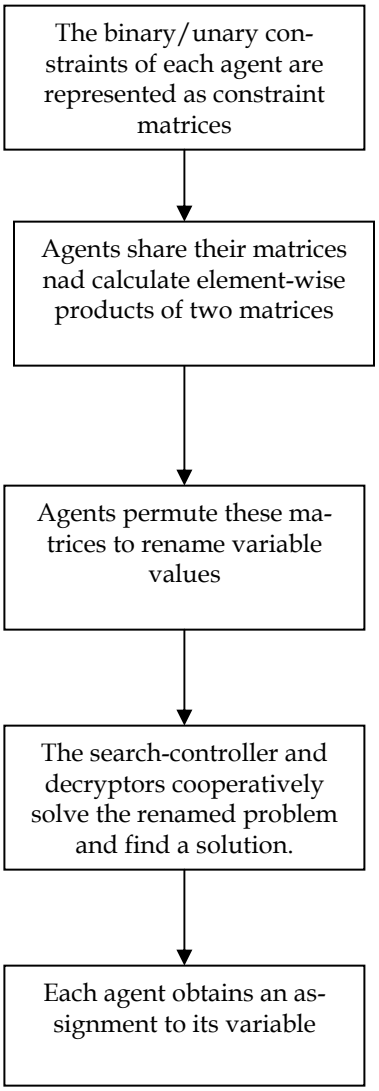

Fig 2 Flowchart of secure DisCSP algorithm (Yokoo et al., 2005)

Let binary constraints between x_i and x_j in an agent i be represented as $C_{x^j_i, x_j}$ by the use of a *m x m* constraint matrix. Let us denote this matrix as $A_{i,xi,xj}$.
Any element of this matrix matrix $A_{i,xi,xj}(k, l)$ is defined as follows. $E(1)$ and $E(z)$ denote the encryption of 1 and a common public element z ($\neq 1$), respectively. The encryption that (Yokoo et al., 2005) used was Elgamal encryption. However, the algorithm can be implemented with other public key encryption algorithms which satisfy certain properties such as indistinguishability, homomorphism and randomizability. For example, Paillier encryption [8] could also be used.

To illustrate the entire process, let us take a distributed 4-Queen problem as an example. Let us assume that agent1 knows only the diagonal constraints between x1 and x2 and agent 2 knows only the column constraints between x1 and x2.

Queen/ position	1	2	3	4
X1	Q			
X2	Q	X	Q	Q

Fig 3. Distributed 4-queens problem with respect to agent x1

In Fig. 3, if X1 takes first position, X2 may take either first, third or fourth position. This is because X1 knows only diagonal constraints.

3.1 Construction of constraint matrix

The constraint matrix $A_{i,xi,xj}$ is constructed using following rules:

1. $A_{i,xi,xj}(k, l) = E(z)$ if $x_i = k$ and $x_j = l$ are inconsistent, i.e., nogood$\{x_i = k, x_j = l\}$ is in $C_{x^j_i, x_j}$, or k violates i's unary constraint, i.e., nogood$\{x_i = k\}$ is in $C_{x^j_i}$.
2. $A_{i,xi,xj}(k, l) = E(1)$ if $x_i = k$ and $x_j = l$ are consistent.

Hence, the 4-Queen's problem with agent x1 knowing only diagonal constraints and agent x2 knowing only column constraints reduces as follows :

X1 pos/ X2 pos	1	2	3	4
1	E(1)	E(z)	E(1)	E(1)
2	E(z)	E(1)	E(z)	E(1)
3	E(1)	E(z)	E(1)	E(z)
4	E(1)	E(1)	E(z)	E(1)

Fig 4. X1's constraint matrix

X1 pos/ X2 pos	1	2	3	4
1	E(z)	E(1)	E(1)	E(1)
2	E(1)	E(z)	E(1)	E(1)
3	E(1)	E(1)	E(z)	E(1)
4	E(1)	E(1)	E(1)	E(z)

Fig 5. X2's constraint matrix

The above two matrices can be combined as follows:

X1 pos/ X2 pos	1	2	3	4
1	E(z)	E(z)	E(1)	E(1)
2	E(z)	E(z)	E(z)	E(1)
3	E(1)	E(z)	E(z)	E(z)
4	E(1)	E(1)	E(1)	E(z)

Fig 6. combined constraint matrix

3.2 Permutation

The above constraint matrix is reordered by using the following technique.

X1 changes rows 1 to 2, 2 to 4, 3 to 1 and 4 to 3 as shown in Fig. 7

X1 pos/ X2 pos	1	2	3	4
1	E(1)	E(z)	E(z)	E(z)
2	E(z)	E(z)	E(1)	E(1)
3	E(1)	E(1)	E(z)	E(z)
4	E(z)	E(z)	E(z)	E(1)

Fig 7. permuted constraint matrix by x1

X1 then sends the permuted constraint matrix to X2. X2 changes columns 1 to 3, 2 to 1, 3 to 4 and 4 to 2 as shown in Fig. 8

X1 pos/ X2 pos	1	2	3	4
1	E(z)	E(z)	E(1)	E(z)
2	E(z)	E(1)	E(z)	E(1)
3	E(1)	E(z)	E(1)	E(z)
4	E(z)	E(1)	E(z)	E(1)

Fig 8. permuted constraint matrix by x2

This encoded matrix is then sent to the search controller along with a key . The search controller then sends the encoded matrix to the decryptors, each of which have a share of the secret key. The decryptor calculates the result and sends to the search controller. The search controller uses the key sent by the agent along with encoded matrix to encode the final result that is obtained from all the encryptors and send it to the agent. The agent will once again permute the matrix as it was done earlier to obtain its actual solution.

4. New suggestion

The algorithm provides a good solution to the problem, although some de-merits have been observed. From the above explaination, it is guaranteed that the algorithmm does not leak any additional information to the agents. This is mainly because the actual problem of an agent is modified during the permutation phase. In addition to this, an agent does not participate in the search for a solution. Therefore, the knowledge abtained by an agent from the constraint matrix is same as if it had obtainned by some imaginary supernatural means.

Also, the algorithm does not leak any information to the centralized servers i.e. search controller and decryptor(s). It also ensures that the server has no knowledge of the kind of problem it is dealing with as the problem itself is given to it in the form of an encrypted constraint matrix. However, an assumption is made the server does not have any a priori knowledge of the problem.

Some de-merits that have been observed in the above proposed methodology are pointed out below. The performance of this algorithm is equivalent to that of a chronological backtracking. The complexity of the algorithm is very high. Hence, the algorithm is inefficient anddoes not make complete use of the distributed architecture. The main cause for this is the centralized third-party server such as search controller and decryptor(s). Also, this method is not free against collusion. Suppose the search controller and one of the agents collude, then information about all the assignments can be obtained by the search controller. Also, the algorithm is not robust.

In our example, the search controller and the decryptor(s) cannot get any idea that it is solving a N-Queen's problem. The factors that help in achieving this is the renaming/permutation operation that is performed by each agent before sending the constraint matrix to the server.

The solution that is suggested is as follows. Rather than sharing the knowledge about the constraints in the form of a constraint matrix, an agent can share only the subdomain of values that the opposite agent can take.

Mathematically, the following explaination gives an idea about the concept. Let *A* and *B* be two agents with only variable each x_a *and* x_b. Let agent A assign x_a a value from its domain D_a and search for the subdomain of values from domain D_a such that the subdomain of values would satisfy the constraints with agent *B*. Agent *A*, later sends the newly obtained subdomain of values to agent *B*. On the other hand, agent *B* on the receipt of the compatible subdomain would assign itself a value from the subdomain. At this point, instead of sharing the newly obtained solution with the agent *A*, it sends agent *B* another subdomain which satisfy the constraints with *A*. On the receipt of the subdomain, Agent *A* would check whether its assigned value is in the same subdomain. If the assigned value is in the same subdomain, then agent *A* has found a compatible solution for that constraint at the present instance of time. Otherwise, the process repeats with the agent *A* once again assigning a new value and sending the compatible subdomain to *B*.

The major advantage of this approach is that an attempt has been made to solve the distributed constraint satisfaction problem privately without the use of any centralized server such as search contoller or decryptor(s). But, a problem that is not yet solved is that of eavesdropping when the subdomain of values is exchanged between agents. This can be achieved by encrypting the subdomain of values using any public key encryption algorithm. The selection of the encryption algorithm is not restricted by constraints such as homomorphism. Hence, a wider range of encryption algorithms are available to choose from specific to the applications.

5. Analysis and Discussion

The major de-merit of the methodology proposed by (Yokoo et al., 2005) was that it was still following the centralized approach despite the heavy efficiency penality it imposed. Hence, with respect to that issue, the newly proposed methodology has been quite successful. The algorithm provides an opportunity to explore the solution to the problem in a distributed manner.

But, the message complexity of the newly proposed algorithm could impose a restriction to its use in a highly congested network. The number of messages that is required to solve each constraint is a minimum of 2. Since, each constraint is represented by an edge, the total number of messages required to solve the problem would be 2|E| in the best case. The best case happens when each agent assigns its variables a value only once and the violation of a constraint never happens. But, if the constraint violations are bound to occur, then 1 message is exchanged per constraint violation.

Without loss of generality, let $d_1, d_2 .. d_n$ be the number of constraint violations for each constraint $C_1, C_2, .. C_n$ such that $d_1+d_2+..+d_n=d$. Thus, the total number of messages required is

$$(\sum_{i=1}^{n} d_i + 1)*|E| \quad \text{.....1.0}$$

The "1" in the equation 1.0 corresponds to the initial assignment. It can be easily observed that if the number of constraint violations is high, higher number of messages will be exchanged.

6. Conclusion

In this paper, the distributed constraint satisfaction problem was discussed. The motivation to provide a solution to DisCSP in a secure manner was briefed out. The methodology of solving DisCSP in a secure manner proposed by (Yokoo et al., 2005) was explained in detail. An alternative approach to solve DisCSP was explained later.

The algorithm tackles issues in providing a secure approach to solving a distributed constraint satisfaction problem. However, additional work is to be done to provide a solution in such a way that the message complexity is reduced even when the number of constraint violations is high.

ACKNOWLEDGMENT

The author wishes to thank Prof. Soumya K Ghosh, School Of Information Technology, Indian Institute Of Technology Kharagpur for providing a foundational basic and motivation to write this paper. The author also thanks Mr. Gulabrao Ashutosh Shinde for good advice. This work was performed as part of the curriculum for the course "Information and System Security".

REFERENCES

[1] Makoto Yokoo , Koutarou Suzuki , Katsutoshi Hirayama, Secure distributed constraint satisfaction: reaching agreement without revealing private information, Artificial Intelligence, v.161 n.1-2, p.229-245, January 2005

[2] M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: Proceedings of the 12th IEEE International Conference on Distributed Computing Systems, 1992, pp. 614–621.

[3] M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, The distributed constraint satisfaction problem: formalization and algorithms, IEEE Trans. Knowledge Data Engrg, 1998, pp. 673–685.

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

[5] W. Zhang, L.Wittenburg, Distributed breakout revisited, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Edmonton, AB, 2002.

[6] Chris Studholme, Minesweeper as a constraint satisfaction problem, Unpublished project report, 2000.

[7] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inform. Theory IT-31 (4) (1985) 469–472.

[8] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proceedings of EUROCRYPT-99, 1999, pp. 223–238.

Yogesh Piolet T is a student pursuing his postgraduate degree in Master of Technology (Information Technology) at the School of Information Technology in Indian Institute of Technology Kharagpur. The above work is presented as part of the curriculum for the course “Information and System Security”. He bears a registration number of 08IT6018.