# A multi-factor concept on guiding constraint relaxation in Distributed Constraint Satisfaction

JINGXUAN MA SYLVAIN PIECHOWIAK RENE MANDIAU
LAMIH-UMR CNRS 8530
Université de Valenciennes et du Hainaut-Cambrésis
59313 Valenciennes Cedex 9 *France*

*Abstract:* - Over-constrained can usually be encountered when some real-life applications are described as Distributed Constraint Satisfaction (DisCSP). When a solution is required in this case, one or more constraints should be released. This paper focuses on the constraint relaxation for handling such an over-constrained situation. Three types of influence factor, variable influence factor, constraint influence factor, agent influence factor, as well as their relations are introduced to guide the cooperation between agents, and to guide the constraint relaxation in case of over-constrained. Finally, the feasibility of proposed concept on the constraint relaxation during problem solving in a hierarchical agent-structure is illustrated.

*Key words:* -CSP, Constraint relaxation, DisCSP, Agent, Influence factor

## 1. Introduction

A Constraint Satisfaction Problem (CSP) is a type of problem in which the goal is to find the values for a set of variables that will satisfy a given set of constraints [1]. While a CSP is divided into a set of sub-CSP and is resolved by the cooperation of a set of agents, each of which is a CSP solver, this CSP is called Distributed CSP (DisCSP). A solution of a DisCSP is an assignment of values to variables which satisfies not only the local constraints existing in each agent, but also the global constraints existing between agents [2]. Many of application problems in multi-agent systems can be formalized as DisCSP. Distributed Resource Allocation [3] and distributed sensor networks [4] are some examples of these types of applications.

However, when a real-life problem is described as a DisCSP, it is usually over-constrained [5, 6]; it can evolve to be over-constrained because of dynamic environment [7], e.g. timetabling problems.

Most of existing algorithms give directly "no solution" in case of over-constrained. In reality, an alternative solution, even an optimal solution that is the closest to a satisfying solution as well as some explanations about no solution, are usually required by application problems.

Several approaches have been proposed for solving over-constrained problem in DisCSP. For example, Distributed Partial CSP is presented as a general model for handling an over-constrained DisCSP and its two sub-classes Distributed Maximal CSP and Distributed hierarchical CSP are proposed to deal with different type over-constraint problem solving [5, 6]; the algorithm *adopt (asynchronous distributed optimisation)* proposed recently by Modi et al [8], can be used for constraint relaxation.

Above methods can always provide optimal solutions according to different optimal functions. However, the problem solving is untraceable, and the constraint satisfaction sequence is usually predefined. They focus on optimal solution searching, but not on the explanation of inconsistence. The weaknesses of this kind of algorithm are not to be able to dynamically and truly reflect the real influence of each constraint and variable and it's not easy to reveal the direct reason of conflict which leads to no solution.

On the other hand, explication-based algorithms have the advantages for improving searching efficiency and the purposes of finding the core raisons for contradiction [9]. The typical explication-based algorithms are dynamic backtracking [10], backjump-based backtracking [11], etc. However, the problems about explications saving as well as the searching amongst explications have to be taken into account

with the increasing of problem scales and problem complexity.

Inspired from existing algorithms, this paper presents a multi-factor concept on guiding constraint relaxation. It is organized as follows: firstly, a notion of influence factor is defined. In the section 2, three types of influence factor, variable influence factor, constraint influence factor, agent influence factor and their relations are introduced. The workspace and the roles of different influence factors are presented. A case study on proposed concept during problem solving in a hierarchical agent-structure is illustrated in the 3 section. The last part gives the conclusion and some perspectives.

## 2. Influence factor for constraint relaxation

### 2.1. Influence factor context

The over-constrained problem is a kind of problem that has no solution. A DisCSP can be defined as:

- V= {V1, V2 …Vn}, a finite set of variables, whose values are taken from finite, discrete domains {D1, D2 … Dn} respectively.
- C = {C1, C2 …Cm}, a finite set of constraints on variables. Each of constraints is a restriction on the values that can be taken simultaneously by the variables.
- A = {A1, A2 …Ai} a set of agents.

After analyzing the existing methods for over-constraints problems solving, a question comes to mind: is it possible to define an or several attributes for making different influences of the same type elements (for example, constraints) on problem solving comparable in a given time or space? We can use "net income" in a certain period to evaluate different companies; we can use weight to compare the hearth state of the children in same age. The attribute as "net income", weight exist everywhere in the world. These kinds of attributes are dynamic; they are evolving with the propagation of the element they attached.

We try to define a dynamic attribute – influence factor for evaluating different relations to conflicts (no solution) during problem solving. The value of influence factor is directly associative to conflicts. Each person has weight. We can compare the weights of three girls of ten years to find the heaviest, we can define the attribute of influence factor, and we can compare the components with the same type according to the values of influence factor. For example, while a combination of values to variables do not satisfy a constraint (conflict), the value of influence factor of the constraint violated evolves. Then, in case of over-constrained, the values of influence factor of all constraints can be compared to find the constraint which leads the most conflicts.

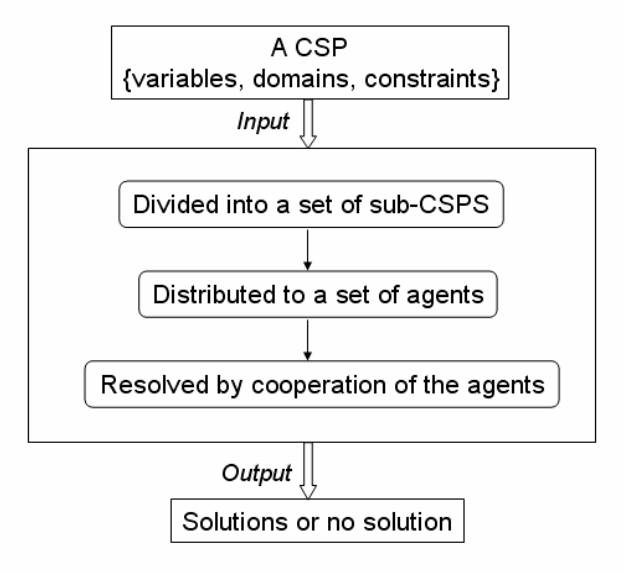


**Fig. 1: the principle structure of a DisCSP**

Considering the essential components of DisCSP as well as its mechanism (see figure 1 for the principle process structure of DisCSP), three types of influence factor are defined for DisCSP: variables influence factor presented as *Var_inf*, constraint influence factor presented as *Cons_inf*, and agent influence factor presented as *Agent_inf* respectively. The domain of each variable is considered together with variable.

The three types of influence factor are taken in to account synthetically to guide agent cooperation in normal situation and to guide the constraint relaxation in case of over-constrained.

To avoid unnecessary ambiguity, following idioms meanings are highlighted.

- A local solution is a combination of the values of agent local variables that satisfies all constraints within this agent.

- A global solution is a value combination of all variables that satisfies not only the intra-agent constraint, but also the inter-agent constraints.
- Constraint relaxation means that the released constraint will not be verified again in the following process of problem solving. In other words, after a constraint is released, solution will be realized without verifying the relaxed one.
- A constraint is violated when all values of it relative variables have been verified without finding an assignment satisfying this constraint.

## 2.2. Definition of three types of influence factor

In this part, we give the detail definitions of *Var_inf*, *Cons_inf* and *Agent_inf* and a detail description of their mechanisms as well as their role in DisCSP.

### 2.2.1 Variable influence factor

Variable influence factor (*Var_inf*) is defined as a priority degree associated to each local variable within an agent. The priority degree means, when there are conflicts among local solutions, which variable will be reassigned firstly within an agent. The *Var_inf* is used to guide next local search in each agent. At the beginning of global solution verification, every local variable's *Var_inf* is (re)set to null. Then, if the values of same variable in different local solutions are not identical, the *Var_inf* of this variable will be increased. According to the value of *Var_inf*, the corresponding agent modifies the sequence of variable's assignment in next local search. It means, if a variable contributes to the conflicts in last global verification, this variable value will be changed firstly in next local search. Bigger the *Var_inf* of a variable is, earlier the variable is reassigned.

### 2.2.2 Constraint influence factor

Constraint influence factor (*Cons_inf*) is defined as a priority degree associated to each local constraint within an agent. The *Cons_inf* highlights different constraints' influence on local solution. It is used to guide constraint relaxation within an agent. In case that an assignment of local variables can't be a local solution because of a constraint, the *Cons_inf* of this constraint will be increased. More conflicts a constraint leads to, bigger the *Cons_inf* it has. While the problem is found to be over-constrained, the constraint with the biggest *Cons_inf* should be considered a priori. The *Cons_inf* works as follows: in agent initialization, each *Cons_inf* is set to null; during agent local solution searching, the value of *Cons_inf* of each constraint is evolved along with constraint verifications. This influence factor is used for an agent to decide its constraint relaxation in case of no local solution or no global solution.

### 2.2.3 Agent influence factor

Agent influence factor (*Agent_inf*) is defined as the biggest *Cons_inf* amongst all of local releasable constraints within an agent. Each agent has its own *Agent_inf*. The value of an *Agent_inf* is modified dynamically along with the change of local agent releasable *Cons_inf* during agent local constraint verifications. So long as an agent finds no further local solution because of global interaction, the *Agent_inf* is used to decide which agent should be taken into account. The agent with the biggest *Agent_inf* will release a constraint at first.

### 2.2.4 Workspace and roles of three types of influence factor

The workspace of three types of influence factor and their roles are summarized in table 1.

| Type | Workspace | Role |
|---|---|---|
| *Var_inf* | Global & local | Guide to modify variable assignment priority |
| *Cons_inf* | local | Guide to constraint relaxation within an agent (local relaxation) |
| *Agent_inf* | Global & local | Decide which agent will release its constraints(global relaxation) |

**Table 1: influence factors' roles and influence spaces**

The values evolving of influence factor can be seen as a dynamic learning. The measure of the value of an influence factor provides the information of the influences of corresponded DisCSP component (a variable, a constraint or an agent) on inconsistence during problem solving. Three influence factors may guarantee that the biggest *Agent_inf* agent modifies its local solution firstly once a global verification is fail; the biggest *Var_inf* variable is re-assigned first its value during each agent; and the biggest *Cons_inf* constraint is always relaxed at first while over-constrained happened.

To illustrate the feasibility of three influence factors, they are embedded in a DisCSP with a simple hierarchical agent-structure. The roles of each type influence factor as well as their mechanisms are illustrated.

# 3. Case study of influence factors in DisCSP

## 3.1. A hierarchical agent-structure

As shown in the figure 2, two agent types are defined in a hierarchical structure: supervisor and executor. The relation between supervisor and its executors is hierarchical. Executors can communicate with its supervisor. There is no any communication between the executors. The communications are always vertical. Each executor handles a sub-CSP, and all executors execute in parallel.

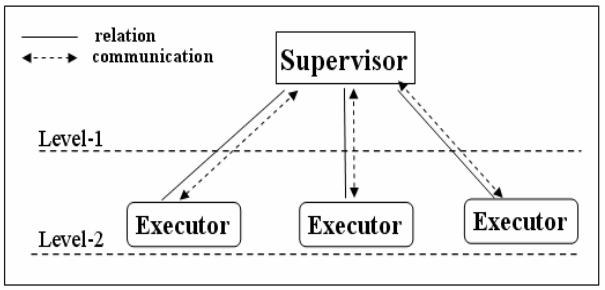


Fig. 2: a hierarchical agent-structure DisCSP

The roles of a supervisor are as follows:

- Dividing a CSP into a set of DisCSP, and then distributing them to its executors;
- Coordinating its executors' activities by sending messages in case of existing conflicts based on global interest;
- Integrating executor's local solutions to a global solution;
- Suspending, activating and stopping its lower supervisors or executor if necessary;
- Stopping system execution.

The roles of an executor are as follows:

- Assigning values to local variables, then verifying if this assignment is satisfied its local constraints.
- Releasing its releasable constraints one by one until finding a local solution or no local solution because of its interior conflicts.
- Communicating actively with its supervisor if necessary, e.g. finding a local solution, etc.

## 3.2. Influence factor evolving

According to the explications in table 1, it can be seen that *Var_inf* and *Agent_inf* are evolved in supervisor and executors, while *Cons_inf* is evolved in executors.

The process of *Var_inf* and *Agent_inf* evolving in supervisor is illustrated in figure 3.

```
Begin
Divide CSP; Distribute sub-CSPs → Executors;
nb_mes←0; violate← fault;
While( nb_mes < nb_ex) do
    Waiting for the messages from executors;
    If (Receive a message( no solution because of local inconsistence))
        Suspends all executors;
        Output no solution information;
        goto End;
    Endif;
    If( Receive a message of (no solution because of global inconsistence)
        If( exist constraints to be relaxed)
            Compare all executors' Agent_infs;
            send "release" message → the executor with the biggest
        Agent_inf;
            send "re-assign" → other executors;
            Agent_infs←0; nb_mes←0;
        Endif;
        Else
            Suspends all executors
            Output no solution information
                goto End;
        Endelse;
    Endif;
    If( receive a local solution) nb_mes←nb_mes+1;
    EndIf;
Endwhile
For each variable
    For each local solution
        If (variable exists)
            If(var_val=null) var_val←var_val of this local solution;
            Endif;
            Else
                If(var_val≠var_val of this local solution)
                    Var_inf←Var_inf+1;violate←true;
                Endif;
            Endelse;
        Endif;
    Endfor;
Endfor;
If(violate=fault) output a global solution;
Endif;
Else
    violate=fault; Agent_infs←0; nb_mes←0;
    send "continue" → all executors;
    goto While;
End
```

nb_mes: message number nb_ex: executor number nb_var:variable number var_val: variable value

**Fig. 3: process of influence factor evolving in supervisor**

*Var_inf* and *Agent_inf* evolving in supervisor are executed as following:

After dividing a CSP into a set of sub-CSPs, then distributing them to executors, the supervisor come into the step for communication as follows:

- If all executors find local solutions, supervisor verifies if there is variation between same variable values in different local solutions. If a conflict is encountered, the relative *Var_inf* is increased. After verifying all variables, if no conflict, a global solution is found; otherwise, "continue" message will be sent to all executors.
- If an executor has no local solution because of itself interior conflicts, the supervisor sends message to stop all executors. Then the supervisor outputs actual variables information, then system is stopped without finding a solution.
- If an executor has no local solution because of global conflicts, the supervisor suspends all its executors. Then the supervisor compares all *Agent_inf*. The agent with the biggest *Agent_inf* will be sent to a "release" message, which means this agent has to release its biggest *Cons_inf* constraint; while all others will be given a "re-assign" message. All executors will be activated.
- By iterating above procedures, supervisor can finally find a global solution or stop the system with all agents after having verified all possibilities without realizing a global solution

The process of *Var_inf* and *Cons_inf* evolving in executor is illustrated in figure 4, and are executed as following:

An executor begins to assign its local variables until finding a complete value combination for its local variables, and then it comes to the step of verifying local constraints. In this step, all local constraints are verified one by one. While a constraint is violates, its *Cons_inf* is increased. If this combination violates any constraint, it means this one is not a local solution, and executor has to continue assigning its variables until finding a local solution, or find no local solution.

- When a local solution is found, a finding a solution message is sent to the supervisor. This message consists of two parts: the local solution and *Agent_inf*.
- When no local solution is confirmed and any local solution was not found, this executor will release its biggest *Cons_inf* constraint, and set its *Agent_inf* to be 0, and re-assign its variables.
- When no local solution is confirmed, and a local solution was found, it means the global inconsistence leads to this no local solution, the executor sends a no local solution message to its supervisor.

Then executor comes to the next step: waiting for message.

```
Begin
Assign / re-assign:
j←0; violate ←fault;
For j<nb_var
    t[j] ←0;
Endfor;
Continue:
assignment ←{};
For every local variables vi
    If t[i]<nb_val(i)
        vi←val(i)[t[i]]; t[i] ← t[i]+1; assignment ← vi;
    Endif;
    Else
        If nb_solution ≠0
            (no solution because of global inconsistence)→ supervisor;
        Endif;
        If nb_solution =0
            Release:
            If constraints exist
                release the biggest Cons_inf constraint;  goto Assign;
            Endif;
            If no any constraint
                (no solution because of global inconsistence)→ supervisor;
                goto wait message;
            EndIf;
        Endif;
    Endelse;
Endfor;
For every local constraint Ck
    If violated by assignment
        Cons_inf (k)← Cons_inf(k)+1; violate ←true;
        for all cons_inf in this executor
            if Cons_inf (k) is the biggest Cons_inf
                Agent_inf ← Cons_inf (k);
            Endif;
        Endfor;
    Endif;
Endfor;
If(violate=fault)
    local solution ← assignment;
    (a local solution+ Agent_inf)→ supervisor;
Endif;
Else
    violate=fault;      goto continue;
Endelse;
Wait message:
While no message do wait;
If "continue " message   goto continue;
Endif;
If "re-assign " message   goto re-assign;
Endif;
If "release " message   goto release;
Endif
End
```

nb_var:variable number  nb_val(i): the number of values of variable i

**Fig. 4: the process of influence factor evolving in executor**

- When the executor receives a "continue" message, it grades its variables from the biggest *Var_inf* to the smallest one, sets all values of *Var_inf* to be 0, and then continues to assign its local variables along with the variables' order.
- When the executor receives a "release" message, it releases the constraint with the biggest *Cons_inf*, and then re-assigns its local variables from their first values.

- When a "re-assign" message is received, the executor re-assigns its local variables from their first values.

| | | Constraint influence factor | | | | | General information | | |
|---|---|---|---|---|---|---|---|---|---|
| | | B+D> F | C+D<F | E<F | B+C+A<E | A+B+C>D+F | Total verifying number | Number of global communication | Number of global solution |
| Hypo-1 | | 96 | 248 | 39.9 | 139141 | 8 | 349762 | 0 | 0 |
| Hypo-2 | | 184 | 410 | 50 | Released | 10 | 350031 | 1 | 1 |
| Hypo-3 | | 49 | 5 | 31 | Released | 49 | 3631 | 47 | 1 |
| Hypo-4 | B+D>F & B+C+A<E<br>C+D<F & E>F<br>A+B+C>D+F | 22 | 165 | 0 | Released | 34 | 2886 | 18 | 1 |
| | B+D>F & C+D<F<br>B+C+A<E & A+B+C>D+F<br>E<F | 42 | Released | 24 | Released | 16 | 3047 | 12 | 1 |

**Table 2: experimental results of 4 hypotheses**

### 3.3. Experimental results

There are 6 variables named (A, B, C, D, E, F) respectively. The domain of each variable of (A, B, C, D, E, F) is [1, 2, 3, 4, 5, 6, 7, 8, 9]. There are 5 constraints: B+D>F, A+B+C<E, C+D<F, E<F, A+B+C>D+F. It is an over-constrained problem because of the conflicts among given constraints, e.g. the conflicts between A+B+C<E, A+B+C>D+F and E<F. The goal is to find assignments of the variables that satisfy all given constraints.

Following hypotheses are proposed in order to verify whether the three influence factors can guide for an efficient constraint relaxation.

**Hypothesis 1:** all constraints are defined as hard constraints and the problem is solved in centralized way.

**Hypothesis 2:** All constraints are defined as soft constraints. The problem is solved in centralized way too.

**Hypothesis 3:** All constraints are defined as soft constraints. The constraints are distributed to agents. An agent is in charge of a constraint.

**Hypothesis 4:** All constraints are defined as soft constraints. And all these constraints are distributed to the agents randomly, an agent should be in charge of 2 constraints at least.

Hypothesis 1 aims at presenting the different influence of each constraint on problem solving, while hypothesis 2, hypothesis 3 and hypothesis 4 aim at illustrating the guiding influence on constraint relaxation of proposed influence factors. The experimental result of these hypotheses is presented in table 2.

The feasibility on the constraint relaxation with influence factors is illustrated here by a simple combination example. The results show that the relaxation of the constraint having bigger *Cons_inf* is more efficiency for finding an alternative solution. The influence of *var_inf* and *Agent_inf* can't be shown directly in the table. The experimental results indicate that the cooperation of three influence factors permits to find the most important constraints in case of different type of distributions.

## 4. Conclusion and perspectives

In this paper, three types of influence factor, *var_inf* *Cons_inf*, *Agent_inf* are proposed for guiding constraint relaxation in case of over-constrained in DisCSP. The feasibility on the constraint relaxation with these influences factors and the interactions between them is then illustrated.

Influence factor reflects the different roles played by each constraint and each variable during the problem solving. The value of influence factor is evolved dynamically and sequentially along with the problem solving. Their cooperation and interaction are useful for guiding an efficient constraint relaxation in case of over-constrained.

In the future, the proposed influence factor concept will be further applied to other contexts, and the searching efficiency within each agent will be taken into account.

*References:*


[1] E. Tsang, *Foundation Constraint Satisfaction*, Academic Press, 1993.

[2] M. Yokoo, *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent System*, Springer, London, 2001.

[3] P. J. Modi, H. Jung, M. Tambe, W. Shen, S. Kulkarni, Dynamic Distributed Resource Allocation, A Distributed Constraint Satisfaction Approach, Lecture Notes in Computer Science, 2001.

[4] W. Zhang, Z. Deng, and G. Wang, Distributed Problem Solving in Sensor Network, AAMAS'02, 2002.

[5] K. Hirayama and M. Yokoo, Distributed Partial Constraint Satisfaction Problem, Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP-97), pp. 222-236, 1997.

[6] K. Hirayama and M. Yokoo, An Approach to Over-constrained Distributed Constraint Satisfaction Problems: Distributed Hierarchical Constraint Satisfaction, Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS-2000), 2000.

[7] I. Miguel, The case for dynamic flexible constraint satisfaction, Seventh International Conference on Principles and Practice of Constraint Programming, November 26 - December 1, Paphos, Cyprus, 2001.

[8] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, An Asynchronous Complete Method For General Distributed Constraint Optimization, Proceedings of Autonomous Agents and Multi-Agent Systems , 2003.

[9] N. Jussien, The versatility of using explanations within constraint programming, École des Mines de Nantes, technical report, no. 03-04-INFO, 2003 .

[10] M. L. Ginsberg, Dynamic Backtracking, *Journal of Artificial Intelligence Research* 1 (1993) 25-46, 1993.

[11] R. Dechter and D. Frost, Backjump-based backtracking for constraint satisfaction problem, *Artificial Intelligence* 136(2002) 147-188, 2002.