Reading Assignment Homework 1

107062313 黃寶萱

- Title : Solution Reuse in Dynamic Constraint Satisfaction Problems
- Authors : Cérard Verfaillie and Thomas Schiex
- Affiliations : ONERA-CERT
- Source : Association for the Advancement of Artificial Intelligence (AAAI)
- Date of publication : 1994
- Link : https://www.aaai.org/Papers/AAAI/1994/AAAI94-302.pdf

According to the lecture in the class, we have known that the definition of the traditional CSP is that the set of variables and constraints which composed the CSP is completely known and **fixed**. However, many of CSPs are actually dynamic which means that the set of constraints to consider evolves because of the environment, the user or other agents. All the possible changes to a CSP including constraint or domain modifications, variable additions or removals can be expressed in terms of constraint additions or removals. According to this paper, the concept of dynamic CSP (DCSP) is a sequence of CSPs, where each one differs from the previous one by the **addition or removal of some constraints**. This is the difference between CSP and DCSP.

In order to solve dynamic CSPs in a more efficient and stable way, we can classify existing methods in three groups : *heuristic* methods, *local repair* methods, and *constraint recording* methods. The algorithm used by the author in this paper is called *local changes (lc)*, which is belongs to *local repair* methods. This algorithm combines the advantages of an efficient backtracking mechanism and an ability to solve any CSP, either starting from an empty assignment, or starting from any previous assignment. The key idea is that, taking binary CSP as example, let *A* be a consistent assignment of a subset *V* of the variables; let *v* be a variable which does not belong to *V*; then we can assign *v* and obtain a consistent assignment of $V \cup \{v\}$ iff there exists a value *val* of *v* such that we can assign *val* to *v*, remove all the assignments which are inconsistent with *(v, val)* and assign these unassigned variables again one after another, without modifying *v*'s assignment. This algorithm also can be improved by using any filtering or learning method, such as *forward-checking* or *nogood-recording*. According to their experiments, the efficient heuristics they used are :

- Choice of the *variable* to be assigned, unassigned or reassigned: choose the variable whose domain is the smallest one;
- Choice of the *value*: for their *lc* algorithm, choose the value which minimizes the number of unsatisfied constraints.

And in the most cases, the algorithm *lc* performs better than other algorithm with both backward-checking and forward-checking.