Title: Solution Reuse in Dynamic Constraint Satisfaction Problems

Authors: Gerard Verfaillie and Thomas Schiex

Affiliation: ONERA-CERT, Toulouse Cedex, France

Publication: *AAAI-94* (January, 1994), pp.307-312, Seattle, Washington.

url: *http://www.aaai.org/Papers/AAAI/1994/AAAI94-302.pdf*

Point out what concepts are different in contrast to the traditional CSP:

Completely known and fixed sets of variable is a strong limitation in traditional CSP especially when handling real situation problem. If users wonder interactive with framework, environment need to execute scheduling applications or other agents in the framework is distributed system, traditional CSP is hard to achieve them.

On the contrary, dynamic CSP is sequence of CSPs, where each one differs from the previous one by addition or removal constraints, including constraints, domain modification and variables. In this context, computing a new solution from scratch after each problem change is possible.

What methods and heuristics are used in the solving such a new CSP in order to find a solution fast:

To solve a sequence of CSPs, it can always solve each one from scratch, which is similar to solve traditional CSP again and again. However, this naive method couldn't remember anything from previous reasoning and solution, which has two significant drawback. One is inefficiency, which may be unacceptable in the framework of real time applications (e.g. scheduling) because of the time limitation. The other one is instability of successive solution if some computation is based on previous solution.

The resulting algorithm is called as *local change*. It's related to *backjumping*, *intelligent backtracking* and *dynamic backtracking* which can avoid useless backtracking because of superfluous choices of which aren't involving in current conflicts. In this way, it can improve the time performance. Moreover, *local change* is related to *heuristic repair* algorithms, which allows the search to be started from previous work. The first three algorithms aren't designed for dynamic CSP, and the last one use usual basic backtracking mechanism

which is inefficient. *Local change* combines advantage of efficient backtracking mechanism with the ability to start the search from previous. Furthermore, *local change* algorithm can be improved by using other filtering or learning methods such as *forward-checking* and *no-good-recording*. Combine local change with ones, it can apply from any assigned, unassigned or reassigned variables.