Assignment #3 - DCSP

Martin Matyášek

January 3, 2015

1 Problem description

The *n-queens* problem is formalized as a *DCSP* $\langle X, D, C, A\rangle$ as follows:

variables $X = \{X_1, \dots, X_n\}$ where X_i represents queen at row $i \in \{1, \dots, n\}$.

domains $D = \{D_1, \dots, D_n\}$ where each $D_i = \{1, \dots, n\}$ corresponds to X_i and represents possible column positions of i-th queen (i.e. queen in the i-th row).

constraints $C = \{C_{ij} : 0 < |X_i - X_j| \neq |i - j| \mid i \neq j;\ i, j = 1, \dots, n\}$.
Each of the C_{ij} constraints restricts positioning of distinct queens i and j so that they do not threaten each other neither vertically nor diagonally[1].

agents $A = \{A_1, \dots, A_n\}$ where each agent A_i is responsible for queen X_i.

2 Algorithm description

The implemented algorithm is an *Asynchronous Bactkracking* algorithm (*ABT*) for *DCSP*. More precisely, it is *ABT-opt* (asynchronous backtracking adapted for optimization) as it is described in [1].

The algorithm is rather direct implementation and no significant changes have been made to it, althbough the *AddLink* messages could have been left out due to this specific problem. The message specification is as follows:

Ok?(v) which higher-priority agent i sends to agent j to ask if $X_i = v$ is ok.

NoGood(nogood) which lower-priority agent j sends to k to inform that he cannot set his value due to *nogood* involving k. The *nogood* is of the form $v : [conditions] : [tag] : cost : exact$ where $X_k = v$ is no good under [*conditions*] with the *exact cost* on which [*tag*] agents participate.

[1]More precisely, there are actually two relations $C^v_{ij} : 0 < |X_i - X_j|$ (i.e. $X_i \neq X_j$) and $C^d_{ij} : |X_i - X_j| \neq |i - j|$ for each C_{ij}. Nonetheless, one can combine these as $C_{ij} \equiv C^v_{ij} \wedge C^d_{ij}$ since both consider the same pair of distinct variables. So alternatively one can define the constraints complactly as $C = \{C_{ij} : |X_i - X_j| \cdot |\,|X_i - X_j| - |i - j|\,| > 0 \mid i \neq j;\ i, j = 1, \dots, n\}$.

AddLink which lower-priority agent k sends to higher priority agent i to ask for constraint link addition.

Stop(solution) which the highest priority agent broadcasts to inform that the problem instance has *solution*. The solution is either empty, in which case the problem instance has no solution, or $*$ indicating that there is a solution and other agents should report their current assignments.

The very last issues to discuss are *priority ordering* and *termination detection*. Priorities are determined at the very beginning of the distributed search. Each agent knows priorities of all agents in advance as opposed to the agents themselves (uses only a *local view*). The ordering itself is an inverse of agent identifiers (i.e. the highest priority agent has priority $n-1$, the lowest priority agent has priority 0).

The implementation of guaranteed termination detection is rather simple since it is an inherent feature of the *ABT-opt* algorithm. The algorithm terminates if the highest priority agent generates a *nogood*. There are two possible outcomes, the *cost* of such *nogood* is non-zero and the problem instance has no solution; or the *cost* is zero and the current assignment is the solution. In either way such a *cost* must be *exact*, meaning that all lower-priority agents must have sent an *exact nogood* for certain value and therefore the termination is guaranteed since, by induction, all agents have agreed on the solution.

References

[1] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.