Catholic University of Louvain

Project 2 : Propagation

LINGI2365 - Constraint Programming

Auteurs :
Vanwelde Romain (3143-10-00)
Crochelet Martin (2236-10-00)

Superviseurs :
Pr. Yves Deville
François Aubry

6 mars 2014

Table des matières

1 Questions **1**
1.1 Q 2.1 1

1 Questions

1.1 Q. 2.1

With the notations :
— $e = \#C$ where C is the set of constraints that set up the problem and e, the number of constraints.
— $d = \max_{1 \leq i \leq n}(\#D(x))$ where $D(x)$ is the domain of the variable x and thus d, the maximum size of the domains.
— r that is defined as the maximum arity of a constraint for the CSP.
And the supposition that each variable is involved in at least one constraint (and that the constraints might have an arity superior to two - otherwise we would reduce to AC3).

Analysing the algorithm shows us that DC3 keeps a queue that contains the set of constraints for whom the domain consistency is no longer guaranteed. In the worst case, we can easily imagine that every constraint of the problem will find it's place in that queue. Knowing that the number of constraints for a CSP is defined as e, we can deduce that the worst-case space complexity involved with the DC3 algorithm is $O(e)$ (Note that this is independent of for data-structures used by the propagate method)

Concerning the time complexity, we will split our analysis into two different parts : first, we will analyse the time-complexity of the standard CDC algorithm since the DC3 one in a basic instance of CDC. Lastly, we will tackle the analysis of the DC3-specific propagate method.

For a non binary CSP, we begin by observing that a constraint can be put at most $r \cdot d$ times in the queue (maximum arity of a constraint time the maximum size of a domain) and that at most, since we iterate over that queue to call propagate, the propagate method will be executed $e \cdot r \cdot d$ times (the number of constraints times the maximum number of times a constraint can be put into the queue).

Furthermore, we can remark that the DC3-specific propagate method iterates over each variables of a constraint and over the domain of that constraint $O(r \cdot d)$ This iteration executes a code verifies for each value if it is still consistent with the constraint and construct the delta set. This implies that the algorithm has to iterates over all other variables of the constraint to check if a possible assignment exists for the current choice of assignment : the execute runs in $O(r \cdot d^{r-1})$ and is executed at most r times. The time-complexity of the DC3-specific propagate method is thus $O(r^2 \cdot d^r)$.

We can then deduce that the overall complexity of DC3 is the product of the two complexities and equals $O(e \cdot r \cdot d \cdot r^2 \cdot d^r) = O(e \cdot r^3 \cdot d^{r+1})$

1.2 Q. 2.2

The algorithm would not be correct since it would never terminate. Indeed, moving the line just after line 16 would remove the constraint from the queue and then re add it into the queue since the enqueueCDC would contain the constraint itself. The algorithm would iterate forever over the constraints and never leave the loop.

1.3 Q. 2.3

1.3.1 a

This directly derives from the definition of domain consistent : suppose that a value that belongs to X generates a value $X' = X - a$