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 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 $X' = X - a$ that does not belong to the domain of Y then the domain would not be consistent since the definition of domain consistent is that for all values that belong to the domain of the variable, there exists at least one value in the domains of the other variables of the constraint that satisfies the constraint.

1.3.2 b)

The bound consistency is a weaker form of domain consistency in the sense that it only considers the bounds of the domain. This means that we actually have two conditions : one on the minima of the variables and one on the maxima :

— on the minima : $\min_{x \in D(X)}(x) - a = \min_{y \in D(Y)}(y)$

— on the maxima : $\max_{x \in D(X)}(x) - a = \max_{y \in D(Y)}(y)$

1.3.3 c)

Indeed, if the constraint is domain consistent, this means that the bound are also consistent since they belong to the domain and that all the values in the domain satisfies the constraint. Bound consistency is a weaker form of domain consistency.

1.3.4 d)

No the implication if false in this sense. Indeed for a constraint such as $x^2 = y$, and the domains $[0, 1, 2, 3, 4, 5, 6, 7, 8]$ for both variables, bound consistency would return the domains : $\{x := [0, 1, 2], y := [0, 1, 2, 3, 4]\}$ while domain consistency would give the domains : $\{x := [0, 1, 2], y := [0, 1, 4]\}$.

2 Problems

2.1 Q. 3.1

2.1.1 1), 2)

Intuitively, the domain consistency is the set of values of the domain that have a support (meaning a value that satisfies the constraint) in the domain of the other variables of the constraint. In this case, the domain consistency and bound consistency are equivalent under the assumption that the domain for X and Y are ordonable values. Indeed, since the constraint is linear, the condition on the bounds of the domains is sufficient to provoke a domain consistency. We have then the conditions :

— $\min_{x \in D(X)}(x) \geq \min_{y \in D(Y)}(y) + a$

— $\max_{x \in D(X)}(x) \geq \max_{y \in D(Y)}(y) + a$

Note that the $\geq$ plays an important role as well : in the previous case of constraint (same constraint but with equality), the domain consistency was different from the bound consistency. Indeed, the equality forces the values in the domain of Y to follow a linear function of the values of X while in the case of the $\geq$, the values of Y "just" have to belong to a subset of the plane that is defined by the minima and maxima of the domain of X.

2.1.2 3)

Since we are using bound consistency (equivalent to domain consistency in this case), we can implement the CDC algorithm by using two different queues of propagation : one for the maxima and one for the minima). This is a simple optimization that allows to reconsider only the constraints that have been "touched" by the modification of the maxima *or* minima of a domain. The two lines in questions defines those queues and initialises them by putting all the constraints of the CSP into the queues. This corresponds to the initCDC method of the algorithm 5.1 of the book.

2.1.3 4)

This implementation begins by updating the min and max of the domains of the variables and, if one of the domain boils down to zero, the propagation returns a failure. Otherwise, the propagation will test if the current assignment satisfies the constraint and, if it's the case, it returns a success. In the last case, the implementation will return a suspend meaning that the current assignment does not satisfies the constraint but that there are still values in the domain that have to be tried.

Furthermore, the propagate method has important side effects : it reduces the domain of the variables in order to accelerate the search by removing bounds of the domains. The specifications demand the new domain to be a subset of the previous one and that it still contains all the possible solutions for the CSP. This is easy to see considering the fact that the propagate method only removes values (via update$Max|Min$) that are proved to not satisfy the constraint.