

# Project sheet 3: Look-ahead and Variable ordering (30 points)

Prof. Dr. Bernhard Nebel, Dr. Stefan Wöflf,  
Dr. Julien Hué, and Matthias Westphal

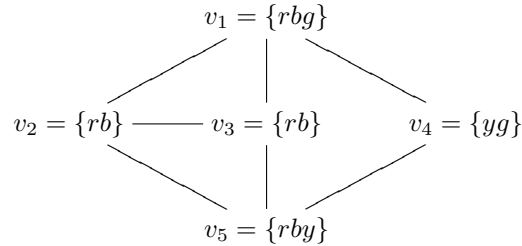
June 18, 2012

The code must be handed in by June, 29th in the git repository as usual.

## 1 Implementing Maintaining Arc Consistency (12 points)

The next step of our project is the use of Arc-consistency as look-ahead technique. Compared with the standard backtracking algorithm, look-ahead requires to keep track of the changes made at every step.

**Example 1.** Let  $N = \langle V, D, C \rangle$  be a constraint network where  $V = \{v_1, \dots, v_5\}$  and the following graph represent the constraints. If two variables are linked in the following graph, then they cannot be instantiated with the same color.



Lets consider that first  $v_1$  has been instantiated to  $g$ . This instantiation triggers  $v_4 \mapsto y$  and thus  $v_5 \not\mapsto y$ . Then,  $v_2 \mapsto r$  is decided and thus  $v_3 \mapsto b$  and  $v_5 \not\mapsto r$  are propagated, which in turn trigger  $v_5 \not\mapsto b$ . The domain of  $v_5$  is empty and a backtrack is happening.

It is thus necessary to maintain a stack of the changes that have been made. In the latter case, it may looks like this:

Level	Decision	Propagation
1	$v_4 \mapsto g$	$v_4 \mapsto y, v_5 \not\mapsto y$
2	$v_2 \mapsto r$	$v_3 \mapsto b, v_5 \not\mapsto r, v_5 \not\mapsto b$

When a backtrack occurs, the deepest level of decision and propagations is reversed.

For maintaining arc consistency, it is necessary to run the arc consistency algorithm every time a variable is assigned (and/or every time a value is discarded depending on whether you chose Enumeration or Binary Branching Procedure).

## 2 Variable Selection Heuristics

In order to improve the efficiency of our CSP implementation, we now implement a heuristic for variable selection. We will explore two static ordering strategies: Max-cardinality and Min-width. Since they are *static orderings*, they should be computed before the backtracking algorithm is started.

### 2.1 Max-Cardinality (6 points)

This heuristic is simple to implement:

1. For the first variable, pick an arbitrary variable.
2. For the next ones, repeatedly add a variable such that the number of constraints whose scope is a subset of the set of added variables is maximal. Break ties arbitrarily.

### 2.2 Min-Width (6 points)

The min-width heuristic aims at minimizing the width of the tree representing the search space. The rough idea is thus summed up by:

1. Select a variable ordering such that the resulting ordered constraint graph has minimal width among all choices.

In practice, the following greedy algorithm computes a min-width ordering.

MIN-WIDTH

Input: A graph  $G=(V,E)$  ;  $V=\{v_1, \dots, v_n\}$   
Output: A min-width ordering of the nodes  $d=(v_1, \dots, v_n)$

1. for  $j=n$  to 1 (with a step of -1)
2.  $r :=$  a node in  $G$  with the smallest degree.
3. Put  $r$  in position  $j$  and  $G := G - r$ .  
(delete  $r$  from  $V$  and all adjacent edges)
4. endfor

### 2.3 Command line

If the name of your executable file is *solver*, the instance is named *instance.xml* and the heuristic to use is *algo*, the user may be able to select which heuristic he wants to use thanks to the following command line:

```
./solver --heuristic=algo instance.xml
```

Therefore, the following command lines would solve the instance with the correct associated algorithm:

```
./solver --heuristic=unspecified instance.xml  
./solver --heuristic=minwidth instance.xml  
./solver --heuristic=maxcardinality instance.xml
```

Unspecified, in this case refers to an arbitrary ordering of variables. For this you can simply initialize the ordering of variables as some list (perhaps lexicographic, or just the order in which you process variables).

## 2.4 Comparing the heuristics (6 points)

Add into your repository the file *heuristics.txt* (dont forget *git add heuristics.txt*). This file must contain a table with the running times of the instances we provided according to variable selection heuristics implemented (max-cardinality, min-width, and unspecified). The solver should run in combination with Maintaining Arc Consistency. We suggest to put a time limit of 5 minutes to your tests thanks to the *ulimit* command.

After the table, write a short text (400-600 characters) commenting and comparing the results in the table.