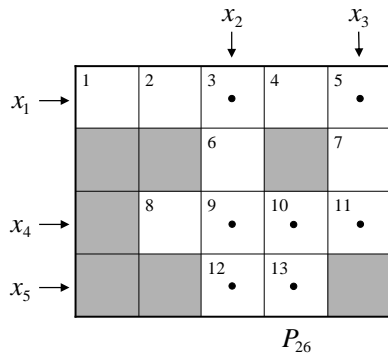


# ICS 275

## Homework 4

### Solutions

#### Question 1 - Ex. 4, Ch. 5

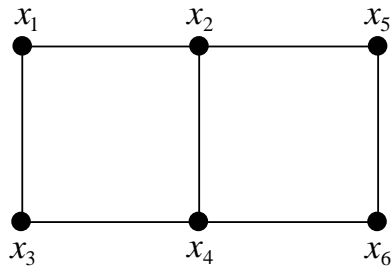


Variables:  $x_1, x_2, x_3, x_4, x_5, x_6$

Domains:  $D_1 = \{hoses, laser, sheet, snail, steer\}$   
 $D_2 = \{also, earn, hike, iron, same\}$   
 $D_3 = \{eat, let, run, sun, ten, yes\}$   
 $D_5 = \{be, it, no, us\}$   
 $D_6 = D_5$   
 $D_4 = D_2$

Constraints:  $R_{12}, R_{13}, R_{24}, R_{25}, R_{34}, R_{46}, R_{56}$

$d = x_1, x_2, x_5, x_6, x_4, x_3$



a) forward checking.

$i = 1$

$x_1 = hoses$   $D_1 = \{laser, sheet, snail, steer\}$   
 $D_2 = \{same\}$   
 $D_3 = \{sun\}$   
 $D_4 = \{also, earn, hike, iron, same\}$   
 $D_5 = \{he, it, no, us\}$   
 $D_6 = \{be, it, no, us\}$

$i = 2$

$x_2 = same$   $D_2 = \{\}$   
 $D_3 = \{sun\}$   
 $D_4 = \{\} \rightarrow$  dead-end, reset domains and backtrack  $i - 1$

$i = 1$

$x_1 = laser$   $D_1 = \{sheet, snail, steer\}$   
 $D_2 = \{same\}$   
 $D_3 = \{sun\}$

$$\begin{aligned}
D_4 &= \{also, earn, hike, iron, same\} \\
D_5 &= \{he, it, no, us\} \\
D_6 &= \{be, it, no, us\}
\end{aligned}$$

$i = 2$

$$\begin{aligned}
x_2 = same \quad D_2 &= \{\} \\
D_5 &= \{\} \rightarrow \text{dead end, set domains, backtrack } i - 1
\end{aligned}$$

b) Dynamic variable ordering *DVFC*.

$i = 1$

$$\begin{aligned}
x_2 = hoses \quad D_1 &= \{laser, sheet, snail, steer\} \\
D_2 &= \{same\} \\
D_5 &= \{he, it, no, us\} \\
D_6 &= \{be, it, no, us\} \\
D_4 &= \{also, earn, hike, iron, same\} \\
D_3 &= \{sun\}
\end{aligned}$$

next variable  $\min(D_i) \rightarrow x_2$

$i = 2$

$$\begin{aligned}
x_2 = same \quad D_2 &= \{\} \\
D_5 &= \{\} \text{ dead-end, reset domains, backtrack } i - 1
\end{aligned}$$

$i = 2$

$$\begin{aligned}
x_1 = laser \quad D_1 &= \{sheet, snail, steer\} \\
D_2 &= \{same\} \\
D_5 &= \{he, it, no, us\} \\
D_6 &= \{be, it, no, us\} \\
D_4 &= \{also, earn, hike, iron, same\} \\
D_3 &= \{sun\}
\end{aligned}$$

next variable  $\min(D_i) \rightarrow x_2$

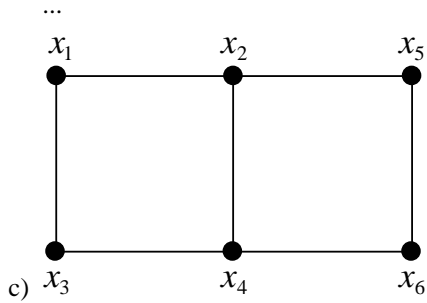
$i = 2$

$$\begin{aligned}
x_2 = same \quad D_2 &= \{\} \\
D_5 &= \{\} \rightarrow \text{dead-end, reset domains, backtrack } i - 1
\end{aligned}$$

$i = 1$

$x_1 = \text{sheet}$      $D_1 = \{\text{snail, steer}\}$   
 $D_2 = \{\text{earn}\}$   
 $D_5 = \{\text{he, it, no, us}\}$   
 $D_6 = \{\text{be, it, no, us}\}$   
 $D_4 = \{\text{also, earn, hike, iron, same}\}$   
 $D_3 = \{\text{ten}\}$

next variable  $\min(D_i) \rightarrow x_2$

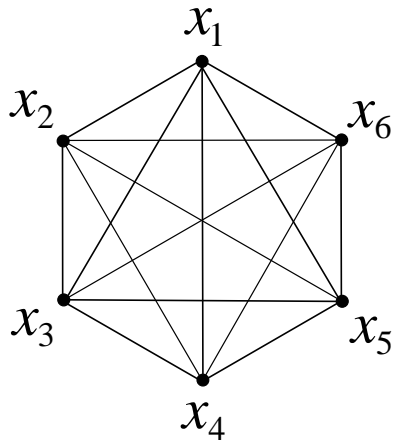


Since we have a binary constraint network, the directional arc consistency for a subproblem is bounded to  $O((w^*)^2 K^2)$ . We have  $K$  subproblems and  $u$  variables. Therefore, the complexity bound would be  $O(u(w^*)^2 K^3)$ .

### Question 2 - Ex. 7, Ch. 5

Variables:  $\{x_1, x_2, x_3, x_4, x_5, x_6\}$

Domains:  $\{1, 2, 3, 4, 5, 6\}$



#### a) Forward checking

Lets assume that the variable will be instantiated in the following order:  $(x_1, x_2, x_3, x_4, x_5, x_6)$

$i = 1$

$x_1 = 1$      $D_1 = \{2, 3, 4, 5, 6\}$   
 $D_2 = \{3, 4, 5, 6\}$

$$\begin{aligned}
D_3 &= \{2, 4, 5, 6\} \\
D_4 &= \{2, 3, 5, 6\} \\
D_5 &= \{2, 3, 4, 6\} \\
D_6 &= \{2, 3, 4, 5\}
\end{aligned}$$

$i = 2$

$$\begin{aligned}
x_2 = 3 \quad D_2 &= \{4, 5, 6\} \\
D_3 &= \{5, 6\} \\
D_4 &= \{2, 6\} \\
D_5 &= \{2, 4\} \\
D_6 &= \{2, 4, 5\}
\end{aligned}$$

$i = 3$

$$\begin{aligned}
x_3 = 5 \quad D_3 &= \{6\} \\
D_4 &= \{2\} \\
D_5 &= \{2, 4\} \\
D_6 &= \{2, 4\}
\end{aligned}$$

$i = 4$

$$\begin{aligned}
x_4 = 2 \quad D_4 &= \{\} \\
D_5 &= \{4\} \\
D_6 &= \{\}
\end{aligned}$$

$i = 3$

$$\begin{aligned}
x_3 = 6 \quad D_3 &= \{\} \\
D_4 &= \{2\} \\
D_6 &= \{2, 4, 5\}
\end{aligned}$$

$i = 4$

$$\begin{aligned}
x_4 = 2 \quad D_4 &= \{\} \\
D_5 &= \{\}
\end{aligned}$$

**b) Dynamic variable ordering DVFC**

$i = 1$

$$\begin{aligned}
x_1 = 1 \quad D_1 &= \{2, 3, 4, 5, 6\} \\
D_2 &= \{3, 4, 5, 6\} \\
D_3 &= \{2, 4, 5, 6\} \\
D_4 &= \{2, 3, 5, 6\} \\
D_5 &= \{2, 3, 4, 6\} \\
D_6 &= \{2, 3, 4, 5\}
\end{aligned}$$

$i = 2$

$$\begin{aligned}x_2 = 3 \quad D_2 &= \{4, 5, 6\} \\ D_3 &= \{5, 6\} \\ D_4 &= \{2, 6\} \\ D_5 &= \{2, 4\} \\ D_6 &= \{2, 4, 5\}\end{aligned}$$

$i = 3$

$$\begin{aligned}x_3 = 5 \quad D_3 &= \{6\} \\ D_4 &= \{2\} \\ D_5 &= \{2, 4\} \\ D_6 &= \{2, 4\}\end{aligned}$$

$i = 4$

$$\begin{aligned}x_4 = 2 \quad D_4 &= \{\} \\ D_5 &= \{4\} \\ D_6 &= \{\}\end{aligned}$$

$i = 3$

$$\begin{aligned}x_3 = 6 \quad D_3 &= \{\} \\ D_4 &= \{2\} \\ D_5 &= \{2\} \\ D_6 &= \{2, 4, 5\}\end{aligned}$$

$i = 4$

$$\begin{aligned}x_4 = 2 \quad D_4 &= \{\} \\ D_5 &= \{\}\end{aligned}$$

**c) Arc-consistency look-ahead**

$i = 1$

$$\begin{aligned}x_1 = 1 \quad D_1 &= \{2, 3, 4, 5, 6\} \\ D_2 &= \{3, 4, 5, 6\} \\ D_3 &= \{2, 4, 5, 6\} \\ D_4 &= \{2, 3, 5, 6\} \\ D_5 &= \{2, 3, 4, 6\} \\ D_6 &= \{2, 3, 4, 5\}\end{aligned}$$

$i = 2$

$$\begin{aligned}x_2 = 3 \quad D_2 &= \{4, 5, 6\} \\ D_3 &= \{5, 6\} \\ D_4 &= \{\}\end{aligned}$$

### Question 3 - Ex. 6, Ch. 5

Node expansion overhead analysis:

#### Backtracking

The overhead due to node expansion during naive backtracking can be bounded to  $O(e K \log t)$ . Since looking up into a table constraints takes constant time:  $O(\log t)$ , and for each node  $x_i$  along the search path there are at most  $e_i$  constraints to be checked, the complexity bound is correct. The  $K$  parameter denotes the domain size bound.

#### Backmarking

Assuming that table update and lookup take constant time  $O(1)$ , let  $x_i$  be a tentative variable. There are at most  $i_e$  constraints the variable is involved in, so the complexity of node expansion can be bounded in the worst-case at  $O(e_i K \log(t))$  for that node. Still, there is a best-case when the overhead can be bounded to a constant time  $O(1)$  (e.g. the  $M_v$  and  $low$  tables can determine this case). In general, the total overhead along a search path is bounded to  $O(eK \log(t))$ .

#### Forward-checking

A constraint check can take at most  $O(\log(t))$  time. For a tentative variable  $x_i$  there are at most  $O(e_j K \log t)$  constraint checks to all  $x_j$  future variables and since we're testing all  $K$  values of  $x_i$  (in the worst-case) the overhead is bounded to  $O(eK^2 \log(t))$  along the search path.

#### Partial look-ahead

Now we have to enforce directional arc-consistency among all future variables relative to a tentative variable  $x_i$ . This task can take at most  $O(e_j K^2 \log(t))$ . By summing along the path and since for a node we might test all its  $K$  values, the total overhead for node expansion is bounded to  $O(eK^3 \log(t))$ .

#### Arc-consistency look-ahead

Now we have to enforce full arc-consistency among all future variables relative to a tentative variable  $x_i$ . If we use an AC-3 technique then this process can take at most  $O(e_j K^3 \log(t))$ . By summing along the path and since for a node we might test all its  $K$  values, the total overhead for node expansion is bounded to  $O(eK^4 \log(t))$ .

### Question 4 - Ex. 9, Ch. 5

- a) If we have a graph having induced width 2 then we know that it can be solved backtrack-free by applying DPC. Let  $c$  be the set of nodes having the following property: by removing those nodes (e.g. instantiate those variables) we get a graph having induced width 2. This, the level of search when DPC gets to be backtrack-free is  $w = m - c$ , where  $m$  denotes the total number of variables and  $c$  denotes the width-2 cutset

- b) Complexity

We know that DPC is able to solve graph with induced width 2 in  $O(mk^3)$ . Let  $c$  be the width-2 cutset of the graph. The variables from  $c$  can be instantiated in a total of  $K^c$  different ways, where  $K$  denotes the bound for the domain size. Thus we have the complexity for our problem:

$$O((m - c)K^3 \cdot K^c) = O((m - c)K^{c+3})$$

## Question 7 - Ex. 10, Ch. 5

The procedure is a general search algorithm that creates resolvents whose size is bounded by  $K$ .

```
DPLL ( $\Psi, K$ )
{
  UNIT-PROPAGATE( $\Psi$ )
  if empty clause then return FALSE;
  else if all variables are assigned then return TRUE;
  else
    while  $\exists x \in Var(\Psi)$  such that degree( $x$ ) is  $\leq K$ 
    {
      Use resolution over  $x$  to generate  $\epsilon$  new clauses
      if empty clause then return FALSE
      else  $\Psi = \Psi \cup \epsilon$ 
      remove  $x$ 
    }
  else let  $x$  be some unassigned variable
  return DPLL( $\Psi \wedge x, K$ )  $\vee$  DPLL( $\Psi \wedge \neg x, K$ )
}
```

## Question 6 (Langford Encodings)

- a) (Two CSP Encodings) Here I'll provide CSP encodings of the  $L(n,2)$  problem. It is easy to generalize it to the  $L(n,k)$  problem

**Encoding 1** Here each number is a variable and so there are  $2n$  variables. The domain of each variable is the position it can take. Therefore the domain of each variable is  $\{1, \dots, 2n\}$ . There are two types of constraints. The first constraint type involves variables  $(X_i, X_{n+i})$  for  $1 \leq i \leq n$  and models the requirement that the two numbers should be placed  $i$  positions apart. For example the constraint between variables  $X_1$  and  $X_{n+1}$  models the requirement that only one color should be placed between the two variables. The allowed tuples for this constraints are  $\{(1, 3), \dots, (2n - 2, 2n)\}$ . The second type of constraints model the requirement that no two numbers should be assigned the same position. These are binary not-equal constraints between all pairs of variables or a single all-different constraint between all variables.

**Encoding 2 (Redundant modeling)** Here, we have two sets of variables. The first set of variables models the numbers  $X_i$   $1 \leq i \leq 2n$  and the second set of variables models the positions  $Y_j$   $1 \leq j \leq 2n$ . The domain of each variable is  $\{1, \dots, 2n\}$ . We have all constraints as in encoding 1 for the first set of variables  $X$ . To maintain consistency between variables that model the numbers and variables that model the positions, we have constraints of the form  $X_i = i$  iff  $Y_j = j$  for  $1 \leq i, j \leq 2n$ . Optionally, we can have additional constraints which model the fact that no two  $Y$  variables take the same value (either as a global all-different constraint or as binary constraints between all pairs of  $Y$  variables).

- b) (Two SAT Encodings)

**Encoding 1** Here, we will encode CSP Encoding 1 as a satisfiability problem. Here we have a boolean variable  $X_{ik}$  which is true iff a CSP variable  $X_i$  is assigned a value  $k$ . We have clauses which ensure that each CSP variable is given a value: for each  $i$ ,  $X_{i1} \vee \dots \vee X_{i2n}$  where  $2n$  is the number of values that the CSP variable  $X_i$  has. We have clauses that ensure that each variable takes no more than one value: for each  $i, a, b$  with  $a \neq b$ ,  $\neg X_{ia} \vee \neg X_{ib}$ . Finally, we have binary clauses that rule out the no-goods. For example, if  $X_1 = 2$  and  $X_3 = 1$  is not allowed then we have the clause  $\neg X_{12} \vee \neg X_{31}$ .

**Encoding 2** Here, we will encode CSP Encoding 2 as a satisfiability problem. Note that the CSP variables that model the positions are already mapped as boolean variables in the SAT Encoding1 described above. This is because we can use  $X_{ij}$  to represent both the  $j$ th value of the CSP variable  $X_i$  and the  $i$ th value for the CSP variable  $Y_j$ . Therefore, the constraints that maintain consistency between the CSP variables that model the numbers and the CSP variables that model the positions as SAT clauses are already modeled in SAT Encoding1. However, we need to add additional clauses which ensure that each CSP variable  $Y_j$  takes just one value and that no two  $Y$  variables take the same value.