

## The data files for problems and the report

For this assignment (latin square completion problem), the components of solution schemes can be organized along two dimensions. Along one dimension (x-axis), we have BT, FC, MAC, etc. Along another dimension (y-axis), we have the variable ordering heuristics like SDF, max-static-degree, max-dynamic-degree, etc.

problem file	solution scheme-1 MAC + SDF		solution scheme-2 FC + DomDeg		solution scheme-3 FC + Brelaz	
	Number of nodes	Number of backtracks				
d-10-01						
d-10-06						
d-10-07						
d-10-08						
d-10-09						
d-15-01						

The data files are provided as text files. For all problems, it is possible to complete a latin square. For each solution scheme, you should mention the number of nodes and the number of backtracks. For example, 4-queens problem and FC required eight nodes and two backtracks. On the other hand, 4-queens problem and MAC required three nodes and one backtrack.

[It is shown in the webpage: <https://ktiml.mff.cuni.cz/~bartak/constraints/propagation.html>]

## Explanation of BT, FC, and MAC

You should visit the following webpages:

- (1) <https://ktiml.mff.cuni.cz/~bartak/constraints/ordering.html>
- (2) <https://ktiml.mff.cuni.cz/~bartak/constraints/propagation.html>
- (3) <https://ktiml.mff.cuni.cz/~bartak/constraints/consistent.html>

From the first webpage, it is necessary to understand the principles behind the variable ordering and the value ordering heuristics. The second and the third webpages explain BT, FC, and MAC algorithm. For implementing MAC, the pseudocode for Algorithm AC-3 (described in third webpage) may be utilized.

Next, an informal description of the search algorithms is provided.

DFS, BT, FC, MAC are search algorithms which can be used for solving a CSP. In a CSP formulation, we have the variables, domains of the variables (or values), and constraints. The constraints can be viewed as relations. Solving a CSP may be viewed as extending a partially constructed solution to a complete solution. During search, we are going to have a set of assigned variables (which forms the partial solution), and a set of unassigned variables.

- **What is the difference between DFS and BT?**

In depth-first search (DFS), we continue to explore (and assign values to one of the unassigned variables) until we reach a leaf node (when there are no more any unassigned variable). At a leaf node, we make the decision if it is a goal node or solution node.

In backtracking search (BT), we may cut-off the search and do not move forward (or downward) if we recognize the partial solution already violates one (or more) constraint(s). We need to satisfy all the constraints to obtain a solution. In BT, at any search node, we can use the set of assigned variables and test the constraints to make the decision if we can backtrack.

To sum up, we learn that reasoning activity can be used to cut down search activity. BT uses more reasoning and less search in comparison with DFS. We are going to find out that FC uses more reasoning than BT, and MAC uses more reasoning than FC.

- **What is the difference between BT and FC?**

In backtracking search (BT), we are using only the set of assigned variables (the partially constructed solution) to make the backtracking decision. In forward checking (FC), we are also going to use some of unassigned variables who are involved in at least one constraint with the assigned variables. It is easy to find them using the constraint graph because these are the neighbor nodes of the assigned variable nodes.

In FC, at a certain search node, assume that we have just made an assignment of a certain value  $d_1$  to a certain variable  $X_i$ . Also, assume  $X_k$  is an unassigned variable which is involved in a constraint  $C(X_i, X_k)$  with  $X_i$ . We can apply a filtering algorithm to remove or filter out those values from the domain of  $X_k$  which are inconsistent with the assignment  $X_i = d_1$ .

For example,  $X_1$  and  $X_2$  are two variables, each with domains  $\{1, 2, 3, 4, 5\}$ , and there is a constraint  $X_1 + X_2 > 7$ . Now if an assignment  $X_2 = 4$  is made, we can filter out 1, 2, and 3 from the domain of  $X_1$ .

When an assignment  $X_2 = 4$  is made, the domain of  $X_2$  has become  $\{4\}$ , and we test all values remaining in the domain of  $X_1$  in order to remove those values which are inconsistent. We find that the following tuples  $\langle 1, 4 \rangle$ ,  $\langle 2, 4 \rangle$ , and  $\langle 3, 4 \rangle$  do not satisfy the constraint  $X_1 + X_2 > 7$ .

We can backtrack if the domain of an unassigned variable becomes empty (after running the filtering algorithm). For example, assume  $X_1$  has a domain of  $\{3, 4\}$  and  $X_2$  has been assigned 5. There are two constraints  $C_1$  and  $C_2$ .

C1:  $X1 + X2 < 7$  and C2:  $X1 + X2 > 3$

In this case, after running the filtering algorithm, the domain of  $X1$  becomes empty and the decision to backtrack is made.

To sum up, we become familiar with consistency checking and filtering algorithms. They are going to be used for reasoning activity in FC. Also, they allow us to make the backtracking decision earlier than BT.

- **Explaining MAC with the 4-queens problem**

In solving a CSP, we keep partial solutions and extend them to construct a complete solution. Also, solving a CSP may be viewed as solving sub-problems or smaller problems, and then joining the solutions together to obtain the complete solution.

Before explaining MAC, we need to learn the definitions of node consistency and arc consistency. Also, we need to consider the constraint graph.

If we consider that the sub-problems are limited to a single node, then we need to ensure unary constraints related to a particular node is satisfied. In this case, the filtering algorithm removes a value from the domain of a variable, if the value is not consistent with the unary constraint. The node representing a variable  $V$  in constraint graph is node consistent if for every value  $x$  in the current domain of  $V$ , each unary constraint on  $V$  is satisfied.

If there is a binary constraint  $C(V_i, V_j)$ , we have an edge  $(V_i, V_j)$  in the constraint graph. This edge is actually two arcs: (1) one arc from  $V_i$  to  $V_j$  and (2) another arc from  $V_j$  to  $V_i$

The Arc  $(V_i, V_j)$  is arc consistent if for every value  $x$  the current domain of  $V_i$  there is some value  $y$  in the domain of  $V_j$  such that  $V_i=x$  and  $V_j=y$  is permitted by the binary constraint between  $V_i$  and  $V_j$ . Here, the sub-problem is limited to an arc. After ensuring the arc consistency of the Arc  $(V_i, V_j)$ , it is guaranteed that every value in the domain of  $V_i$  can participate in at least one valid solution (or be a part of one valid solution) of this sub-problem.

A binary constraint  $C(x_1, x_2)$  is arc consistent if for all values  $d_1 \in D(x_1)$  there exists a value  $d_2 \in D(x_2)$  such that  $(d_1, d_2) \in C$ , and for all values  $d_2 \in D(x_2)$  there exists a value  $d_1 \in D(x_1)$  such that  $(d_1, d_2) \in C$ . Here, the sub-problem is limited to a binary constraint. After ensuring the arc consistency of the binary constraint  $C(x_1, x_2)$ , it is guaranteed that for each variable, every value in the domain can be a part of at least one valid tuple.

Finally, A constraint  $C(x_1, \dots, x_m)$  ( $m > 1$ ) is hyper-arc consistent if for all  $i \in \{1, \dots, m\}$  and all values  $d_i \in D(x_i)$ , there exist values  $d_j \in D(x_j)$  for all  $j \in \{1, \dots, m\} - i$  such that  $(d_1, \dots, d_m) \in C$ . Here, the sub-problem is limited to a single constraint which can be binary or ternary or any constraint with arity  $> 1$ . Arc consistency is equal to hyper-arc consistency applied to binary constraints. hyper-arc consistency is also referred to as generalized arc-consistency (GAC).

Now, we are going to describe how to solve the 4-queens problem with MAC. It is an informal description of AC-3 algorithm.

Let us assume the CSP formulation of the 4-queens has four variables  $q_1, q_2, q_3, q_4$ . The variable  $q_i$  refers to the queen placed in column  $i$ . Each variable has a domain  $\{r_1, r_2, r_3, r_4\}$ . If we assign  $q_1 = r_4$ , it means the first queen is placed on row 4.

#### Step-1

Assign the value  $r_1$  to  $q_1$ .

$q_1 = \{r_1\}$

$q_2 = \{r_3, r_4\}$  /\* make the arc  $(q_2, q_1)$  arc consistent by removing value  $r_1, r_2$  \*/

$q_3 = \{r_2, r_4\}$  /\* make the arc  $(q_3, q_1)$  arc consistent by removing value  $r_1, r_3$  \*/

$q_4 = \{r_2, r_3\}$  /\* make the arc  $(q_4, q_1)$  arc consistent by removing value  $r_1, r_4$  \*/

#### Task list

check the arc  $(q_3, q_2)$  and arc  $(q_4, q_2)$  for arc consistency /\* this is because domain of  $q_2$  has changed \*/

check the arc  $(q_2, q_3)$  and arc  $(q_4, q_3)$  for arc consistency /\* this is because domain of  $q_3$  has changed \*/

check the arc  $(q_2, q_4)$  and arc  $(q_3, q_4)$  for arc consistency /\* this is because domain of  $q_4$  has changed \*/

let us check the arc  $q_3 \rightarrow q_2$

we can not remove  $r_2$  from the domain of  $q_3$  because  $[q_3 = r_2 \text{ and } q_2 = r_4]$  is valid

we can remove  $r_4$  from the domain of  $q_3$  because both  $[q_3 = r_4 \text{ and } q_2 = r_3]$  and  $[q_3 = r_4 \text{ and } q_2 = r_4]$  are not valid.

so,  $q_1 = \{r_1\}, q_2 = \{r_3, r_4\}, q_3 = \{r_2\}, q_4 = \{r_2, r_3\}$

Now, let us check the arc  $q_4 \rightarrow q_2$

we can not remove  $r_2$  from the domain of  $q_4$  because  $[q_4 = r_2 \text{ and } q_2 = r_3]$  is valid.

we can not remove  $r_3$  from the domain of  $q_4$  because  $[q_4 = r_3 \text{ and } q_2 = r_4]$  is valid.

so,  $q_1 = \{r_1\}, q_2 = \{r_3, r_4\}, q_3 = \{r_2\}, q_4 = \{r_2, r_3\}$

Now, let us check the arc  $q_2 \rightarrow q_3$

we can remove  $r_3$  from the domain of  $q_2$  because  $[q_2 = r_3 \text{ and } q_3 = r_2]$  is not valid.

we can not remove r4 from the domain of q2 because [q2 = r4 and q3 = r2] is valid.

so,  $q1 = \{r1\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r2\}$ ,  $q4 = \{r2, r3\}$

Now let us check the arc  $q4 \rightarrow q3$

we can remove r2 from the domain of q4 because [q4 = r2 and q3 = r2] is not valid.

we can remove r3 from the domain of q4 because [q4 = r3 and q3 = r2] is not valid.

so,  $q1 = \{r1\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r2\}$ ,  $q4 = \{\}$

The decision to backtrack is made because the domain of q4 is empty.

## Step-2

Assign the value r2 to q1.

$q1 = \{r2\}$

$q2 = \{r4\}$  /\* make the arc (q2, q1) arc consistent by removing value r2, r1, r3 \*/

$q3 = \{r1, r3\}$  /\* make the arc (q3, q1) arc consistent by removing value r2, r4 \*/

$q4 = \{r1, r3, r4\}$  /\* make the arc (q4, q1) arc consistent by removing value r2 \*/

## Task list

check the arc (q3, q2) and arc (q4, q2) for arc consistency /\* this is because domain of q2 has changed \*/

check the arc (q2, q3) and arc (q4, q3) for arc consistency /\* this is because domain of q3 has changed \*/

check the arc (q2, q4) and arc (q3, q4) for arc consistency /\* this is because domain of q4 has changed \*/

Let us check the arc  $q3 \rightarrow q2$

we can not remove r1 from the domain of q3 because [q3 = r1 and q2 = r4] is valid

we can remove r3 from the domain of q3 because [q3 = r3 and q2 = r4] is not valid

So,  $q1 = \{r2\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r1\}$ ,  $q4 = \{r1, r3, r4\}$

Let us check the arc  $q4 \rightarrow q2$

we can not remove r1 from the domain of q3 because [q4 = r1 and q2 = r4] is valid

we can not remove r3 from the domain of q3 because [q4 = r3 and q2 = r4] is valid

we can remove r4 from the domain of q4 because [q4 = r4 and q2 = r4] is not valid

So,  $q1 = \{r2\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r1\}$ ,  $q4 = \{r1, r3\}$

Let us check the arc  $q2 \rightarrow q3$

we can not remove  $r4$  from the domain of  $q2$  because  $[q2 = r4 \text{ and } q3 = r1]$  is valid

So,  $q1 = \{r2\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r1\}$ ,  $q4 = \{r1, r3\}$

Let us check the arc  $q4 \rightarrow q3$

we can remove  $r1$  from the domain of  $q4$  because  $[q4 = r1 \text{ and } q3 = r1]$  is not valid

we can not remove  $r3$  from the domain of  $q4$  because  $[q4 = r3 \text{ and } q3 = r1]$  is valid

So,  $q1 = \{r2\}$ ,  $q2 = \{r4\}$ ,  $q3 = \{r1\}$ ,  $q4 = \{r3\}$

We have found a solution.

You should visit the webpage <https://ktiml.mff.cuni.cz/~bartak/constraints/consistent.html> and you should work out Algorithm AC-1 by hand step by step. It is easier to understand MAC algorithm by AC-1, but AC-3 is more efficient.