

## 1 Presented Problems

### Problem 3.1: Turning $n$ -ary constraints into binary constraints

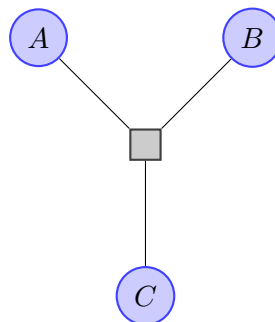
(from *Russell & Norvig 3ed.* q. 7.6) Suppose that we have  $CSP = (X, D, E^1)$  with

$$\begin{aligned} X &= \{A, B, C\}, \\ D &= \{\text{dom}(A), \text{dom}(B), \text{dom}(C)\}, \\ E &= \{\langle (A, B, C), A + B = C \rangle\}, \end{aligned}$$

where  $\text{dom}(A)$ ,  $\text{dom}(B)$ , and  $\text{dom}(C)$  denote the domain of variable  $A$ ,  $B$ , and  $C$ , respectively, and each domain can be  $\{0, 1, \dots, 9\}$  for example.

**Problem 3.1.1:** Draw the constraint hypergraph for the CSP. In this case, a hypergraph is a graph with two types of nodes. The first type of node represents the *variables*, depicted by  $\bigcirc$ , and the second type of node represents the constraint, depicted by  $\square$ . Based on the number of variables involved, what is the type of the constraint?

*Solution:*



There is only one constraint in set  $E$ , and there are three variables involved in this constraint. Therefore, this constraint is a higher-order constraint (3-ary constraint).

**Problem 3.1.2:** We can eliminate the higher-order constraint in  $E$  by replacing the constraint node  $\square$  with a new variable node  $Z$ . (We denote this new CSP as  $CSP'$ .) What is the domain for variable  $Z$ ? (Hint: The domain for variable  $Z$  can be ordered pairs of other values.) What is the new constraint set  $E'$  after introducing the new variable  $Z$ ?

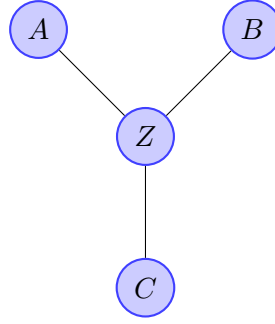
*Solution:* The domain for variable  $Z$  is a 3-tuple:

$$\text{dom}(Z) = \{(z_1, z_2, z_3) \mid z_1 \in \text{dom}(A) \wedge z_2 \in \text{dom}(B) \wedge z_3 \in \text{dom}(C)\}.$$

Instead of the higher-order constraint  $E$ , we now have the constraint set  $E'$  containing an unary constraint on node  $Z$ , which is  $\langle (Z), z_1 + z_2 = z_3 \rangle$ , and binary constraints between adjacent nodes. The new CSP

---

<sup>1</sup>the symbol  $E$  is taken from German word *Einschränkung*.



can be formulated as  $CSP' = (X', D', E')$  with

$$\begin{aligned}
X' &= \{A, B, C, Z\}, \\
D' &= D \cup \text{dom}(Z), \\
E' &= \{ \langle (A, Z), \text{fst}(Z) = A \rangle, \langle (B, Z), \text{snd}(Z) = B \rangle, \langle (C, Z), \text{thrd}(Z) = C \rangle, \\
&\quad \langle (Z), z_1 + z_2 = z_3 \rangle \},
\end{aligned}$$

where *fst*, *snd*, and *thrd* are operators to get the first, second, and third element of the tuple in  $Z$ .

**Problem 3.1.3:** Modify  $CSP'$  such that it only contains binary constraints and formally express the new  $CSP'' = (X'', D'', E'')$ .

*Solution:* We can eliminate the unary constraint of a variable by altering its domain such that all values in the domain satisfy the constraint. This is called node-consistency.

Thus, we reduce the domain of variable  $Z$  to be

$$\text{dom}(Z) = \{(z_1, z_2, z_3) \mid z_1 + z_2 = z_3 \wedge z_1 \in \text{dom}(A) \wedge z_2 \in \text{dom}(B) \wedge z_3 \in \text{dom}(C)\}.$$

As a result, we obtain a CSP with only binary constraints:  $CSP'' = (X'', D'', E'')$  with

$$\begin{aligned}
X'' &= \{A, B, C, Z\}, \\
D'' &= D \cup \text{dom}(Z), \\
E'' &= \{ \langle (A, Z), \text{fst}(Z) = A \rangle, \langle (B, Z), \text{snd}(Z) = B \rangle, \langle (C, Z), \text{thrd}(Z) = C \rangle \}.
\end{aligned}$$

**Problem 3.1.4:** Taking inspiration from previous solutions, how can you generally turn a  $n$ -ary constraint into binary constraints?

*Solution:* Suppose that we have a constraint of order  $n$ . We can represent this constraint of order  $n$  with a relation  $R(x_1, x_2, \dots, x_n)$  of order  $n$ .

1. We replace the  $n$ -ary constraint by a new variable  $Z$ . The domain of  $Z$  is a  $n$ -tuple and must be restricted such that it satisfies the relation  $R$  (e.g., see the domain of variable  $Z$  in Problem 3.1.3).
2. Then, we introduce new binary constraints to match the values of variable  $Z$  with the values of the neighboring variables  $x_1, x_2, \dots, x_n$  (i.e.,  $\text{fst}(Z) = x_1$ ,  $\text{snd}(Z) = x_2$ , and so on).

This procedure is known as hidden transformation<sup>2</sup>. Research also gives guidance for when transforming CSP is beneficial<sup>3</sup>.

<sup>2</sup>[https://doi.org/10.1016/S0004-3702\(02\)00210-2](https://doi.org/10.1016/S0004-3702(02)00210-2)

<sup>3</sup><http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.7551&rep=rep1&type=pdf>

### Problem 3.2: Arc consistency and backtracking search for binary constraints

Consider the constraint satisfaction problem in Fig. 1. According to the picture, we have  $CSP = (X, D, E)$  with

$$\begin{aligned} X &= v_1, v_2, v_3, v_4, v_5, \\ D &= \{\text{dom}(v_1), \text{dom}(v_2), \text{dom}(v_3), \text{dom}(v_4), \text{dom}(v_5)\}, \\ E &= \{ \langle (v_1, v_2), v_2 = v_1 + 1 \rangle, \\ &\quad \langle (v_1, v_3), v_1 \neq v_3 \rangle, \\ &\quad \langle (v_2, v_3), v_2 \neq v_3 \rangle, \\ &\quad \langle (v_3, v_4), v_3 \neq v_4 \rangle, \\ &\quad \langle (v_3, v_5), v_3 \neq v_5 \rangle, \\ &\quad \langle (v_4, v_5), v_4 \neq v_5 \rangle, \\ &\quad \langle (v_1, v_5), v_1 \neq v_5 \rangle \}, \end{aligned}$$

where  $\text{dom}(v_1)$ ,  $\text{dom}(v_2)$ ,  $\text{dom}(v_3)$ ,  $\text{dom}(v_4)$  and  $\text{dom}(v_5)$  denote the domain of variable  $v_1$ ,  $v_2$ ,  $v_3$ ,  $v_4$  and  $v_5$ , respectively, and each domain is initially  $\{2, 3, 4\}$ . Note that all constraints in the graph are binary constraints.

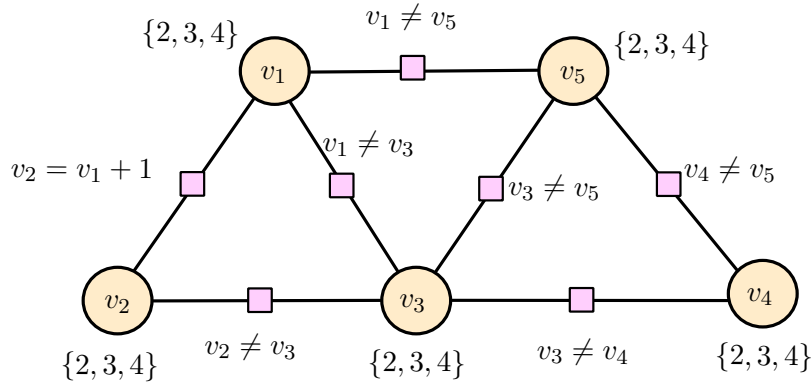


Figure 1: Constraint graph for Problem 3.2

**Problem 3.2.1:** Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

*Solution:* Note that the minimum-remaining-values (MRV) heuristic always chooses the variable that has the smaller domain, while degree heuristics chooses the variable with the highest degree, i.e., involved in the highest number of constraints.

Initially all variables have the same domain size, i.e., 3. Using Fig. 1, we can count the degree of each variable and then sort the variables by their degree (cf. Table 1).

Variable	Degree
$v_3$	4
$v_1, v_5$	3
$v_2, v_4$	2

Table 1: Variables sorted by their degree.

**Problem 3.2.2:** Perform backtracking search by hand to solve the CSP problem: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

*Solution:* For each recursion depth, the assigned variable and the remaining values for each variable (after performing forward checking) is shown in Table 2. If the number of remaining values is zero for at least one variable, we need to backtrack. In the following, we describe the search steps (where the bold number denotes the depth of recursion).

step	assignment	current domains					degree					backtrack
		$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	
0	$\emptyset$	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	3	2	4	2	3	
1	$v_3 = 3$	{2, 4}	{2, 4}	/	{2, 4}	{2, 4}	2	1	/	1	2	
2	$v_1 = 2$	/	$\emptyset$	/	{2, 4}	{4}	/	0	/	1	1	backtrack to 2
2	$v_1 = 4$	/	$\emptyset$	/	{2, 4}	{2}	/	0	/	1	1	backtrack to 1
1	$v_3 = 2$	{3, 4}	{3, 4}	/	{3, 4}	{3, 4}	2	1	/	1	2	
2	$v_1 = 3$	/	{4}	/	{3, 4}	{4}	/	0	/	1	1	
3	$v_5 = 4$	/	{4}	/	{3}	/	/	0	/	0	/	
4	$v_2 = 4$	/	/	/	{3}	/	/	/	/	0	/	
5	$v_4 = 3$	/	/	/	/	/	/	/	/	/	/	

Table 2: Backtracking search with forward checking as inference. For each variable, the number of remaining values (domain size) and the number of constraints (degree) is listed. The rows correspond to the steps of the backtracking algorithm, we assign a value to one variable and perform forward checking (i.e. updating the domain of connected variables).

- 1** We apply MRV heuristics to choose the variable to expand first. Since all variables have the same number of remaining values, MRV holds a tie. Thus, we use degree heuristics and choose the variable with the highest degree (cf. Table 1), i.e.,  $v_3$ . To choose a value, we use least-constraining-value heuristic. Every value of  $v_3$  will leave the same number of values for the neighboring variables  $v_1$ ,  $v_2$ ,  $v_4$  and  $v_5$ . Since we have a tie, we choose  $v_3 = 3$  (see exercise description). Then, we perform forward checking: The reduced domain of  $v_1$ ,  $v_2$ ,  $v_4$  and  $v_5$  are all equal to  $\{2, 4\}$ . All remaining variables were connected to the assigned variable  $v_3$  by a constraint, thus we reduce their degree by one.
- 2** All variables have the same number of remaining values, but  $v_1$  and  $v_5$  have a higher degree. We choose variable  $v_1$ , since it has the smaller lower index. Applying least-constraining-value heuristic shows that both values of  $v_1$  leave the same number of values for  $v_2$  and  $v_5$ . Since we cannot assign the value of 3, we assign the lowest value  $v_1 = 2$ . For forward checking, we update the variable  $v_2$  (and  $v_5$ ) and we see that its domain size is reduced to zero, and we backtrack immediately to the assignment of  $v_1$  restoring the domains.
- 2** The only other possible value for  $v_1$  is 4. Therefore, we assign  $v_1 = 4$ . During forward checking, we update the variable  $v_2$  (and  $v_5$ ) and we see that its domain size is again reduced to zero, so we backtrack and restore the domains. Since we already tried all possible values for  $v_1$ , we backtrack to the assignment of  $v_3$ .
- 1** We assign  $v_3 = 2$  and reduce the domain of all other variables to  $\{3, 4\}$ .

- 2 Analogously to the previous try, we have a tie between  $v_1$  and  $v_5$  and choose variable  $v_1$ . Using least-constraining-value heuristic, we choose  $v_1 = 3$  (Note: Assignment  $v_1 = 4$  would leave no options for  $v_2$ ). With forward checking we reduce the domains of  $v_2$  and  $v_5$ .
- 3 Notice that the domain size of  $v_2$  is equal to one and the variable itself is not involved in any constraints. Since we did not backtrack, this actually means that a valid assignment for  $v_2$  has already been found. But according to MRV heuristics and degree heuristics we select  $v_5 = 4$  and reduce the domain of  $v_4$ .
- 4 The variable  $v_2$  and  $v_4$  have the same number of remaining values and the same degree. Since it is a tie, we choose the variable with the smaller lower index, i.e.,  $v_2$ . Thus, we assign  $v_2 = 4$ .
- 5 In the final step we choose  $v_4 = 3$ .

**Problem 3.2.3:** Perform backtracking search again, but with a different inference: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform the arc consistency algorithm. Backtrack if you find an inconsistency.

*Solution:* For each recursion depth, the assigned variable and the remaining values for each variable is shown in Table 3 (after performing arc consistency algorithm). If the number of remaining values is zero for at least one variable, we need to backtrack. In the following, we describe the search steps (where the bold number denotes the depth of recursion).

step	assignment	current domains					degree					backtrack
		$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	
0	$\emptyset$	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	3	2	4	2	3	backtrack to 1
1	$v_3 = 3$	{2, 4}	$\emptyset$	/	{2, 4}	{2, 4}	2	1	/	1	2	
1	$v_3 = 2$	{3}	{4}	/	{3}	{4}	2	1	/	1	2	
2	$v_1 = 3$	/	{4}	/	{3}	{4}	/	0	/	1	1	
3	$v_4 = 3$	/	{4}	/	/	{4}	/	0	/	/	0	
4	$v_2 = 4$	/	/	/	/	{4}	/	/	/	/	0	
5	$v_5 = 4$	/	/	/	/	/	/	/	/	/	/	

Table 3: Backtracking search with arc consistency algorithm as inference. For each variable, the number of remaining values (domain size) and the number of constraints (degree) is listed. The rows correspond to the steps of the backtracking algorithm, we assign a value to one variable and perform arc consistency algorithm (i.e. updating the domain of all variables).

- 1 Analogously to the first step of the previous subproblem, we choose  $v_3 = 3$ . We apply the arc consistency algorithm, initializing the queue with the arcs  $(v_1, v_3)$ ,  $(v_2, v_3)$ ,  $(v_4, v_3)$  and  $(v_5, v_3)$ :
  - We pop the first element from the queue, i.e.  $(v_1, v_3)$ , and remove the inconsistent values. Since we remove the value 3 from the domain of  $v_1$ , we add the arcs to all neighbours of  $v_1$  (except  $v_3$ ) to the queue, i.e.,  $(v_2, v_1)$  and  $(v_5, v_1)$ .
  - In the next step, we check arc consistency for  $(v_2, v_3)$  and thus remove the value 3 from the domain of  $v_2$ . Again we add the arcs to all neighbours of  $v_2$  (except  $v_3$ ) to the queue, i.e.,  $(v_1, v_2)$ .

- In the same way the inconsistent value 3 is removed from the domains of  $v_4$  and  $v_5$  and the arcs to all neighbours of  $v_4$  and  $v_5$  (except  $v_3$ ), respectively, are added to the queue. The queue after these steps is:  $(v_2, v_1)$ ,  $(v_5, v_1)$ ,  $(v_1, v_2)$ ,  $(v_5, v_4)$ ,  $(v_1, v_5)$ ,  $(v_4, v_5)$ .
- In the following step, we make  $v_2$  arc consistent with  $v_1$ . Therefore, we remove the values 2 and 4 from the domain of  $v_2$ . The arc consistency algorithm returns failure, since the size of the domain of  $v_2$  is zero. We immediately backtrack to the assignment of  $v_3$  restoring the domains.

1 According to the exercise description, we assign to  $v_3$  the lowest value: we choose  $v_3 = 2$ . We apply again the arc consistency algorithm, initializing the queue with the arcs  $(v_1, v_3)$ ,  $(v_2, v_3)$ ,  $(v_4, v_3)$  and  $(v_5, v_3)$ :

- Similar to the previous assignment, the value 2 is removed from the domains of  $v_1$ ,  $v_2$ ,  $v_4$  and  $v_5$ , and the arcs to all neighbours of  $v_1$ ,  $v_2$ ,  $v_4$  and  $v_5$  (except  $v_3$ ), respectively, are added to the queue. The queue after these steps is:  $(v_2, v_1)$ ,  $(v_5, v_1)$ ,  $(v_1, v_2)$ ,  $(v_5, v_4)$ ,  $(v_1, v_5)$ ,  $(v_4, v_5)$ .
- After that, we remove the inconsistent value 3 from  $v_2$  in order to make  $v_2$  arc consistent with  $v_1$ . We add the arcs to all neighbours of  $v_2$  (except  $v_1$ ), i.e.,  $(v_3, v_2)$ .
- $(v_5, v_1)$  is already arc consistent. Thus, we do not remove any values from the domain of  $v_5$  and do not add any new arcs to the queue.
- Next, we make  $v_1$  arc consistent with  $v_2$  by removing the value 4 from the domain of  $v_1$ . Then we add the arcs to all neighbours of  $v_1$  (except  $v_2$ ) to the queue, i.e.,  $(v_3, v_1)$  and  $(v_5, v_1)$ . The queue after these steps is:  $(v_5, v_4)$ ,  $(v_1, v_5)$ ,  $(v_4, v_5)$ ,  $(v_3, v_2)$ ,  $(v_3, v_1)$ ,  $(v_5, v_1)$ .
- Since  $(v_5, v_4)$ ,  $(v_1, v_5)$ ,  $(v_4, v_5)$ ,  $(v_3, v_2)$  and  $(v_3, v_1)$  are already arc consistent, these arcs are popped from the queue and no new arcs are added to the queue.
- In order to make  $v_5$  arc consistent with  $v_1$ , the value 3 is removed from the domain of  $v_5$ . We add the arcs to all neighbours of  $v_5$  (except  $v_1$ ) to the queue, i.e.,  $(v_3, v_5)$ ,  $(v_4, v_5)$  to the queue.
- $(v_3, v_5)$  is already arc consistent. Thus, this arc is popped from the queue without any changes in the domain of  $v_3$ .
- Now we remove the inconsistent value 4 from the domain of  $v_4$  in order to make  $v_4$  arc consistent with  $v_5$ . We add the arcs to all neighbours of  $v_4$  (except  $v_5$ ) to the queue, i.e.,  $(v_3, v_4)$ .
- Since  $(v_3, v_4)$  is already arc consistent, this arc is popped from the queue and the algorithm terminates, as the queue is empty.

The resulting CSP has only one possible value per variable and, since we did not backtrack, it is arc consistent. This means that we have actually already found a solution for the CSP. In the next steps we assign the found values to the variables in the order defined by the exercise description.

2 Since  $v_1$  and  $v_5$  have both the highest degree, we have a tie and assign  $v_1 = 3$ .

3 Similar to the previous step, we have a tie between  $v_4$  and  $v_5$ . Thus, we assign  $v_4 = 3$ .

4 Now we have a tie between  $v_2$  and  $v_5$  and assign  $v_2 = 4$ .

5 Finally, we assign  $v_5 = 4$ .

**Problem 3.2.4:** Consider the CSP in Fig. 1 at its initial state. Is the CSP arc consistent? Is this a convenient initial condition if we plan to apply backtracking search? Describe the domains after the arc consistency algorithm has been applied to the CSP as a preprocessing step.

*Solution:* In order to find out whether the CSP is arc consistent or not, we could apply the arc consistency algorithm by initializing the queue with all arcs in the CSP. If one or more domains are pruned,

then the CSP was initially not arc consistent. However, this is actually not strictly necessary in this case, since the CSP has only few variables with small domains and relatively simple constraints. In general, variables  $X$  and  $Y$  with equal domains 2, 3, 4 involved in a  $X \neq Y$  constraint are always arc consistent with each other: independent of which value from the domain 2, 3, 4 is selected for  $X$ , there will always be a feasible value among 2, 3, 4 for  $Y$ . Therefore, in our CSP all variables involved in an inequality constraint are already arc consistent with each other.

Let us now consider the constraint  $v_2 = v_1 + 1$  between variable  $v_1$  and  $v_2$ , both having domain 2, 3, 4. If we assigned  $v_1 = 4$ , there would not be any value left for  $v_2$ , since  $4 + 1 = 5$  and 5 is not in the domain of  $v_2$ . This means that the arc  $(v_1, v_2)$  is not arc consistent. A similar reasoning holds for the arc  $(v_2, v_1)$ . Therefore, the CSP is not arc consistent and thus is not in a convenient initial situation for applying backtracking search, since those will probably lead to a higher number of iteration and/or backtracking steps. In order to make both arcs  $(v_1, v_2)$  and  $(v_2, v_1)$  arc consistent, we need to prune the domains of  $v_1$  and  $v_2$  to  $\{2, 3\}$  and  $\{3, 4\}$  respectively. The resulting CSP is shown in Fig. 2.

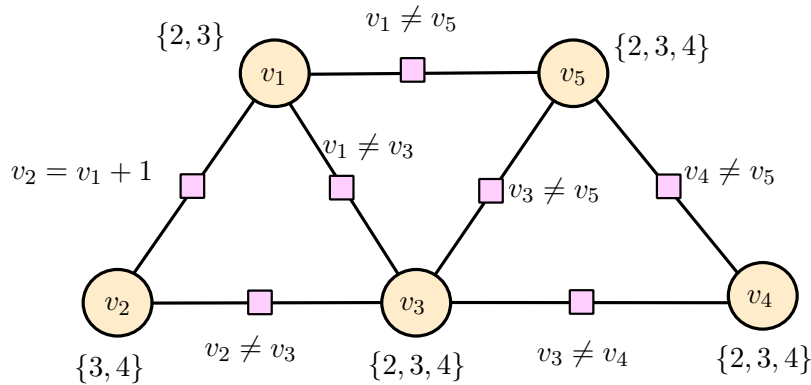


Figure 2: Constraint graph after applying the arc consistency algorithm for preprocessing

**Problem 3.2.5:** Perform backtracking search after the preprocessing step of the previous task: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose one randomly. After each assignment, perform arc consistency algorithm. Backtrack if you find an inconsistency. Assume the data structure of the queue is a set, i.e., if we add an element to the queue which is already in the queue, the element will not be added a second time (each element is unique).

*Solution:* For each recursion depth, the assigned variable and the remaining values for each variable is shown in Table 4 (after performing arc consistency algorithm). The search starts after we used the arc consistency algorithm for preprocessing, i.e., we made the CSP arc consistent. In the following, we describe the search steps (where the bold number denotes the depth of recursion).

**1** The variables with minimum remaining values are  $v_1$  and  $v_2$ , but  $v_1$  has a higher degree. Since the least-constraining-value heuristic holds a tie, we randomly assign  $v_1 = 2$ . We apply the arc consistency algorithm, initializing the queue with the arcs  $(v_2, v_1)$ ,  $(v_3, v_1)$  and  $(v_5, v_1)$ :

- We pop the first element from the queue, i.e.  $(v_2, v_1)$ , and remove the inconsistent value 4 from the domain of  $v_2$ . Then we add the arcs to all neighbours of  $v_2$  (except  $v_1$ ) to the queue, i.e.,  $(v_3, v_2)$ .
- In the following step,  $v_3$  is checked for arc consistency with  $v_1$ . We have to remove the value 2 from the domain of  $v_3$  to make it arc consistent with  $v_1$ . Again, we add the arcs to all neighbours of  $v_3$  (except  $v_1$ ) to the queue, i.e.,  $(v_2, v_3)$ ,  $(v_4, v_3)$ , and  $(v_5, v_3)$ .

step	assignment	current domains					degree					backtrack
		$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	
0	$\emptyset$	{2, 3}	{3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	3	2	4	2	3	
1	$v_1 = 2$	/	{3}	{4}	{2}	{3}	/	1	3	2	2	
2	$v_3 = 4$	/	{3}	/	{2}	{3}	/	0	/	1	1	
3	$v_4 = 2$	/	{3}	/	/	{3}	/	0	/	/	0	
4	$v_5 = 3$	/	{3}	/	/	/	/	0	/	/	/	
5	$v_2 = 3$	/	/	/	/	/	/	/	/	/	/	

Table 4: For each variable, the number of remaining values (domain size) and the number of constraints (degree) is listed. The rows correspond to the steps of the backtracking algorithm, we assign a value to one variable and perform arc consistency algorithm (i.e. updating the domain of connected variables).

- Next, we look at the arc  $(v_5, v_1)$ . We remove the inconsistent value 2 from the domain of  $v_5$  and add the arcs to all neighbours of  $v_5$  (except  $v_1$ ) to the queue. Thus, the queue after this step is:  $(v_3, v_2), (v_2, v_3), (v_4, v_3), (v_5, v_3), (v_3, v_5), (v_4, v_5)$ .
- The arc  $(v_3, v_2)$  is not consistent, since there is no value in the domain of  $v_2$  which fulfills the constraint  $v_2 \neq v_3$  for the value 3 in the domain of  $v_3$ . We remove the inconsistent value 3 from the domain of  $v_3$  and add the arcs to all neighbours of  $v_3$  (except  $v_2$ ) to the queue, i.e.,  $(v_1, v_3), (v_4, v_3)$  and  $(v_5, v_3)$ . Since the data structure of the queue is a set, and  $(v_4, v_3)$  and  $(v_5, v_3)$  are already in the queue, these arcs are not added to the queue a second time. Thus, the queue after this step is:  $(v_2, v_3), (v_4, v_3), (v_5, v_3), (v_3, v_5), (v_4, v_5), (v_1, v_3)$ .
- $(v_2, v_3)$  is already arc consistent. Therefore, we do not remove any values from the domain of  $v_2$  and do not add any new arcs to the queue.
- In the next step, we remove the inconsistent value 4 from the domain of  $v_4$  to make  $v_4$  arc consistent with  $v_3$ . We add  $(v_5, v_4)$  to the queue.
- In order to make  $v_5$  arc consistent with  $v_3$ , we have to remove the value 4 from the domain of  $v_5$ . Then we add the arcs to all neighbours of  $v_5$  (except  $v_3$ ) to the queue, i.e.,  $(v_1, v_5)$  and  $(v_4, v_5)$ . However, as  $(v_4, v_5)$  is already in the queue, only  $(v_1, v_5)$  is added.
- Since  $(v_3, v_5)$  is already arc consistent, this arc is popped from the queue and no new arcs are added to the queue. The queue after these steps is:  $(v_4, v_5), (v_1, v_3), (v_5, v_4), (v_1, v_5)$ .
- Next, we remove the value 3 from the domain of  $v_4$  to make it arc consistent with  $v_5$ . We add the arcs to all neighbours of  $v_4$  (except  $v_5$ ) to the queue, i.e.,  $(v_3, v_4)$ .
- Since the remaining arcs  $(v_1, v_3), (v_5, v_4), (v_1, v_5)$  and  $(v_3, v_4)$  are already arc consistent, these arcs are popped from the queue and no new arcs are added to the queue. As the queue is empty, the algorithm terminates.

After applying the arc consistency algorithm the resulting CSP has only one possible value per variable and, since we did not backtrack, it is arc consistent. We have actually already found a solution for the CSP. Note that this would also hold for a different assignment of  $v_1$ . In the next steps we assign the found values to the variables in the order defined by the exercise description.

- 2 Since  $v_3$  has the highest degree, we assign  $v_3 = 4$ .
- 3 In the next step,  $v_4$  and  $v_5$  have both the highest degree, we have a tie and assign randomly  $v_4 = 2$ .
- 4 Similar to the previous step, we have a tie between  $v_2$  and  $v_5$ . Thus, we assign randomly  $v_5 = 3$ .
- 5 Finally, we assign  $v_2 = 3$ .

**Problem 3.2.6:** For each of the previous performances of backtracking search with a different infer-



ence (forward checking, arc consistency, arc consistency after preprocessing), compare the number of iterations and the number of times you needed to backtrack.

*Solution:*

**Forward Checking.** The backtracking search required eight iterations and two backtracking steps. (Note: Assigning  $v_3 = 2$  or  $v_3 = 4$  in the first step would have solved the CSP without backtracking once.) There are no solutions for the assignment  $v_3 = 3$ , but forward checking was not able to identify this inconsistency immediately and we noticed it only after ruling out all options for  $v_1$ .

**Arc Consistency Algorithm.** A total of six iterations and one backtracking step was required. Note that the arc consistency algorithm was able to immediately identify that the assignment  $v_3 = 3$  leads to a subproblem with no solution. Furthermore, by looking at the state of all variable domains after assigning  $v_3 = 2$  and performing the inference we could tell that a solution exists. That was not the case for forward checking. This fact underlines a fundamental difference between the two inferences: forward checking establishes arc consistency between the neighbours of the assigned variable and the assigned variable itself (i.e. makes all arcs arc consistent in both directions for a radius of length one from the assigned variable); the arc consistency algorithm makes the whole CSP arc consistent.

**Arc Consistency Algorithm after preprocessing.** The backtracking search finished within five iterations without backtracking. While applying the arc consistency algorithm as a preprocessing step does not exclude the possibility of backtracking, it surely speeds up the whole search. Note that arc consistency algorithm is not always the best inference, since it is more resource consuming. However, applying the arc consistency algorithm as a preprocessing step is always a good choice. For a practical example, take a look at the additional problems, where we solve this CSP with backtracking search applying forward checking after preprocessing with the arc consistency algorithm.

## 2 Additional Problems

### Problem 3.3: Arc consistency and backtracking search for binary constraints

Consider again the constraint satisfaction problem from Problem 3.2 described in Fig. 1.

**Problem 3.3.1:** Perform backtracking search by hand with forward checking as inference after preprocessing with the arc consistency algorithm: Apply the arc consistency algorithm to the CSP initializing the queue with all arcs, then start backtracking search. Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose randomly. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

*Solution:* For each recursion depth, the assigned variable and the remaining values for each variable are shown (after performing forward checking) in Table 5. In the following, we describe the search steps (where the bold number denotes the depth of recursion).

step	assignment	current domains					degree					backtrack
		$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	
0	$\emptyset$	{2, 3}	{3, 4}	{2, 3, 4}	{2, 3, 4}	{2, 3, 4}	3	2	4	2	3	
1	$v_1 = 2$	/	{3}	{3, 4}	{2, 3, 4}	{3, 4}	/	1	3	2	2	
2	$v_2 = 3$	/	/	{4}	{2, 3, 4}	{3, 4}	/	/	2	2	2	
3	$v_3 = 4$	/	/	/	{2, 3}	{3}	/	/	/	1	1	
4	$v_5 = 3$	/	/	/	{2}	/	/	/	/	0	/	
5	$v_4 = 2$	/	/	/	/	/	/	/	/	/	/	

Table 5: Backtracking search with forward checking as inference after preprocessing with the arc consistency algorithm.

- 1 MRV heuristics holds a tie between variables  $v_1$  and  $v_2$ . Since  $v_1$  has a higher degree and least-constraining-value heuristic holds a tie, we assign randomly  $v_1 = 2$ . Using forward checking prunes the domains of  $v_2$ ,  $v_3$  and  $v_5$ .
- 2 The variable with minimum remaining values is  $v_2$ . Since there is just one value left, we assign  $v_2 = 3$ . Forward checking leaves only one value for  $v_3$ .
- 3 The only possible value for  $v_3$  is 4. Therefore, we assign  $v_3 = 4$  and prune the domain of the neighboring variables  $v_4$  and  $v_5$ .
- 4 There is only one value left for  $v_5$ . We assign  $v_5 = 3$  and forward checking prunes the domain of  $v_4$ .
- 5 The only assignment left is  $v_4 = 2$ .

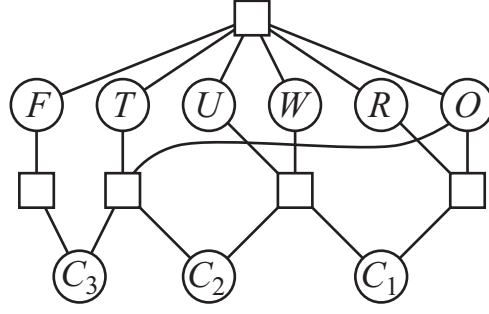
The search ended without backtracking once. Furthermore, choosing the variable and the value to assign was straightforward. Choosing a different value for  $v_1$  in the first step does not change these aspects.

**Problem 3.4: Solving a CSP by hand performing backtracking search with minimum-remaining-values (MRV) and degree heuristics, least-constraining-value heuristics, and forward checking**

(from *Russell & Norvig 3ed.* q. 7.5) Suppose that we have the cryptarithmic problem as shown in Fig. 3.

$$\begin{array}{r}
T \ W \ O \\
+ \ T \ W \ O \\
\hline
F \ O \ U \ R
\end{array}$$

(a)



(b)

Figure 3: (a) A cryptarithmic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct. (b) The constraint hypergraph for the cryptarithmic problem, showing the `AllDiff` constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables  $C1$ ,  $C2$ , and  $C3$  represent the carry digits for the three columns.

We model the cryptarithmic problem as  $CSP = (X, D, C)$  with

$$\begin{aligned}
X &= \{F, T, U, W, R, O, C_1, C_2, C_3\} \\
D &= \{\text{numbers}, \dots, \text{numbers}, \text{binary}, \text{binary}, \text{binary}\} \\
C &= \{ \langle (O, R, C_1), O + O = R + 10 \cdot C_1 \rangle, \\
&\quad \langle (U, W, C_1, C_2), C_1 + W + W = U + 10 \cdot C_2 \rangle, \\
&\quad \langle (O, T, C_2, C_3), C_2 + T + T = O + 10 \cdot C_3 \rangle, \\
&\quad \langle (C_3, F), C_3 = F \rangle, \\
&\quad \langle (F, T, U, W, R, O), \text{AllDiff}(F, T, U, W, R, O) \rangle \}
\end{aligned}$$

where  $\text{numbers} = \{0, 1, 2, \dots, 9\}$  and  $\text{binary} = \{0, 1\}$ .

**Problem 3.4.1:** Replace all boxes which correspond to higher-order constraints by binary constraints. Use the approach of Problem 3.1 and introduce variables such as  $X_1$ ,  $X_2$ , and  $X_3$ , etc.

*Solution:* For constraint  $\langle (O, R, C_1), O + O = R + 10 \cdot C_1 \rangle$ , we introduce variable  $X_1$  with a domain of

$$X_1 \in \{(o, r, c_1) \mid o \in \text{numbers} \wedge r \in \text{numbers} \wedge c_1 \in \text{binary} \wedge o + o = r + 10 \cdot c_1\}.$$

Similarly, we also define  $X_2$  and  $X_3$  as

$$\begin{aligned}
X_2 &\in \{(u, w, c_1, c_2) \mid u, w \in \text{numbers} \wedge c_1, c_2 \in \text{binary} \wedge c_1 + w + w = u + 10 \cdot c_2\}, \\
X_3 &\in \{(o, t, c_2, c_3) \mid o, t \in \text{numbers} \wedge c_2, c_3 \in \text{binary} \wedge c_2 + t + t = o + 10 \cdot c_3\}.
\end{aligned}$$

We also need a variable  $X_4$  for the `AllDiff` constraint:

$$X_4 \in \{(f, t, u, w, r, o) \mid f, t, u, w, r, o \in \text{numbers} \wedge \text{AllDiff}(f, t, u, w, r, o)\}.$$

Thus, we get the constraint hypergraph as shown in Fig. 4, and the new constraint set  $C'$  is

$$\begin{aligned}
C' = \{ & \langle (O, X_1), fst(X_1) = O \rangle, \\
& \langle (R, X_1), snd(X_1) = R \rangle, \\
& \langle (C_1, X_1), thrd(X_1) = C \rangle, \\
& \langle (U, X_2), fst(X_2) = U \rangle, \\
& \langle (W, X_2), snd(X_2) = W \rangle, \\
& \langle (C_1, X_2), thrd(X_2) = C_1 \rangle, \\
& \langle (C_2, X_2), frth(X_2) = C_2 \rangle, \\
& \langle (O, X_3), fst(X_3) = O \rangle, \\
& \langle (T, X_3), snd(X_3) = T \rangle, \\
& \langle (C_2, X_3), thrd(X_3) = C_2 \rangle, \\
& \langle (C_3, X_3), frth(X_3) = C_3 \rangle, \\
& \langle (C_3, F), C_3 = F \rangle, \\
& \langle (F, X_4), fst(X_4) = F \rangle, \\
& \langle (T, X_4), snd(X_4) = T \rangle, \\
& \langle (U, X_4), thrd(X_4) = U \rangle, \\
& \langle (W, X_4), frth(X_4) = W \rangle, \\
& \langle (R, X_4), ffth(X_4) = R \rangle, \\
& \langle (O, X_4), sxth(X_4) = O \rangle \}.
\end{aligned}$$

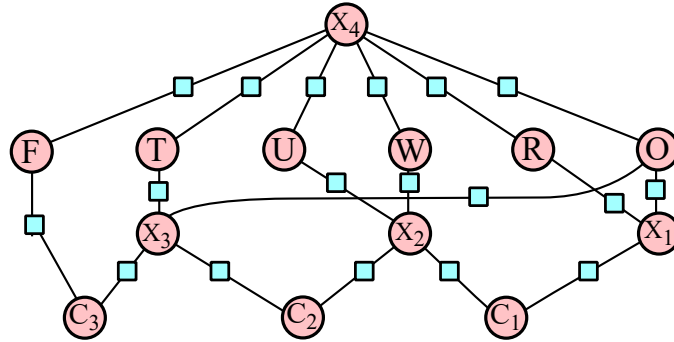


Figure 4: The constraint hypergraph for the cryptarithmic problem after turning all constraints into binary constraints.

**Problem 3.4.2:** Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

*Solution:*

**Domain size** Table 6 sorts the variables by their domain size. Whereas the domain size of the variables of the original CSP is easy to determine (see  $D$  in the given  $CSP$  formulation), we discuss the size of  $X_1$ ,  $X_2$ ,  $X_3$ , and  $X_4$  in the following.

Variable	Number of remaining values
$C_1, C_2, C_3$	2
$F, T, U, W, R, O$	10
$X_1$	10
$X_2, X_3$	20
$X_4$	$P(10, 6) = 151.200$

Table 6: Variables sorted by their number of remaining values.

**Combinatorial analysis for  $X_1$ .** If we assume  $c_1 = 0$ , we must find all value pairs of  $(o, r)$  which satisfy  $o + o = r$ . Since  $r \in \{0, \dots, 9\}$  and  $r$  must be even,  $(o, r) \in \{(0, 0), (1, 2), (2, 4), (3, 6), (4, 8)\}$ . Thus, we have 5 possible values for the tuple  $(o, r)$ .

If we assume  $c_1 = 1$ , we have the same 5 possible values for  $r$  as before, since  $o + o = r + 10$  can also only be satisfied if  $r$  is even. Together with  $o = \frac{r}{2} + 5$ , it follows that  $5 \leq o \leq 9$  and thus  $o \in \{5, 6, 7, 8, 9\}$ . In total, we have 10 possible values for variable  $X_1 = (o, r, c_1)$ .

**Combinatorial analysis for  $X_2$  and  $X_3$ .** Variables  $X_2$  and  $X_3$  have similar structure of their domain. Therefore, we analyze only  $X_2$ , and the result applies to  $X_3$  analogously.

If we assume  $c_1 = 0$ , the constraint becomes  $2w = u + 10 \cdot c_2$ , which is equivalent to the one of  $X_1$ . Thus, we have 10 possible values for  $(u, w, c_2)$ .

If we assume that  $(c_1, c_2) = (1, 0)$ , the constraint becomes  $1 + 2w = u$ . With  $u \leq 9$ , it follows that  $2w \leq 8$ . Again, 5 value pairs satisfy this inequality. If we assume that  $(c_1, c_2) = (1, 1)$ , the constraint becomes  $1 + 2w = u + 10$  and  $2w - 9 = u$ . Since the minimum value for  $u$  is 0, it follows that  $w \in \{5, \dots, 9\}$ . Thus, we have 5 possible values for this case.

In total,  $X_2$  has 20 possible values, as well as variable  $X_3$ .

**Combinatorial analysis for  $X_4$ .** Since we can assign 10 values to 6 variables which have to be different, the total number of possibilities is  $10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 = \frac{10!}{4!} = \frac{10!}{(10-6)!} = P(10, 6)$ , where  $P(n, k)$  denotes the  $k$ -permutations of  $n$ .

**Degree** In Fig. 4, we can count the degree of each variable. Table 7 sorts the variables by their degree.

Variable	Degree
$X_4$	6
$X_2, X_3$	4
$X_1$	3
$O$	3
$F, T, U, W, R$	2
$C_1, C_2, C_3$	2

Table 7: Variables sorted by the their degree.

**Problem 3.4.3:** Perform backtracking search to solve the cryptarithmic problem: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use

degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign. After each assignment, perform forward checking. Backtrack if you find an inconsistency.

*Solution:* The term backtracking search is used for a depth-first search that chooses values for one variable at a time and backtracks when a variable has no values left to assign (see lecture 5, slide 15). Whenever a variable is assigned, the forward checking process establishes arc consistency for this variable with its neighbors (i.e. reduces their domain so that all their values satisfy the binary constraints; see lecture 5, slide 26).

Table 8 summarizes the search: For each recursion depth, the assigned variable and the remaining values for each variable is shown (after performing forward checking). In the following, we describe the search steps (where the bold number denotes the depth of recursion).

step	$C_1$	$C_2$	$C_3$	$X_1$	$X_2$	$X_3$	$X_4$	$F$	$T$	$U$	$W$	$R$	$O$	assignment
0	2	2	2	10	20	20	$P(10,6)$	10	10	10	10	10	10	$\emptyset$
1	2	2	0	10	20	10	$P(10,6)$	1	10	10	10	10	10	$C_3 = 1$
2	2	2	0	10	20	10	$P(9,5)$	0	10	10	10	10	10	$F = 1$
3	2	0	0	10	10	5	$P(9,5)$	0	10	10	10	10	10	$C_2 = 1$
4	0	0	0	5	5	5	$P(9,5)$	0	10	10	10	10	10	$C_1 = 1$
5	0	0	0	5	5	0	$P(9,5)$	0	1	10	10	10	1	$X_3 = (5, 7, 1, 1)$
6	0	0	0	1	5	0	$P(8,4)$	0	1	10	10	10	0	$O = 5$
7	0	0	0	0	5	0	$P(8,4)$	0	1	10	10	1	0	$X_1 = (5, 0, 1)$
8	0	0	0	0	5	0	$P(7,3)$	0	0	10	10	1	0	$T = 7$
9	0	0	0	0	5	0	$P(6,2)$	0	0	10	10	0	0	$R = 0$
10	0	0	0	0	0	0	$P(6,2)$	0	0	1	1	0	0	$X_2 = (3, 6, 1, 1)$
11	0	0	0	0	0	0	$P(5,1)$	0	0	0	1	0	0	$U = 3$
12	0	0	0	0	0	0	1	0	0	0	0	0	0	$W = 6$
13	0	0	0	0	0	0	0	0	0	0	0	0	0	$X_4 = (1, 7, 3, 6, 0, 5)$

Table 8: For each variable, the number of remaining values is listed (rather than the list of the actual values, as in the previous executions of backtracking search). In each step (the rows corresponds to the recursion depth), we assign a value to one variable and perform forward checking (i.e. updating the domain of connected variables).

- 1 We use MRV heuristics to choose the variable to expand first. The variables  $C_1$ ,  $C_2$ , and  $C_3$  have the minimum number of remaining values (cf. Table 6), and also the same degree (cf. Table 7). Therefore, we randomly choose  $C_3$  to expand first. To choose a value, we use least-constraining-value heuristic. Every value of  $C_3$  will leave the same number of values for variables  $F$  and  $X_3$ . Thus, we just choose  $C_3 = 1$ . Then, we perform forward checking: The reduced domain of  $F$  is  $\{1\}$ , and for  $X_3$  it is

$$X_3 \in \{(o, t, c_2, 1) \mid o, t, c_2 \in \text{numbers} \wedge c_2 + t + t = o + 10\}.$$

- 2 Using MRV heuristic, we choose variable  $F$ . We assign it with value  $F = 1$ . For forward checking, we update the variable  $X_4$  such that the first value of the tuple is always equal to 1.
- 3 The variables with minimum remaining values are  $C_1$  and  $C_2$ , and they have the same degree. We just randomly choose  $C_2$ . Since least-constraining-value yields a tie, we just choose to assign  $C_2 = 1$ .

Performing forward checking, we update the variable  $X_2$  and  $X_3$  such that the  $c_2$  component will always be 1.

- 4 Next,  $C_1$  is chosen by MRV. Again, both possible values are the same in terms of least-constraining-value heuristic. We just choose  $C_1 = 1$ . Performing forward checking, we constraint the  $c_1$  component of  $X_1$  and  $X_2$  to be 1.

At this point, it might be helpful to list the possible values of  $X_1$ ,  $X_2$ , and  $X_3$ :

$$\begin{aligned} X_1 &\in \{(o, r, c_1) \mid (5, 0, 1), (6, 2, 1), (7, 4, 1), (8, 6, 1), (9, 8, 1)\} \\ X_2 &\in \{(u, w, c_1, c_2) \mid (1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\} \\ X_3 &\in \{(o, t, c_2, c_3) \mid (1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\} \end{aligned}$$

- 5 The variables  $X_1$ ,  $X_2$ , and  $X_3$  have the same number of remaining values and the same degree of 2. (Recall that the degree is the number of constraints on other unassigned variables.) We randomly choose  $X_3$ . Possible values for  $X_3$  are

$$X_3 = \{(1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\}$$

We try the least-constraining-heuristic; but whichever value we choose, the domains of  $T$  and  $O$  are both reduced to one. We just assign  $X_3 = (1, 5, 1, 1)$ . By forward checking, the domain size of  $T$  and  $O$  is reduced to 1.

- 6 Between  $T$  and  $O$  (both have only one remaining value), we choose  $O$  because it has a higher degree (which is 2), and assign  $O = 1$  (which is the only possible value). Forward checking reduces the domain size of  $X_1$  to 0 (see possible values above) and the domain size of  $X_4$  to 0. **Oh no!** The assigned value is not consistent. We backtrack immediately.

- 5 We again expand  $X_3$  with possible values of

$$X_3 = \{(1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\}$$

So let us now try  $X_3 = (3, 6, 1, 1)$ .

- 6 Thus, we assign  $O = 3$ , which reduces the domain size of  $X_1$  to 0. Backtrack again...

- 5 We again expand  $X_3$  with possible values of

$$X_3 = \{(1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\}$$

So let us now try  $X_3 = (5, 7, 1, 1)$ .

- 6 Thus, we assign  $O = 5$ , which reduces the domain size of  $X_1$  to 1 and of  $X_4$  to  $P(8, 4)$ .  
7  $T$  and  $X_1$  have both the least number of remaining values. Since they have the same degree of 1, we randomly choose  $X_1$ . We assign  $X_1$  with the only possible value  $X_1 = (5, 0, 1)$ . It leaves  $R$  with only 1 possible value.  
8  $T$  and  $R$  have the same number of remaining values and degree. We randomly choose  $T$ , and assign it with  $T = 7$ . This reduces the number of possible values for variable  $X_4$  to  $P(7, 3)$ .  
9 The next smallest domain is  $R$ , and we assign with the only possible value  $R = 0$ . By forward checking, we can reduce the domain size of  $X_4$  to  $P(6, 2)$ .  
10 The next smallest domain is  $X_2$ , and the possible values are

$$X_2 = \{(1, 5, 1, 1), (3, 6, 1, 1), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\}.$$

Since the least-constraining-value heuristics still does not tell us anything, we just try  $X_2 = (1, 5, 1, 1)$ . By performing forward checking, the number of remaining values of  $U$  and  $W$  is reduced to 1.

- 11 The next variable to assign is either  $U$  or  $W$  (same MRV and degree). We randomly choose  $U$  and assign the only possible value  $U = 1$ . By forward checking, we see that this reduces the domain size of  $X_4$  to 0. Backtrack!

**10** We again expand  $X_2$

$$X_2 = \{(1, 5, 1, 1), (\mathbf{3}, \mathbf{6}, \mathbf{1}, \mathbf{1}), (5, 7, 1, 1), (7, 8, 1, 1), (9, 9, 1, 1)\},$$

and try  $X_2 = (3, 6, 1, 1)$ . By performing forward checking, the domain of  $U$  and  $W$  are reduced to  $\{3\}$  and  $\{6\}$ , respectively.

**11** Now, only  $U$ ,  $W$ , and  $X_4$  are not yet assigned. Since  $U$  and  $W$  have a domain size of 1, but the same degree, we randomly choose  $U$  and assign  $U = 3$ . Forward checking reduces the domain size of  $X_4$  to  $P(5, 1) = 5$ .

**12** Next variable is  $W = 6$ . Forward checking reduces the domain size of  $X_4$  to  $P(4, 0) = 1$ .

**13** The last one to fill is variable  $X_4 = (1, 7, 3, 6, 0, 5)$ . And we have found a solution satisfying all (binary) constraints.