Universität Zürich[UZH]

Introduction to Artificial Intelligence
Exercise Sheet 4

Laurin van den Bergh, 16-744-401
Yufeng Xiao, 19-763-663
Nora Beringer, 19-734-227

Universität Zürich
Institut für Informatik

Due: March 23, 2022

Exercise 4.1

We can formulate a CSP for the Latin Square Problem as follows:
Set of variables X: X = {A, B, C, 1, 2, 3 }
The set of variables defines here the column and row space.
We have got a 3x3 Latin Square with rows = {A, B, C } and columns = {1, 2, 3 }.
Set of Domain D: D = {1, 2, 3 }
D denotes the variables used to fill the Latin Square.
Set of Constraints C: C = {A1 $\neq$ A2, A1 $\neq$ A3, A1 $\neq$ B1, A1 $\neq$ C1, A2 $\neq$ A3, A2 $\neq$ B2, A2 $\neq$ C2, A3 $\neq$ B3, A3 $\neq$ C3, $\sum$A = 6, $\sum$B = 6, $\sum$C = 6, $\sum$1 = 6, $\sum$2 = 6, $\sum$3 = 6, A $\neq \emptyset$, B $\neq \emptyset$, C $\neq \emptyset$, 1 $\neq \emptyset$, 2 $\neq \emptyset$, 3 $\neq \emptyset$ }
Where $X_{ij} \in$ X, hence for example A1 depicts the location of the variable in row A and column 1.
Neighbouring locations and locations in the same row and column space are only allowed to use unused variables from D, so that every column and row depicts each variable exactly once.
Therefore no neighbouring location and locations in the same row and column can show the same entry as its neighbour or as any location in the same row and column space.
Furthermore the sum of each row and of each column must be exactly 6 (as $1 + 2 + 3 = 6$), meanwhile each row and column must be completely filled with variables of the Domain D.

Exercise 4.2

a) node-consistent: A single variable (corresponding to a node in the CSP graph) is node-consistent if all the values in the variabe's domain satisfy the variable's unary constraints.

The unary constraint for dom(x) = {2, 4, 6} is $c_1 = \langle (x), x \in \{2x' | x' \in \mathbf{N}\} \rangle$.
This constraint is satisfied as every variable in dom(x) multiplied by 2 is in fact a natural number.

The unary constraint for dom(y) = {1, 4, 9} is $c_2 = \langle (y), y \neq 3 \rangle$.
This constraint is satisfied as every variable in dom(y) is $\neq 3$.

The unary constraint for dom(z) = {0, 1, 2, 3} is $c_3 = \langle (z), z < 4 \rangle$.
This constraint is satisfied as every variable in dom(z) is < 4.

b) arc-consistent: A variable is arc-consistent if every value in its domain satisfies the variable's binary constraints. More formally, X_i is arc-consistent with respect to another

variable X_j if for every value in the current Domain D_i there is some value in the Domain D_j that satisfies the binary constraint on the arc (X_i, X_j).

For $c_4 = \langle (x, y), x^2 = 4y \rangle$ this is not the case as only the variable 2 from dom(x) together with variable 1 from dom (y) satisfies this binary constraint ($2^2 = 4$).
There is no other variable in dom(x) which equals 4y when squared.

For $c_5 = \langle (y, z), y = z^2 \rangle$ this is not the case as for the variable $0 \in$ dom(z) the constraint is not fullfilled as there is no variable y' $\in$ dom(y) for which y' = 0^2 holds.

For $c_6 = \langle (x, z), x = 2z \rangle$ this is not the case as for the variable $0 \in$ dom(z) the constraint is not fullfilled as there is no variable x' $\in$ dom(x) for which x' = 2*0 holds.

c) path-consistent: Path-consistency tightens the binary constraints by using implicit constraints that are inferred by looking at triples of variables. A two-variable set $\{X_i, X_j\}$ is path-consistent with respect to a third variable X_m if, for every assignment $\{X_i = a, X_j = b\}$ consistent with the constraints (if any) on $\{X_i, X_j\}$, there is an assignment to X_m that satisfies the constraints on $\{X_i, X_m\}$ and $\{X_m, X_j\}$. The name refers to the overall consistency of the path from X_i to X_j with X_m in the middle.

An example for a path-consistency in this graph would be for
x' = 2, with x' $\in$ dom(x),
y' = 1, with y' $\in$ dom(y) and
z' = 1, with z' $\in$ dom(z).
For the pair (x,y), respectively (2, 1) the constraint $c_4 = \langle (x, y), x^2 = 4y \rangle$ holds.
Further the pair (x,z), respectively (2,1) holds for the constraint $c_6 = \langle (x, z), x = 2z \rangle$.
And finally for (y,z), respectively (1,1) holds for the constraint $c_5 = \langle (y, z), y = z^2 \rangle$.
Hence there is path-consistency as there is overall consistency of the path from X_i = x' $\in$ dom(x) to X_j = y' $\in$ dom(y) with X_m = z' $\in$ dom(z) in the middle.

d) strongly 3-consistent: A CSP is strongly 3-consistent if it is 3-consistent and is also (3 - 1)-consistent and (3 - 2)-consistent. 2-consistency is the same as arc-consistency. For binary constraint graphs, 3-consistency is the same as path-consistency.

As we have found a path-consistency in c) the CSP problem is also 3-consistent as we can find a way to connect the binary-constraints $c_4 = \langle (x, y), x^2 = 4y \rangle$, $c_5 = \langle (y, z), y = z^2 \rangle$ and $c_6 = \langle (x, z), x = 2z \rangle$ with each other.
Therefore we have found a 3-path connection linking dom(x), dom(y) and dom(z) together. (using the values x' $\in$ dom(x), y' = 1, with y' $\in$ dom(y) and z' = 1, with z' $\in$ dom(z))

Exercise 4.3

a) Please consider the graph for detailed understanding:

b) Variable and value ordering: V7=r, V3=b, V4=b, V1=g, V2=b, V5=g, V6=r/y
Please consider the graph for detailed understanding:

The search in b) is faster as the heuristic avoids pointless searches and therefore we don't need to do as much back-tracking as in a).

c) If we pick the variable which is most likely to fail soon then the failure will be detected immediately which therefore avoids pointless searches through other variables and on average have fewer sucessful assignments to backtrack over. The least-constraining value prefers the value that rules out the fewest choices for neighboring variables in the constriant graph. Therefore it leaves the maximum flexibility in order to assign following variables.

Exercise 4.4

a) No solution can be found for this inconsistent graph, as we end up with an empty set for either node v1 or for node v3. Please consider the graph for detailed understanding:

b) Following the pseudocode on page 171 from the Book Artificial Intelligence - A Modern Approach we get the following result:
queue = {(v1, v2), (v1, v3), (v2, v1), (v2, v4), (v2, v3), (v3, v1), (v3, v2), (v4,v2)}

In the first iteration of the while loop we set X_i = v1 and X_j = v2, meaning that we check the arc-consistency of (v1, v2).
dom(v1) = {1, 2}, dom(v2) = {1, 2}
For 1 ∈ dom(v1) < 1 ∈ dom(v2) doesn't hold but 1 ∈ dom(v1) < 2 ∈ dom(v2) holds. Therefore we do have an arc-consistency between the pair (v1, v2) for the values 1 ∈ dom(v1) and 2 ∈ dom(v2).
For 2 ∈ dom(v1) < 1 ∈ dom(v2) doesn't hold as well as 2 ∈ dom(v1) < 2 ∈ dom(v2) doesn't hold.
Therefore dom(v1) is revised in the function REVISE to dom(v1) = {1}, as the inconsistent variable is deleted from the domain.

In the second iteration the queue looks as following due to the popping of (v1,v2) during the first iteration:
queue = {(v1, v3), (v2, v1), (v2, v4), (v2, v3), (v3, v1), (v3, v2), (v4,v2)}
We check the arc-consistency of the pair (v1, v3), with dom(v1) = {1} and dom(v3) = {2, 3, 4}.
For 1 ∈ dom(v1) >= 2 ∈ dom(v3), 1 ∈ dom(v1) >= 3 ∈ dom(v3) and 1 ∈ dom(v1) >= 4 ∈ dom(v3) doesn't hold.
Therefore dom(v1) is revised in the function REVISE to dom(v1) = {}, as the inconsistent variable is deleted from the domain.
The domain of v1 is now empty. As the size of D_{v1} = 0 the while loop will break and return false, indicating that no solution can be found for the CSP Problem. This because arc-consistency can't be found for every arc in the queue after revising their domains.