SAARLAND UNIVERSITY ARTIFICIAL INTELLIGENCE
Prof. Jörg Hoffmann and Prof. Wolfgang Wahlster
Dr. Álvaro Torralba, Daniel Gnad, Marcel Steinmetz
Christian Bohnenberger, Cosmina Croitoru, Akram Elkorashy, Sophian Guidara, Daniel Heller, Björn Mathis, Lukas Schaal, Julia Wichlacz

Exercise Sheet 3.
Solutions due Tuesday, **May 24**, 16:00 – 16:15, in the lecture hall.[1]

Exercise 8. (2 Points)

It is possible to formulate the naïve backtracking approach (Chapter 7, slide 31) as a blind search in a classical search space. To show this, apply the following steps.

(a) Given an arbitrary constraint network $\gamma = (V, D, C)$, describe a state space such that some blind search algorithm behaves as backtracking over γ. Assume that the variable selection is based on a fixed variable ordering $v_1, v_2, \dots v_n$, so that at the ith recursion level of NaïveBacktracking, "select some variable v for which a is not defined" will select v_i .

Specify a state space $\Theta = (S, A, c, T, I, S^G)$ that corresponds to this. It is not necessary to provide a full formalization of all components. Rather, an informal description, i.e., one in natural language, is enough (as long as we can see that you get the ideas right). For example, the state set S can be described as the set of all partial assignments that are consistent with the constraints. (In the described setting, in a partial state only the first k variables are assigned some value; all variables v_j with $j > k$ are not assigned a value.)

In detail, do the following:

- Specify the action set A.
- Specify the cost function c.
- Specify the transition relation T.
- Specify the initial state I.
- Specify the goal states S^G.

(b) Which blind search algorithm in the search space defined in (a) behaves as naïve backtracking in the CSP problem? Explain why they behave the same.

[1]Solutions in paper form only, and solution submission only at the stated time at the stated place. At most 3 authors per solution. All authors must be in the same tutorial group. All sheets of your solution must be stapled together. At the top of the first sheet, you must write the names of the authors and the name of your tutor. Your solution must be placed into the correct box for your tutorial group. If you don't comply with these rules, 3 points will be subtracted from your score for this sheet.

Exercise 9. (2 Points)

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Constraints: $2|a - c| > 3$; $b^2 - 3d < 9$; $b + 3 < c$.

Run the AC-3(γ) algorithm, as specified in the lecture. Precisely, for each iteration of the while-loop, give the content of M at the start of the iteration, give the pair (u, v) removed from M, give the domain D_u of u after the call to Revise(γ, u, v), and give the pairs (w, u) added into M.

Note: Initialize M as a lexicographically ordered list (i.e., (a, b) would be before (a, c), both before (b, a) etc., if any of those exist). Furthermore, use M as a FIFO queue, i.e., always remove the element at the front and add new elements at the back.

(Solution)

Here is the required information for the iterations of the while-loop as executed. In this case, we always removed the element at the front first, and added new elements to the end.

1. M content: $\{(a, c), (b, c), (b, d), (c, a), (c, b), (d, b)\}$; pair selected: (a, c); No change, so nothing added to M.
2. M content: $\{(b, c), (b, d), (c, a), (c, b), (d, b)\}$; pair selected: (b, c); $D_b = \{1, 2, 3, 4, 5, 6\}$. Candidate (d, b) already present in M.
3. M content: $\{(b, d), (c, a), (c, b), (d, b)\}$; pair selected: (b, d); No change, so nothing added to M.
4. M content: $\{(c, a), (c, b), (d, b)\}$; pair selected: (c, a); No change, so nothing added to M.
5. M content: $\{(c, b), (d, b)\}$; pair selected: (c, b); $D_c = \{5, 6, 7, 8, 9, 10\}$. Candidate (a, c)added into M.
6. M content: $\{(d, b), (a, c)\}$; pair selected: (d, b); No change, so nothing added to M.
7. M content: $\{(a, c)\}$; pair selected: (a, c); No change, so nothing added to M.
8. M empty; return modified γ with reduced domains.

(/Solution)

Exercise 10. (2.5 Points)

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d, e, f, g\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Constraints: $a \leq c$; $b = c^2 - 2$; $|d - c| \leq 4$; $d < e - 6$; $|3e - f^2| \leq 6$; $|3g^2 - e| < 3$

Run the AcyclicCG(γ) algorithm, as specified in the lecture. Precisely execute its 4 steps as follows:

(a) Draw the constraint graph of γ. Pick a as the root and draw the directed tree obtained by step 1 (see Chapter 8 slides 36 and 37 for examples). Give a resulting variable order as can be obtained by step 2.

(b) List the calls to Revise($\gamma, v_{parent(i)}, v_i$) in the order executed by step 3, and for each of them give the resulting domain of $v_{parent(i)}$.

(c) For each recursive call of BacktrackingWithInference during step 4, give the domain D'_{v_i} of the selected variable v_i after Forward Checking, and give the value $d \in D'_{v_i}$ assigned to v_i.

Note: Step 4 runs BacktrackingWithInference with variable order $v_1, \ldots, v_n$. This means that, at the ith recursion level, "select some variable v for which a is not defined" will select v_i.

(Solution)

a) Possible variable ordering: a, c, b, d, e, f, g

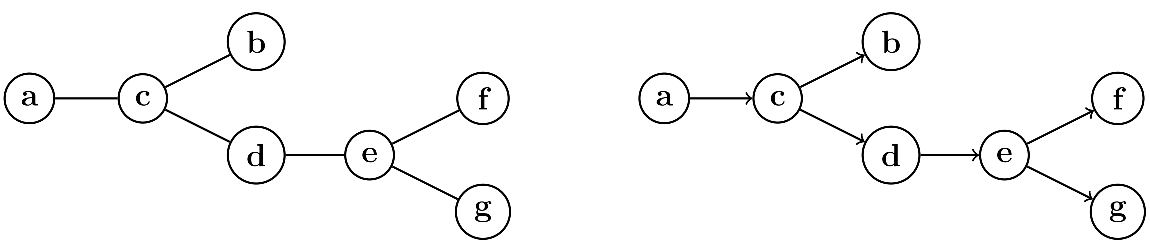


Figure 1: The constraint graph of this constraint network (left), and the directed tree (right).

b) The calls to $Revise(\gamma, v_{parent(i)}, v_i)$ and the resulting domains are:

- $Revise(\gamma, e, g)$: $D_e = \{1, 2, 3, 4, 5, 10\}$.
- $Revise(\gamma, e, f)$: $D_e = \{1, 2, 3, 4, 5, 10\}$.
- $Revise(\gamma, d, e)$: $D_d = \{1, 2, 3\}$.
- $Revise(\gamma, c, d)$: $D_c = \{1, 2, 3, 4, 5, 6, 7\}$.
- $Revise(\gamma, c, b)$: $D_c = \{2, 3\}$.
- $Revise(\gamma, a, c)$: $D_a = \{1, 2, 3\}$.

c) Possible D'_{v_i} and $d \in D'_{v_i}$ are:

- $D'_a = \{1, 2, 3\}$; $d = 1$.
- $D'_c = \{2, 3\}$; $d = 2$.
- $D'_b = \{2\}$; $d = 2$.
- $D'_d = \{1, 2, 3\}$; $d = 1$.
- $D'_e = \{10\}$; $d = 10$.
- $D'_f = \{5, 6\}$; $d = 5$.
- $D'_g = \{2\}$; $d = 2$.

(/Solution)

Exercise 11. (1.5 Points)

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d, e, f, g, h, i, j\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8\}$.
- Constraints: $a < c$; $a < b$; $b < d$; $c < d$; $d < e$; $d < f$; $e < g$; $f < g$; $g < h$; $g < i$; $h < i$; $h < j$; $i < j$;

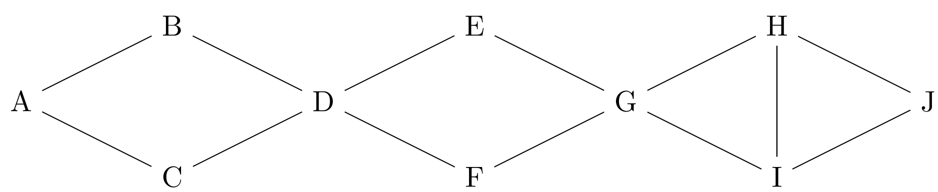
(a) Draw the constraint graph of γ.

(b) What is the optimal (minimal) cutset for γ, V_0?

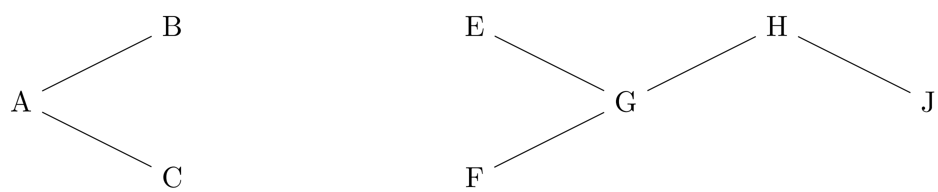
(c) If the CutsetConditioning algorithm from the lecture is called with such minimal cutset V_0, how many calls to AcyclicCG will be performed in the worst case? Justify your answer.

(Solution)

(a) The constraint graph of γ is:



(b) $V_0 = \{D, I\}$ ($V_0 = \{D, H\}$ is also a valid solution)



(c) Only one assignment of variables in V_0 results in a valid global assignment, so in the worst case we have $D_D \times D_I = 8 \times 8 = 64$ calls to AcyclicCG before finding it.

(/Solution)

Exercise 12. (2 Points)

Assume a solvable constraint network γ, and assume that the AcyclicCG(γ) algorithm, as specified on Chapter 8 slide 35, reaches step 4 (i.e., AcyclicCG(γ) does not return "inconsistent" at some point during step 3). Assume that BacktrackingWithInference, as specified on Chapter 8 slide 14, uses the variable order $v_1, \dots, v_n$, i.e., at the i-th recursion level, "select some variable v for which a is not defined" will select v_i.

For each variable v_i, denote by $D^3_{v_i}$ the domain of v_i after the completion of AcyclicCG(γ) step 3 (i.e., the domain as in the input to backtracking in step 4). Denote by D_{v_i} the domain of v_i at the beginning of the call of BacktrackingWithInference at the ith recursion level, and denote by D'_{v_i} the domain of v_i after forward checking, i.e., after "Inference(γ', a)" has been run.

Prove that: (i) $a \cup \{v_i = d\}$ is consistent for every $d \in D'_{v_i}$, and (ii) $D'_{v_i} \neq \emptyset$.

(Note: (i) and (ii) together imply that step 4 finds a solution without ever having to backtrack.)