Technische Universität München
Department of Computer Science
Prof. Michael Beetz, Ph.D.
Daniel Nyga, M.Sc.

WS 2011/2012
Solution for Worksheet 4
22.11.2011

Exercises for the Lecture Techniques in Artificial Intelligence

Heuristic Search and CSPs

1) *Properties of Heuristics*

(a) Prove the following proposition: If a heuristic function h never over-estimates the true cost by more than c, then A^* using h finds a path with cost that is at most c higher than the optimal path cost ("c-optimal path").

Solution

It holds for all nodes n that $h(n) \leq h^*(n) + c$ (where $h^*(n)$ is the true cost of an optimal path from n to the goal node.)

Let C^* the the cost of the optimal path and let G_2 be an arbitrary node with $g(G_2) > C^* + c$.

To be proved: A^* finds a path with cost $\leq C^* + c$ before a node G_2 is visited.

Let n be an arbitrary node on the optimal path to the goal node (including that goal node). It holds that:

$$\begin{aligned} f(n) &= g(n) + h(n) \\ &\leq g(n) + h^*(n) + c \\ &= C^* + c \\ &< g(G_2) \leq f(G_2) \end{aligned}$$

Since the estimated cost f for all nodes on the optimal path are smaller than the estimated cost f of G_2, A^* will never visit a node like G_2, as a c-optimal solution wil be found before. Thus, a path that is worse than c-optimal will not be returned, as such a path must contain a node like G_2.

(b) Prove: Every consistent heuristic function is admissible

A heuristic function h is *consistent* (for a state space S), if for all $n \in S$, all successors $n' \in succ(n)$ and all actions a leading from n zu n':

$$h(n) \leq c(n, a, n') + h(n')$$

Where c is the step cost. We assume that heuristic functions return 0 for the goal node.

Solution

Remember: A Heuristic is *admissible* if for all nodes n: $h(n) \leq h^*(n)$

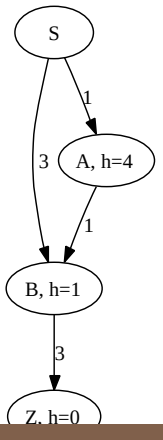
Proof by induction over the number k of steps along the shortest path of a node n to a goal node. (Remark: It suffices to consider the shortest path, as it yields an upper bound for $h(n)$, which implies admissibility. Other paths might yield even smaller bounds.)

- For $k = 1$: Let n' be a goal node. It holds that:
  $h(n) \leq c(n, a, n') + h(n') = c(n, a, n')$
- Inductive step: $k \to k + 1$.
  Let n be $k + 1$ steps away from a goal node and n' be k steps away from a goal node.
  $h(n) \leq c(n, a, n') + h(n') \overset{(IA)}{\leq} c(n, a, n') + h^*(n') = h^*(n)$

(c) Show that the inverse is not true by constructing an admissible Heuristic that is not consistent

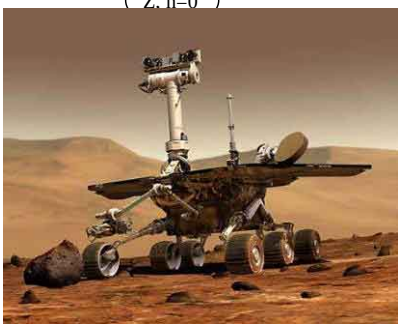
Solution

The Heuristics in this figure is admissible, since it never over-estimates the true cost to the goal node Z. However, it is not consistent, since the heuristic value of node A does not fulfill the required condition: $h(A) \not\leq c(A, a, B) + h(B) = 2$.



Remark: A^*-Search requires a *consistent* Heuristics when avoiding to re-visit nodes (by maintaining a list of visited nodes; see GraphSearch), in order to ensure optimality; admissibility is no longer sufficient in this case. Using GraphSearch, A^* would find the path SBZ in this graph, as B is visited before A (due to the inconsistent Heuristics), and re-visiting A is suppressed. The shortest path is $SABZ$. A consistent Heuristics ensures that the shortest path to a node is visited first by A^*.

2) *Search on Mars*



A mars robot shall leave its landing base in order to collect rock samples at three places and then return to the base afterwards. Assume that the robot has a navigation module that can lead it directly to the points of interest (rock1, rock2, rock3, lander), i.e. it has the actions go-to-lander, go-to-communications-location, go-to-rock-1, go-to-rock-2, and go-to-rock-3. The times that are necessary to go from one location ot another are known (every connection). The goal is to find a sequence of actions that fulfills the task (in the shortest time possible, if the search algorithm allows for that).

The symmetric path costs are listed in the following table:

	lander	comm-location	rock1	rock2	rock3
lander	0	2	4	2	1
comm-location	2	0	3	3	3
rock1	4	3	0	4	5
rock2	2	3	4	0	2
They	1	3	5	2	0

(a) Phrase the problem as a search problem by specifying a search space, a starting state, a cost function and a goal test.

(b) In the lecture we presented a general search algorithm, where the behaviour is determined by a function Queuing-Fn. Apply this search algorithm on the previously phrased search problem, using the following methods:

i. insert n at front

ii. insert n at back

iii. insert n into the list which is sorted by $f(n) = h(n)$.

iv. insert n into the list which is sorted by $f(n) = g(n) + h(n)$.

Where $g(n)$ denotes the path cost from the start node to n and $h(n)$ denotes a heuristic estimation for the cost to the goal. We use the heuristic $h(n)$=„number of not yet collected rocks“.

At (iii) and (iv) the sorting can be can be ambiguous; in such a case, the nodes with higher depth should be expanded first. Further, the actions should be considered in the following order: go-to-rock-1, go-to-rock-2, go-to-rock-3, go-to-comm-location, go-to-lander.

Now, answer the following questions for all four search algorithms:

- What is the common name for this search algorithm?
- How many nodes were expanded? (if necessary, use assignment 2c)
- How high is the path cost for the found solution? (if necessary, use assignment 2c)

- Does the algorithm find the optimal solution (in this example / in general)?

(c) *Programming Assignment*

Implement the search algorithm as a refinement of an abstract search by specializing the function QUEUING-FN, and solve the given search problem in order to answer above questions. Strive for a compact and object-oriented implementation.

(d) Now, assume additionally that the robot has to appear every day at 15:00 at a special location for communicating to earth – also the day it is doing the rock sampling. A plan that misses the communication rendevous is invalid.

How would you change the problem description in order to fulfill this additional requirement? Assume that there is an action 'communicate', which can only be executed if the robot is at the location for communication, and which – when executed – waits until 15:00 and then exchanges information with earth for an hour.

Solution

(a)
- search space: current-location × have-rock-1? × have-rock-2? × have-rock-3?
- starting state: (lander,0,0,0)
- successor function: Let (a, r_1, r_2, r_3) be the starting state; on the execution of go-to-x, substitue x/a and for x = rock-i substitute $1/r_i$. (The collection of the rock samples shall be included in the go-to actions).
- goal test: (lander,1,1,1)
- cost function: Step cost according to problem description

(b)

Algorithm	Name	exp. nodes[1]	path cost	optimal here?	optimal in general?
(i)	depth-first search	8 (∞)	31	nein	nein
(ii)	breath-first search	27 (92)	11	ja	nein
(iii)	greedy search	7 (7)	11	ja	nein
(iv)	A* search	27 (59)	11	ja	ja[2]

[1] The first value is obtained using GRAPH-SEARCH, the value in parentheses is without avoidance of repeated states.

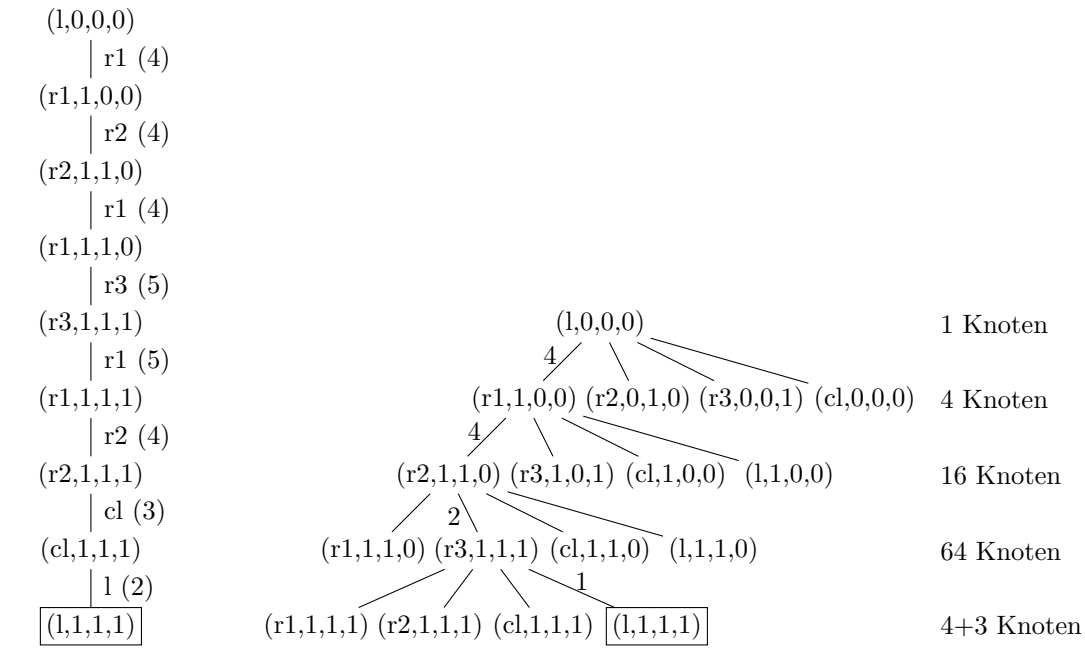
[2] assuming a consistent / admissible heuristic.

Parts of the search trees are depicted in fig. 1.

(c) Please refer to the excercise home page: You can follow the steps using the script marssearch.py.

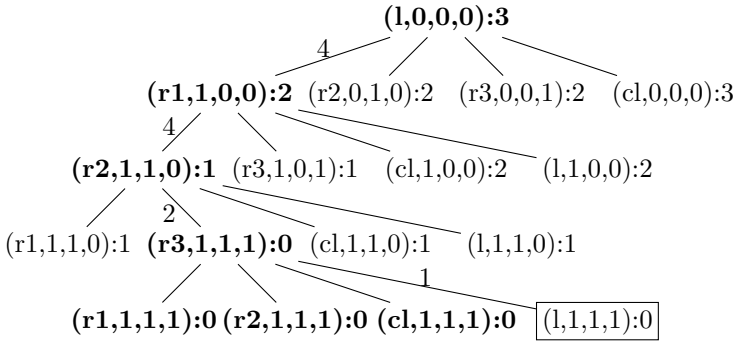
(d) The new state space must include the time as well as a bit which expresses whether a communication rendevous was missed. This bit shall be set by the successor function whenever an action passes the 15:00 time and it is not the communication action. The costs listed in the table are the times that the navigation action requires. They are used for updating the time component of the current state.

Thus the new state is: current-location × have-rock-1? × have-rock-2? × have-rock-3? × time × missed-com-requirement?. Accordingly, the new goal-test is (lander, 1, 1, 1, t, 0) mit t > 15:00 Uhr.

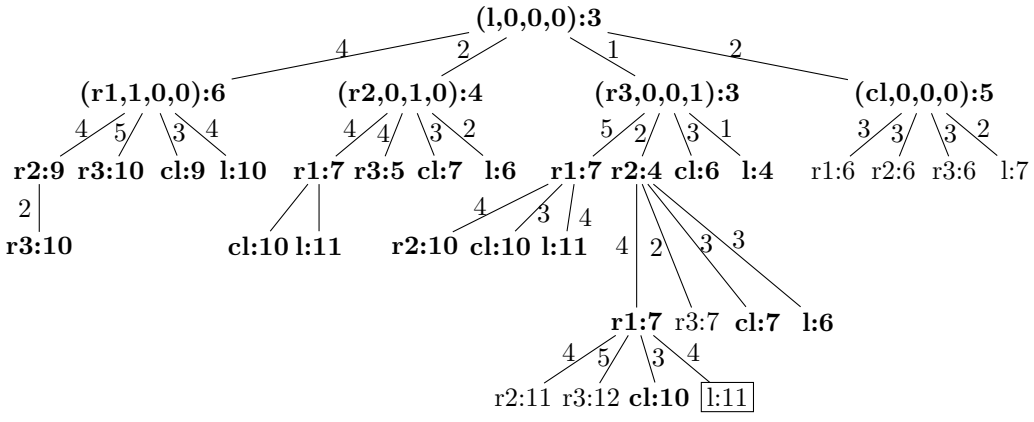


(a) depth-first search

(b) breath-first search



(c) greedy search



(d) A* search

Abbildung 1: search trees. For informed search algorithms, the expanded nodes are marked in bold.

3) *CSP: kryptoarithmetic riddle*

Solve the riddle you know from the lectures by formulating it as a CSP and solving it using forward test and backtracking, using the minimal-remaining-values heuristic.

	T	W	O
+	T	W	O
F	O	U	R

In the adjacent calculation, the letters are to be replaced by digits from 0 to 9, so that a correct calculation in the decimal system appears. The same letter stands for same digits, different letters cannot represent the same digit. F cannot be 0.

Solution

- Variables: F, O, R, T, U, W, Helper Variables X_1, X_2, X_3
- Domains: $D_{F,X_3} = \{1\}$, $D_{X_1,X_2} = \{0,1\}$, $D_{O,R,T,U,W} = \{0,\ldots,9\}$
- Constraints:
  - Variables F, O, R, T, U, W have different value
  - $O + O = R + 10 \cdot X_1$
  - $W + W + X_1 = U + 10 \cdot X_2$
  - $T + T + X_2 = O + 10 \cdot X_3$
  - $X_3 = F > 0$

In the following table, all possible assignments of values to variables are denoted in the first row. The assignments made in the respective step are printed in bold characters. When backtracking is necessary, a row is added that denotes all possible values at this state

Note: The solution presented here is shorter than one that would be found by a program using the given constraints. We assume knowledge of general mathmatical relationships that can be calculated easily, which leads to a more rapid solution of the riddle.
We re-formulate the constraints:

- $X_3 = F = 1$
- $R = 2O - 10X_1 = 2(O - 5X_1)$
- $U = 2W + X_1 - 10X_2$
- $O = 2T + X_2 - 10X_3 \qquad O = \frac{R+10X_1}{2}$
- $W = \frac{U+10X_2-X_1}{2}$
- $T = \frac{O+10X_3-X_2}{2} = \frac{O+10-X_2}{2} = 5 + \frac{O-X_2}{2} \quad \Rightarrow T \geq 5$

In row 3 3, the actual algorithm starts, assigning in each step and using the constraints that incorporate the assigned variable to reduce the possible values for the other variables present in the respective constraint (forward check). The MRV-heuristic makes sure that the variable with the smallest amount of possible values remaining gets assigned first.

	F	O	R	T	U	W	X_1	X_2	X_3	
1	*1*	0–9	0–9	0–9	0–9	0–9	0,1	0,1	*1*	Previous knowledge
2		0,2–9	0,2,4,6,8	5,6,7,8,9	0,2–9	0,2–9				T $\geq$ 5, 2|R, Alldiff
3		0,2,3,4	0,4,6,8		0,2,4,6,8		**0**			$X_1 = 0 \Rightarrow$ O $\leq$ 4, R = 2·O
4		0,2,4			0,4,6,8	0,2,3,4		**0**		$X_2 = 0 \Rightarrow$ W $\leq$ 4, U = 2·W, O = 2·T-10
5		**0**	—	5	4,6,8	2,3,4				No possible values for R remain!
6	*1*	2,4	0,4,6,8	5,6,7,8,9	0,4,6,8	0,2,3,4	*0*	*0*	*1*	Backtracking to row 4
7		**2**	4	6	0,4,6,8	0,3,4				$R = 2O, T = 5 + O/2, U = 2W, W = U/2$
8			**4**		0,6,8	0,3				
9				**6**	0,8	0,3				
10					**0**	—				No possible values for W remain!
11	*1*	*2*	*4*	*6*	8	0,3	*0*	*0*	*1*	Backtracking to row 9
12					**8**	—				No possible values for R remain!
13	*1*	4	0,2,4,6,8	5,6,7,8,9	0,4,6,8	0,2,3,4	*0*	*0*	*1*	Backtracking to row 6
14		**4**	8	7	0,6,8	0,2,3				
15			**8**		0,6					
16				**7**						
17					**0**	—				No possible values for R remain!
18	*1*	*4*	*8*	*7*	6	0,2,3	*0*	*0*	*1*	Backtracking to row 16
19					**6**	3				
20						**3**				
	1	4	8	7	6	3	0	0	1	result