**University of Paderborn**
**Planning and Heuristic Search**
**WS 2021/22**
**Dr. Th. Lettmann**

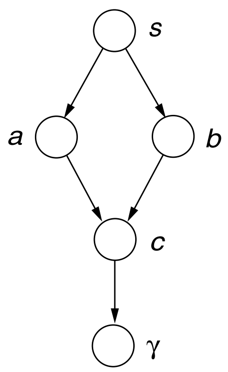
November 25, 2021

# Class Exercise 5

**Exercise 1 :** Suboptimum solution paths

For solving optimization problems with A*, the evaluation function $f$ has to return optimistic estimates. There are examples showing that a non-optimum solution path is returned if $f$ overestimates (at least sometimes).

For the following graph $G$ with start node $s$ and goal node $\gamma$, give edge cost values and $h$-values such that A* will terminate suboptimally.

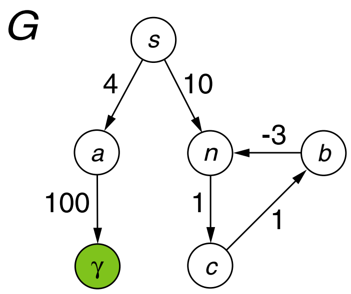



| $n$ | $n'$ | $c(n, n')$ |
|---|---|---|
| $s$ | $a$ | |
| $s$ | $b$ | |
| $a$ | $c$ | |
| $b$ | $c$ | |
| $c$ | $\gamma$ | |

| $n$ | $h(n)$ |
|---|---|
| $s$ | |
| $a$ | |
| $b$ | |
| $c$ | |
| $\gamma$ | 0 |

**Exercise 2 :** Negative Weight Cycles

Consider the following search space graph $G$:




For the application of the A* algorithm we use the edge cost values from the graphic and some heuristic function $h$.

(a) If $h$ returns 0 for all nodes, a problem arises when processing $G$ with A*. Describe the problem.

(b) Is there a heuristic function $h$, such that the problem can be avoided?

(c) Can you characterize the class of heuristic functions $h$ for which this problem arises?

(d) Can you characterize the class of heuristic functions $h$ for which this problem does not arise?

(e) Are there heuristic functions $h$ that do not belong to any of the two previous classes?

(f) Is it necessary to choose the value $h(\gamma)$ in a specific way?

**Exercise 3 :** Termination of BF*

Let be given a finite search space graph $G$ with almost all properties in $\mathit{Prop}_{A*}(G)$ except that path cost is computed as maximum of edge cost values and heuristic value at the tip node.

Does BF* terminate on $G$ when using this evaluation function?

**Exercise 4 :** Existence of optimum paths

The following result holds:

Let $G$ be a search space graph with $\mathit{Prop}_{A*}(G)$. If there is a path in $G$ from node $n$ to node $n'$ in $G$, then there is also a cheapest path from $n$ to $n'$ in $G$.

Which step in the proof is no longer valid if we skip the precondition of local finiteness for $G$.

**Exercise 5 :** Cost Modeling in BF

Consider the following problem:

**The suspension bridge problem**

It is pitch black and it will stay that way – at least until you have solved the task. Father, mother, son and daughter are standing in front of a suspension bridge allof them have to cross but on which only two people can walk at a time. The father needs 25 minutes, the mother 20, the daughter 10 minutes and the son 5 minutes to cross the bridge.

The family has only one flashlight with a burning time of exactly 60 minutes! Without lighting nobody dares to go on the suspension bridge. So two people must always cross the bridge together; the longer time counts. Then someone has to return with the flashlight. Also this time counts.

**Question:**
How do the four of them have to proceed so that everyone can cross the bridge with the lamp on?

The person with the flashlight must always go with you to the end of the bridge. The light range of the flashlight is not taken into account.

No tricks are allowed, such as son carries father, or the flashlight is thrown over the bridge, or the battery recovers when the flashlight is switched off.

(a) Is there a solution for this problem?
(b) Describe this search in our formalism. It it an optimization task or a constraint satisfaction task?
(c) Is it an informed search? If yes, what is the information used? In this case give a (recursive) definition of a cost function.

**Exercise 6 :** Binary Search as Informed Search

Binary search is a fast search algorithm with run-time complexity of $\mathcal{O}(\log n)$. This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form and it should not contain duplicates. The binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of the item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the sub-array reduces to zero.

(a) Can we describe this search in our formalism?
(b) Is it an informed search? Is yes, what is the information used?