Lecture *Hierarchical Planning*

Chapter: *Search*

Dr. Pascal Bercher

Institute of Artificial Intelligence,
Ulm University, Germany

Winter Term 2018/2019

(Compiled on: September 25, 2021)

ulm university universität ulm

Overview:

BFS Algorithm

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier)  /* chooses the shallowest node in frontier */
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored then
        if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
        frontier ← INSERT(child, frontier)
```

Breadth-first search on a graph.

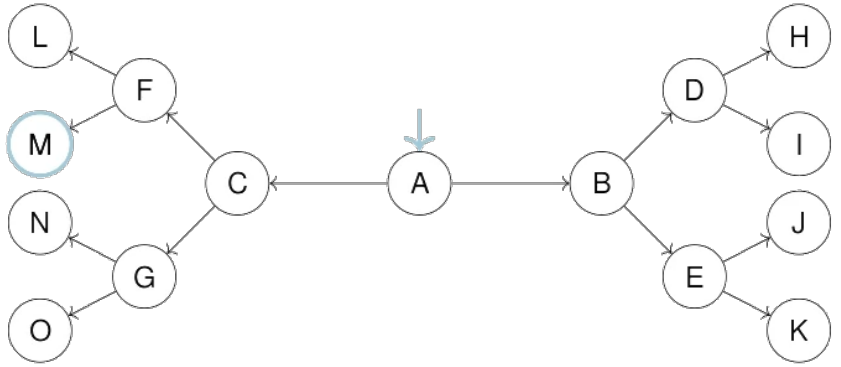
copyright: see slide 13[1] (modified)

Question:
In which way does this algorithm differ (e.g., is more precise) than the generic graph-search algorithm?

- Here, the goal test is done *before insertion into the fringe* (“early goal test”), not after selection from the fringe.
- The fringe is not “generic”, but implemented as FIFO.

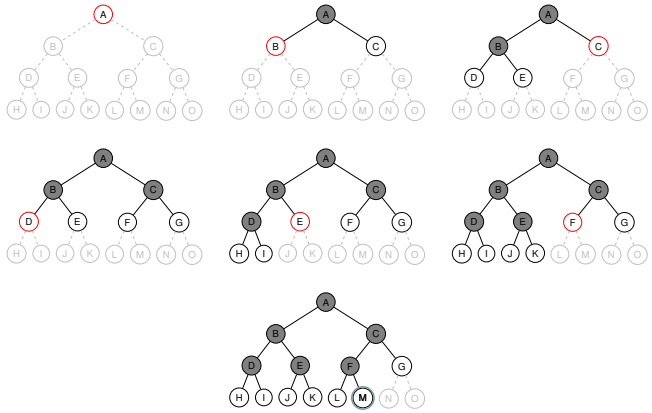
Example of BFS

Transition system:



Example of BFS, cont'd

At one glance:



Properties of BFS

Optimality Depends on the costs:

- Without action costs or with unit costs:
  - Obviously (both for tree- and for graph search)
  - Even optimal with an "early goal test"
- With action costs: No

Completeness Depends on properties of the transition system:

- Finite: Yes
- Infinite: Only with finite branching factor.

Correctness Yes

Space $O(b^d)$

Time Just as space. (If the fringe is implemented as a priority queue rather than as queue (FIFO), the queue sorting overhead needs to be added. This does not change the asymptotic runtime, however.)

DFS Algorithm

function BREADTH-FIRST-SEARCH(*problem*) **returns** a solution, or failure

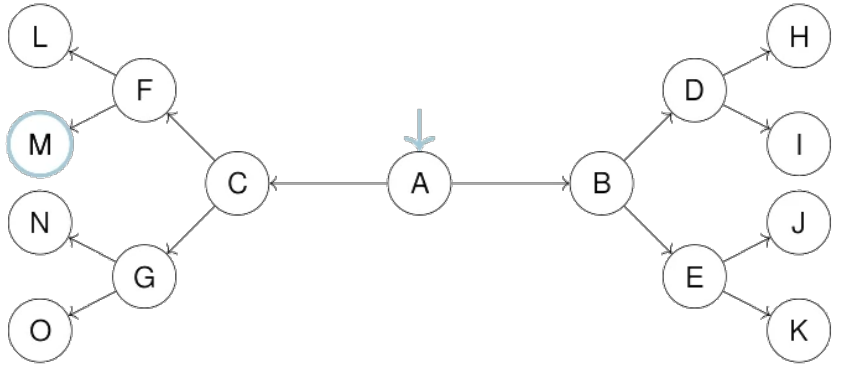
node ← a node with STATE = *problem*.INITIAL-STATE, PATH-COST = 0
if *problem*.GOAL-TEST(*node*.STATE) **then return** SOLUTION(*node*)
frontier ← a FIFO queue with *node* as the only element
explored ← an empty set
loop do
  if EMPTY?(*frontier*) **then return** failure
  node ← POP(*frontier*) /* chooses the shallowest node in *frontier* */
  add *node*.STATE to *explored*
  for each *action* **in** *problem*.ACTIONS(*node*.STATE) **do**
    child ← CHILD-NODE(*problem*, *node*, *action*)
    if *child*.STATE is not in *explored* **then**
      if *problem*.GOAL-TEST(*child*.STATE) **then return** SOLUTION(*child*)
      frontier ← INSERT(*child*, *frontier*)

Breadth-first search on a graph.

Just replace the FIFO fringe by a LIFO fringe.

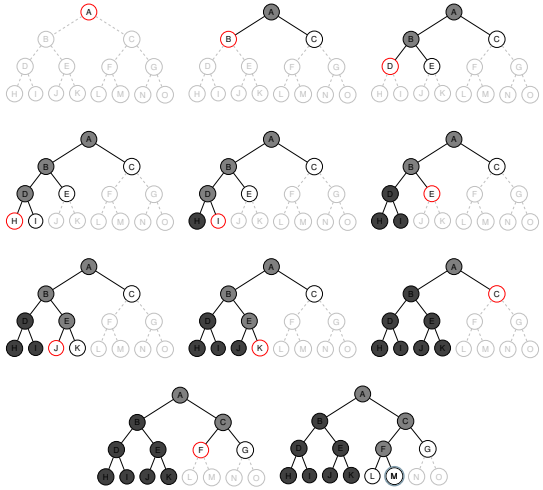
Example of DFS

Transition system:



Example of DFS, cont'd

At one glance:



Properties of DFS

Optimality No

Completeness Depends on duplicate management:

- Tree search: only if the transition system is acyclic
- Graph search: only the weakest form of completeness (and this only if the transition system is acyclic)

Correctness Yes

Space $O(b \cdot m)$ (If you only store the fringe.)

Time $O(b^m)$ (If the fringe is implemented as a priority queue rather than as stack (LIFO), the queue sorting overhead needs to be added. This does not change the asymptotic runtime, however.)

Algorithm and Remarks

- Implements the generic tree-search or graph-search algorithms.
- Implements fringe as priority queue that selects a node with minimal cost value $g(n)$.
- UCS can be a regarded a modification of BFS by expanding the cheapest rather than the shallowest node. *Note:* In contrast to BFS, the early goal test is not allowed here! (Why? Example?)
- UCS can also be regarded a special case of A^* (covered later this chapter), where no heuristic is used.
- UCS is equivalent to Dijkstra's algorithm.

Properties of Uniform Cost

Optimality Yes

Completeness Depends on duplicate management and action costs:

- Tree search: If all action costs are strictly larger than 0.
- Graph search: Yes (except for the strongest form of completeness)
- Again, for infinite transition systems the situation is more complicated.

Correctness Yes

Space $O(b^{1+\lfloor g^*/\varepsilon \rfloor})$, where g^* denotes the cost of an optimal solution, and ε the (positive) cost of the cheapest action.

Time Similar to space.

Copyright Notes and Licenses

[1] Title: *Artificial Intelligence: A Modern Approach (Third Edition)*
Url: `https://aima.cs.berkeley.edu/`
Authors: *Stuart Russel* and *Peter Norvig*