Basics of AI and Machine Learning

State-Space Search: Data Structures for Search Algorithms

Jendrik Seipp

Linköping University

Slides modified from Basel AI group, with permission

State-Space Search: Overview

Chapter overview: state-space search

- Foundations
- Basic Algorithms
  - Data Structures for Search Algorithms
  - Tree Search and Graph Search
  - Breadth-first Search
  - Uniform Cost Search
  - Depth-first Search
- Heuristic Algorithms

Introduction

Search Algorithms

- We now move to search algorithms.
- As everywhere in computer science, suitable data structures are a key to good performance.
  - ⇝ common operations must be fast
- Well-implemented search algorithms process up to ∼30,000,000 states/second on a single CPU core.
  - ⇝ bonus materials (Burns et al. paper)

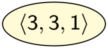
this chapter: some fundamental data structures for search

Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.

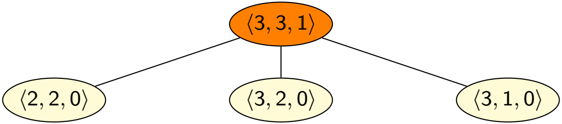
Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



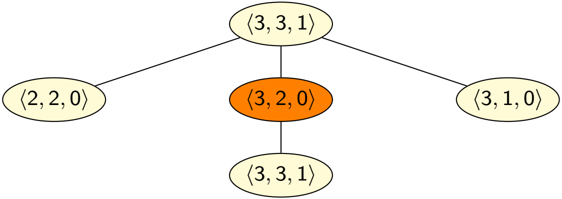
Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



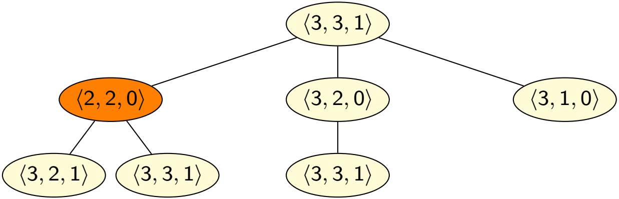
Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



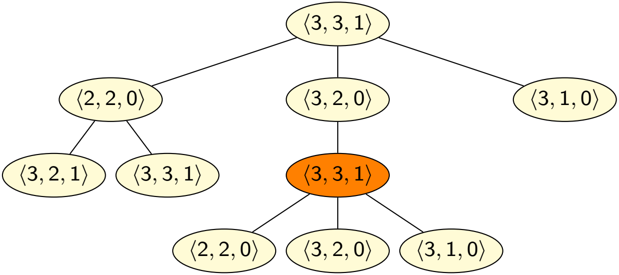
Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



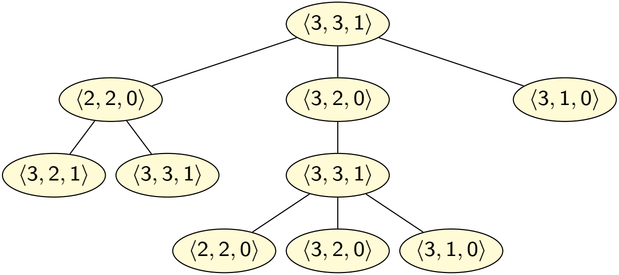
Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



Example: Search Algorithm

- Starting with initial state,
- repeatedly expand a state by generating its successors.
- Stop when a goal state is expanded
- or all reachable states have been considered.



…and so on (expansion order depends on search algorithm used)

Fundamental Data Structures for Search

We consider three abstract data structures for search:

- search node: stores a state that has been reached, how it was reached, and at which cost
  - ⇝ nodes of the example search tree
- open list: efficiently organizes leaves of search tree
  - ⇝ set of leaves of example search tree
- closed list: remembers expanded states to avoid duplicated expansions of the same state
  - ⇝ inner nodes of a search tree

Not all algorithms use all three data structures,
and they are sometimes implicit (e.g., in the CPU stack)

Search Nodes

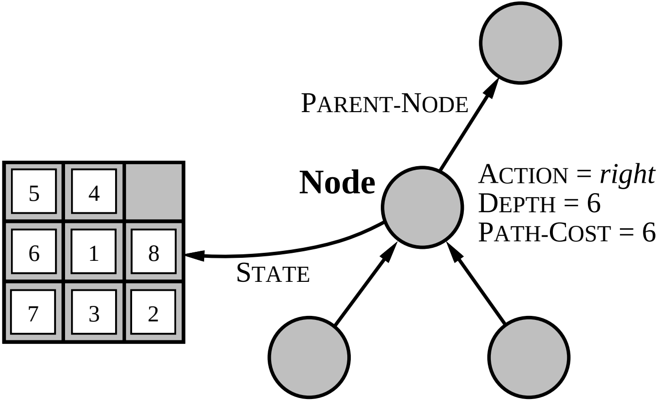
Search Nodes

Search Node

A search node (node for short) stores a state
that has been reached, how it was reached, and at which cost.

Collectively they form the so-called search tree.

Node in a Search Tree



Open Lists

Open Lists

Open List

The open list (also: frontier) organizes the leaves of a search tree.

It must support two operations efficiently:

- determine and remove the next node to expand
- insert a new node that is a candidate node for expansion

Remark: despite the name, it is usually a very bad idea to implement open lists as simple lists.

Closed Lists

Closed Lists

Closed List

The closed list remembers expanded states
to avoid duplicated expansions of the same state.

It must support two operations efficiently:

- insert a node whose state is not yet in the closed list
- test if a node with a given state is in the closed list;
  if yes, return it

Remark: despite the name, it is usually a very bad idea to implement closed lists as simple lists: membership test for lists needs linear time.

Summary

Summary

- search node:
  represents states reached during search
  and associated information
- node expansion:
  generate successor nodes of a node by applying all actions
  applicable in the state belonging to the node
- open list or frontier:
  set of nodes that are currently candidates for expansion
- closed list:
  set of already expanded nodes (and their states)