

Foundations of Artificial Intelligence

5. State-Space Search: State Spaces

Malte Helmert

University of Basel

February 25, 2019

Foundations of Artificial Intelligence

February 25, 2019 — 5. State-Space Search: State Spaces

5.1 State-Space Search Problems

5.2 Formalization

5.3 State-Space Search

5.4 Summary

5.1 State-Space Search Problems

Classical State-Space Search Problems Informally

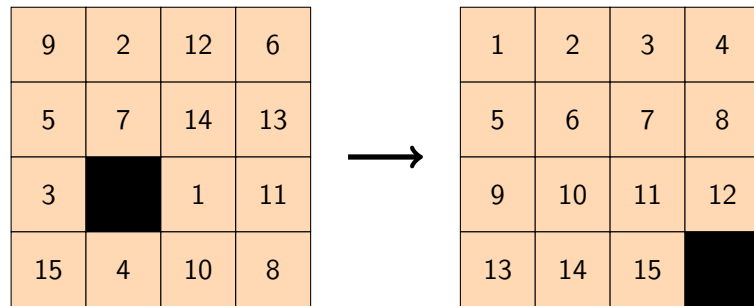
(Classical) state-space search problems are among the “simplest” and most important classes of AI problems.

objective of the agent:

- ▶ from a given initial state
- ▶ apply a sequence of actions
- ▶ in order to reach a goal state

performance measure: minimize total action cost

Motivating Example: 15-Puzzle



Classical Assumptions

“classical” assumptions:

- ▶ no other agents in the environment (**single-agent**)
 - ▶ always knows state of the world (**fully observable**)
 - ▶ state only changed by the agent (**static**)
 - ▶ finite number of states/actions (in particular **discrete**)
 - ▶ actions have **deterministic** effect on the state
- ↪ can all be generalized (but not in this part of the course)

For simplicity, we omit “classical” in the following.

Classification

Classification:

State-Space Search environment:

- ▶ **static** vs. dynamic
- ▶ **deterministic** vs. non-deterministic vs. stochastic
- ▶ **fully** vs. partially vs. not **observable**
- ▶ **discrete** vs. continuous
- ▶ **single-agent** vs. multi-agent

problem solving method:

- ▶ **problem-specific** vs. general vs. learning

Search Problem Examples

- ▶ **toy problems**: combinatorial puzzles (Rubik’s Cube, 15-puzzle, towers of Hanoi, ...)
- ▶ **scheduling** of events, flights, manufacturing tasks
- ▶ **query optimization** in databases
- ▶ behavior of **NPCs** in computer games
- ▶ **code optimization** in compilers
- ▶ **verification** of soft- and hardware
- ▶ **sequence alignment** in bioinformatics
- ▶ **route planning** (e.g., Google Maps)
- ▶ ...

thousands of practical examples

State-Space Search: Overview

Chapter overview: state-space search

- ▶ 5.–7. Foundations
 - ▶ 5. State Spaces
 - ▶ 6. Representation of State Spaces
 - ▶ 7. Examples of State Spaces
- ▶ 8.–12. Basic Algorithms
- ▶ 13.–19. Heuristic Algorithms

5.2 Formalization

Formalization

preliminary remarks:

- ▶ to cleanly study search problems we need a **formal model**
- ▶ fundamental concept: **state spaces**
- ▶ state spaces are (labeled, directed) **graphs**
- ▶ **paths** to goal states represent **solutions**
- ▶ **shortest paths** correspond to **optimal solutions**

State Spaces

Definition (state space)

A **state space** or **transition system** is a 6-tuple

$S = \langle S, A, cost, T, s_0, S_* \rangle$ with

- ▶ S : finite set of **states**
- ▶ A : finite set of **actions**
- ▶ $cost : A \rightarrow \mathbb{R}_0^+$ **action costs**
- ▶ $T \subseteq S \times A \times S$ **transition relation**; **deterministic in $\langle s, a \rangle$**
(see next slide)
- ▶ $s_0 \in S$ **initial state**
- ▶ $S_* \subseteq S$ set of **goal states**

German: Zustandsraum, Transitionssystem, Zustände, Aktionen, Aktionskosten, Transitions-/Übergangsrelation, deterministisch, Anfangszustand, Zielzustände

State Spaces: Transitions, Determinism

Definition (transition, deterministic)

Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_* \rangle$ be a state space.

The triples $\langle s, a, s' \rangle \in T$ are called **(state) transitions**.

We say \mathcal{S} **has the transition** $\langle s, a, s' \rangle$ if $\langle s, a, s' \rangle \in T$.

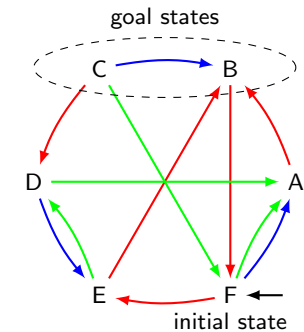
We write this as $s \xrightarrow{a} s'$, or $s \rightarrow s'$ when a does not matter.

Transitions are **deterministic** in $\langle s, a \rangle$: it is forbidden to have both $s \xrightarrow{a} s_1$ and $s \xrightarrow{a} s_2$ with $s_1 \neq s_2$.

State Spaces: Example

State spaces are often depicted as **directed graphs**.

- ▶ **states**: graph vertices
- ▶ **transitions**: labeled arcs (here: colors instead of labels)
- ▶ **initial state**: incoming arrow
- ▶ **goal states**: marked (here: by the dashed ellipse)
- ▶ **actions**: the arc labels
- ▶ **action costs**: described separately (or implicitly = 1)



State Spaces: Terminology

We use common terminology from graph theory.

Definition (predecessor, successor, applicable action)

Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_* \rangle$ be a state space.

Let $s, s' \in S$ be states with $s \rightarrow s'$.

- ▶ s is a **predecessor** of s'
- ▶ s' is a **successor** of s

If $s \xrightarrow{a} s'$, then action a is **applicable** in s .

German: Vorgänger, Nachfolger, anwendbar

State Spaces: Terminology

We use common terminology from graph theory.

Definition (path)

Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_* \rangle$ be a state space.

Let $s^{(0)}, \dots, s^{(n)} \in S$ be states and $\pi_1, \dots, \pi_n \in A$ be actions such that $s^{(0)} \xrightarrow{\pi_1} s^{(1)}, \dots, s^{(n-1)} \xrightarrow{\pi_n} s^{(n)}$.

- ▶ $\pi = \langle \pi_1, \dots, \pi_n \rangle$ is a **path** from $s^{(0)}$ to $s^{(n)}$
- ▶ **length** of π : $|\pi| = n$
- ▶ **cost** of π : $cost(\pi) = \sum_{i=1}^n cost(\pi_i)$

German: Pfad, Länge, Kosten

- ▶ paths may have length 0
- ▶ sometimes "path" is used for state sequence $\langle s^{(0)}, \dots, s^{(n)} \rangle$ or sequence $\langle s^{(0)}, \pi_1, s^{(1)}, \dots, s^{(n-1)}, \pi_n, s^{(n)} \rangle$

State Spaces: Terminology

more terminology:

Definition (reachable, solution, optimal)

Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_* \rangle$ be a state space.

- ▶ state s is **reachable** if a path from s_0 to s exists
- ▶ paths from $s \in S$ to some state $s_* \in S_*$ are **solutions for/from s**
- ▶ solutions for s_0 are called **solutions for \mathcal{S}**
- ▶ **optimal solutions** (for s) have minimal costs among all solutions (for s)

German: erreichbar, Lösung von/für s , optimale Lösung

5.3 State-Space Search

State-Space Search

State-Space Search

State-space search is the algorithmic problem of finding solutions in state spaces or proving that no solution exists.

In **optimal** state-space search, only optimal solutions may be returned.

German: Zustandsraumsuche, optimale Zustandsraumsuche

Learning Objectives for State-Space Search

Learning Objectives for the Topic of State-Space Search

- ▶ **understanding state-space search:**
What is the problem and how can we formalize it?
- ▶ **evaluate search algorithms:**
completeness, optimality, time/space complexity
- ▶ **get to know search algorithms:**
uninformed vs. informed; tree and graph search
- ▶ **evaluate heuristics for search algorithms:**
goal-awareness, safety, admissibility, consistency
- ▶ **efficient implementation** of search algorithms
- ▶ **experimental evaluation** of search algorithms
- ▶ **design and comparison of heuristics** for search algorithms

5.4 Summary

Summary

- ▶ **classical state-space search problems:** find action sequence from initial state to a goal state
- ▶ **performance measure:** sum of action costs
- ▶ formalization via **state spaces:**
 - ▶ **states, actions, action costs, transitions, initial state, goal states**
- ▶ terminology for transitions, paths, solutions
- ▶ definition of (optimal) state-space search