Chapter S:I

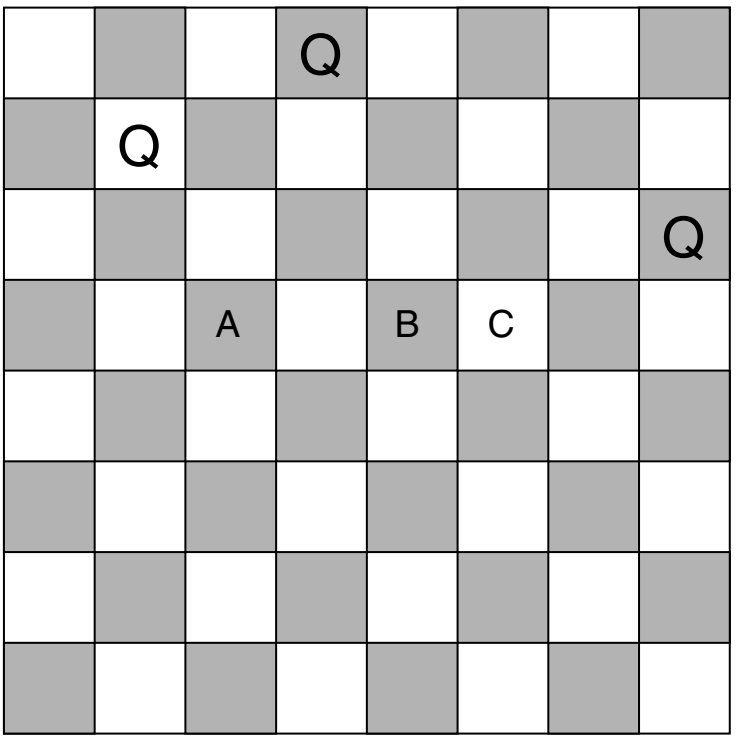
I. Introduction

- Examples for Search Problems
- Search Problem Abstraction

Examples for Search Problems

8-Queens Problem

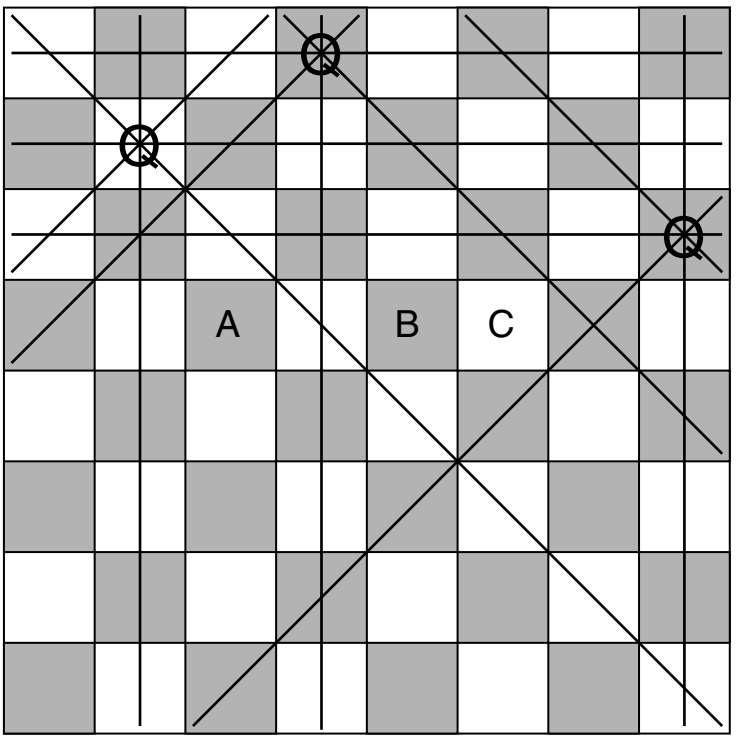
Incremental placement of queens:



Examples for Search Problems

8-Queens Problem (continued)

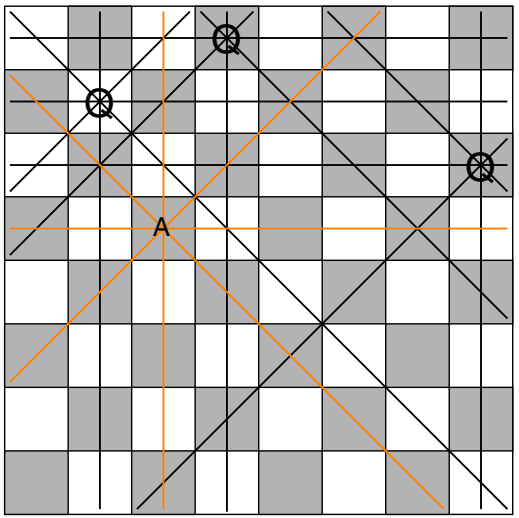
Analysis of the current board:

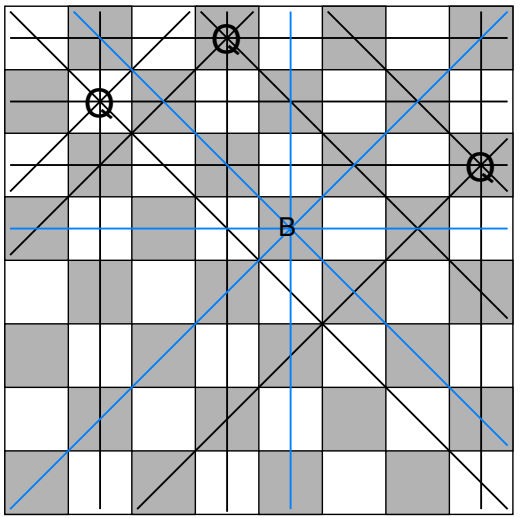


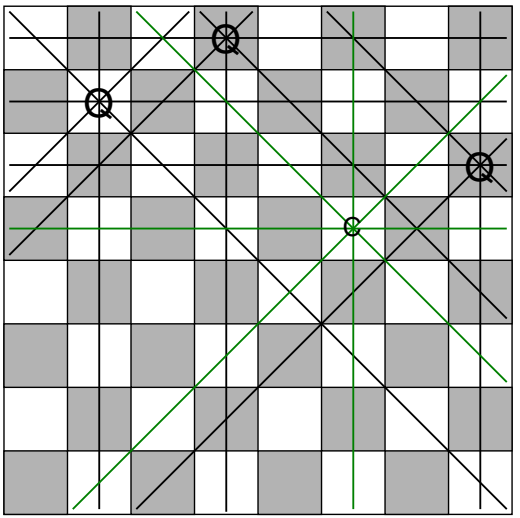
Examples for Search Problems

8-Queens Problem (continued)

Analysis of possible successor boards:



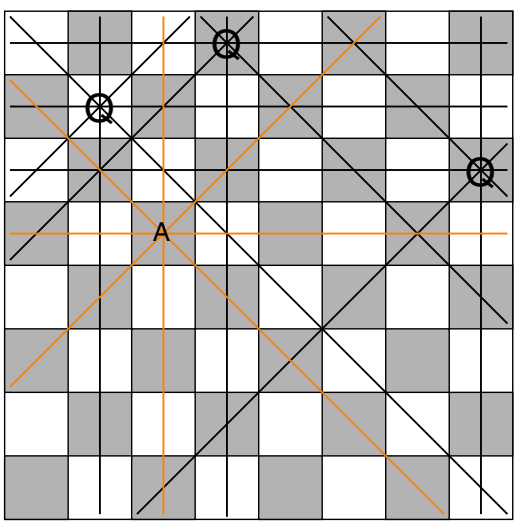


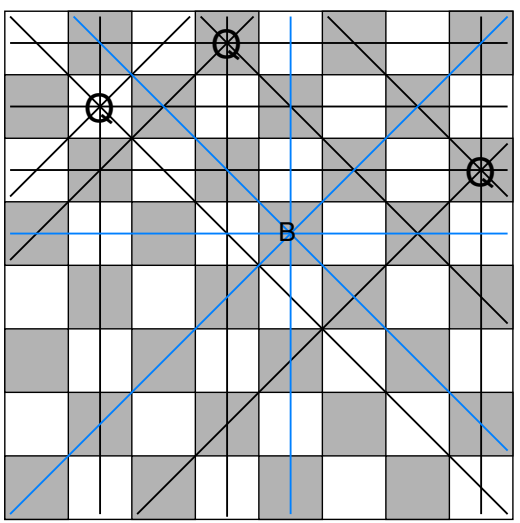


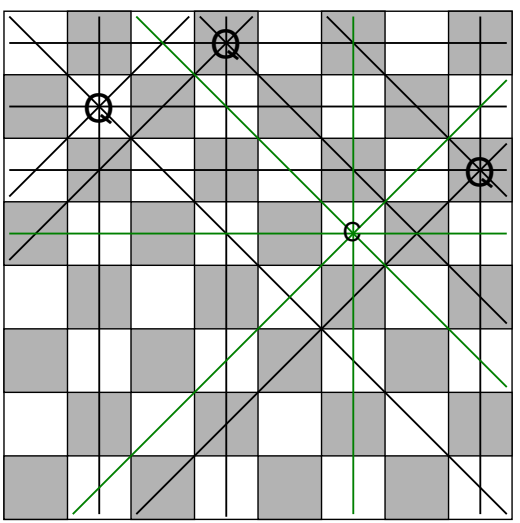
Examples for Search Problems

8-Queens Problem (continued)

Analysis of possible successor boards:







- Each queen defines *constraints* (restrictions).
- *Monotone* situation: constraint violations cannot be repaired.

Examples for Search Problems

8-Queens Problem (continued)

Desired: A heuristic for the next queen's placement.

Idea: Which placement x is least restrictive regarding future decisions?

Examples for Search Problems

8-Queens Problem (continued)

Desired: A heuristic for the next queen's placement.

Idea: Which placement x is least restrictive regarding future decisions?

- Heuristic 1: Maximize $h_1(x)$, with $h_1(x) =$ number of unattacked cells.

$$\begin{aligned} h_1(A) &= 8 \\ h_1(B) &= 9 \\ h_1(C) &= 10 \end{aligned}$$

Examples for Search Problems

8-Queens Problem (continued)

Desired: A heuristic for the next queen's placement.

Idea: Which placement x is least restrictive regarding future decisions?

- Heuristic 1: Maximize $h_1(x)$, with $h_1(x) =$ number of unattacked cells.

$$\begin{aligned} h_1(A) &= 8 \\ h_1(B) &= 9 \\ h_1(C) &= 10 \end{aligned}$$

- Heuristic 2: Maximize $h_2(x)$, with $h_2(x) = \min\{uc(x,r) \mid r \in \{r_5, \ldots, r_8\}\}$, where $uc(x,r)$ is the number of unattacked cells in row r if queen on x.

$$\begin{aligned} h_2(A) &= 1 \\ h_2(B) &= 1 \\ h_2(C) &= 2 \end{aligned}$$

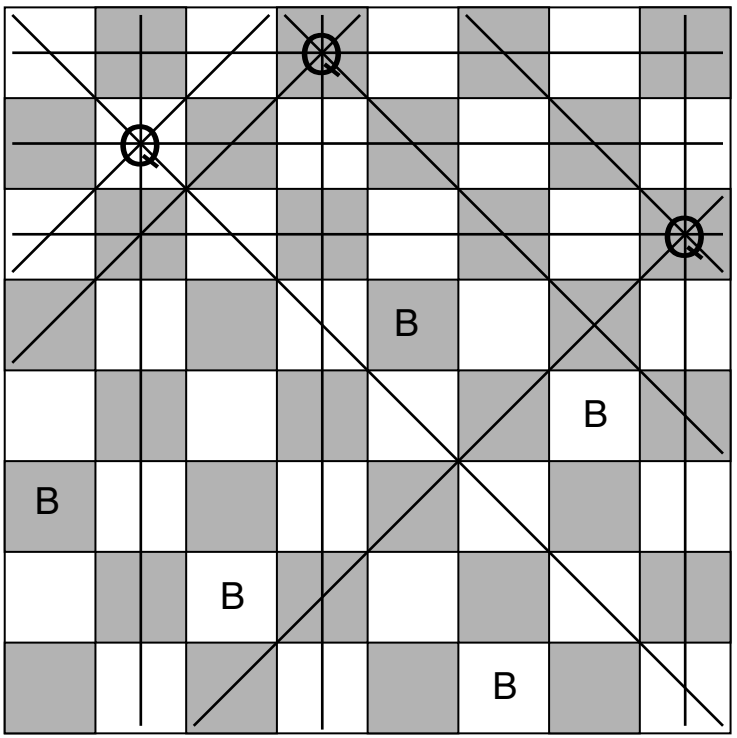
Remarks:

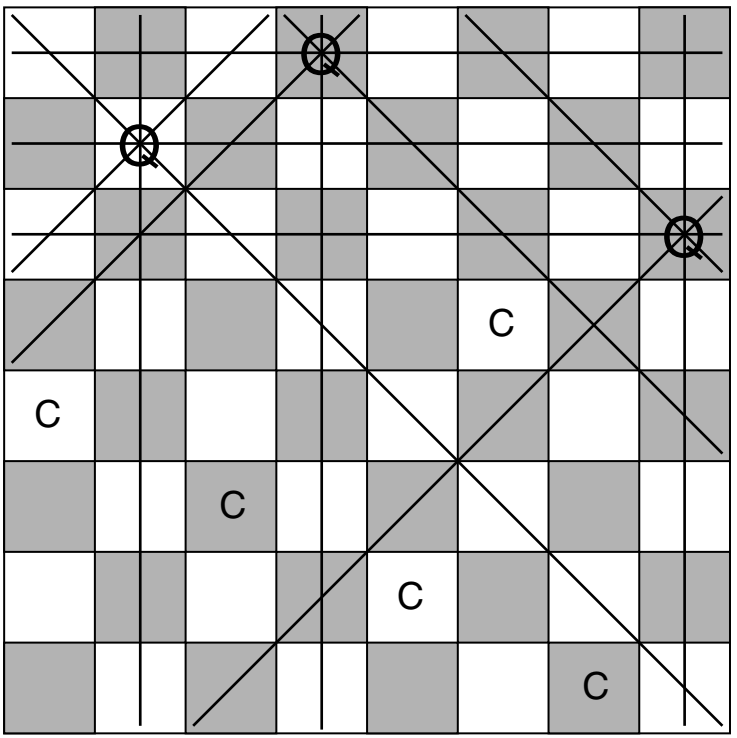
- The described idea became known in the field of Artificial Intelligence as the *Principle of Least Commitment*.
- Both 8-queens heuristics assess the board after the next placement.
- Both 8-queens heuristics compute merit values (not costs).

Examples for Search Problems

8-Queens Problem (continued)

Possible solutions:





Examples for Search Problems

8-Queens Problem (continued)

Comparison of h_1 and h_2:

- consider computational effort
- consider early recognition of dead ends:

  Predicate $\perp_f (x)$, $x \in \{A, B, C\}$: "x is a dead end under f"

$$\perp_{h_1} (x) = \begin{cases} \textit{True} & \text{If } h_1(x) < \text{number of remaining rows,} \\ \textit{False} & \text{Otherwise.} \end{cases}$$

$$\perp_{h_2} (x) = \begin{cases} \textit{True} & \text{If } h_2(x) = 0, \\ \textit{False} & \text{Otherwise.} \end{cases}$$

Heuristic 2 is more general than Heuristic 1: $\perp_{h_1} (x) \Rightarrow \perp_{h_2} (x)$

Compare the definition of "more general" in [ML:II Concept Learning: Search in Hypothesis Space].

Examples for Search Problems

8-Queens Problem (continued)

Problem: 8-Queens Problem

Instance: Fixed. (8×8 is size of chessboard and 8 is number of queens.)

Solution: Board positions for all queens
so that no two queens threaten each other.

Examples for Search Problems

8-Queens Problem (continued)

Problem: 8-Queens Problem

Instance: Fixed. (8×8 is size of chessboard and 8 is number of queens.)

Solution: Board positions for all queens
so that no two queens threaten each other.

Algorithmization

1. Encoding of solution candidates:
   (A2, B5, C3, D8, E7, F6, G4, H1)
2. Encoding of partial solutions:
   (A2, B5, C3, *, *, *, *, *)
3. Operators:
   Deciding the position of the next queen.

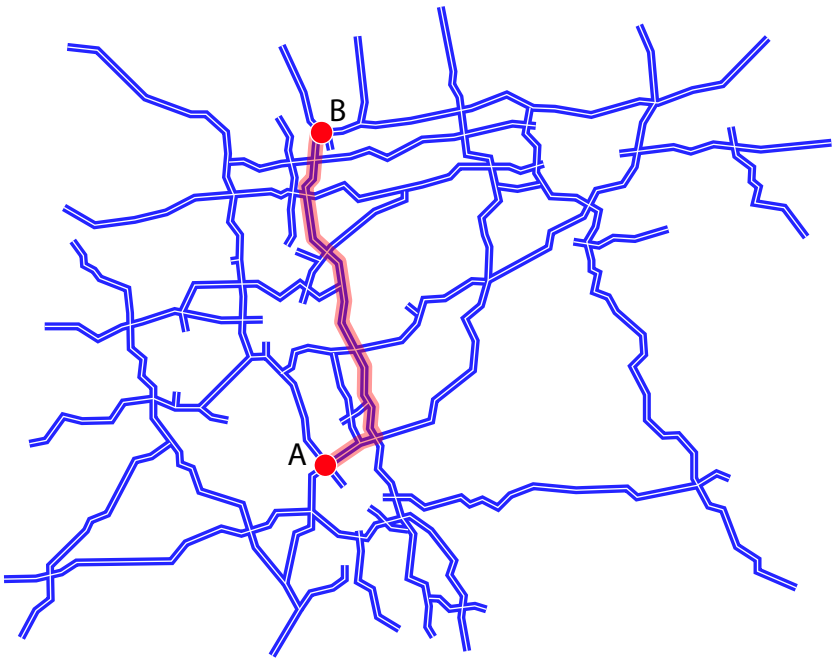
Remarks:

- Obviously, there is only a finite number of possible positionings. Therefore, enumerating and testing the positionings is a possible approach to solving the problem.
- The 8-queens problem can be generalized by allowing boards of different sizes. Then, additional parameters are needed in a description of a problem instance, e.g. $n \in \mathbf{N}$ for $n \times n$ chessboards and n queens.
- In the 8-Queens Problem we are interested board configurations. Such a configuration is easily computed from the sequence of positionings, but the sequence contains additional information that is not needed, e.e. an ordering of the positions. The encoding of solution candidates as lists enables an step-by-step construction of solution candidates.

Examples for Search Problems

Road Map Problem

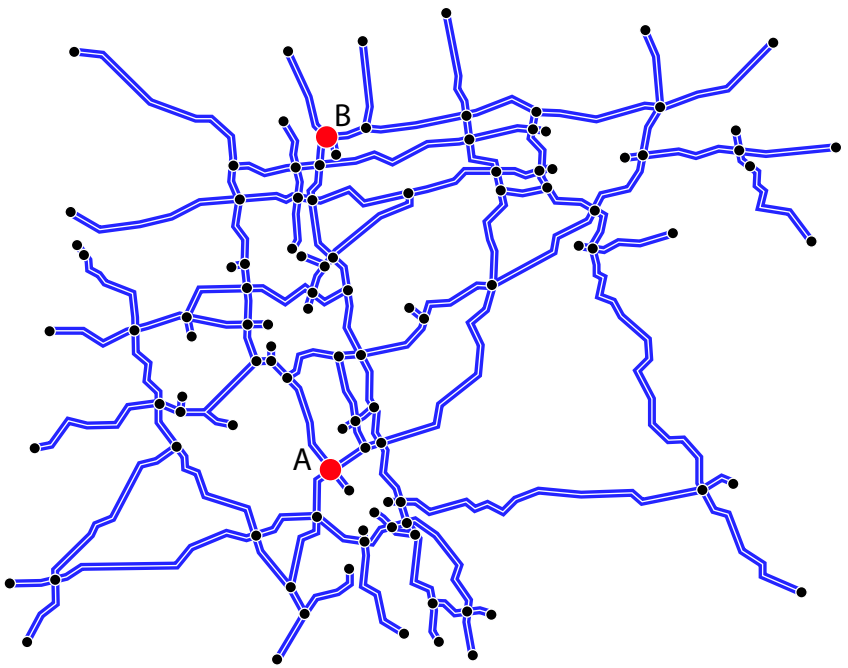
Search for a route from place A to B:



Examples for Search Problems

Road Map Problem (continued)

Using a graph model:

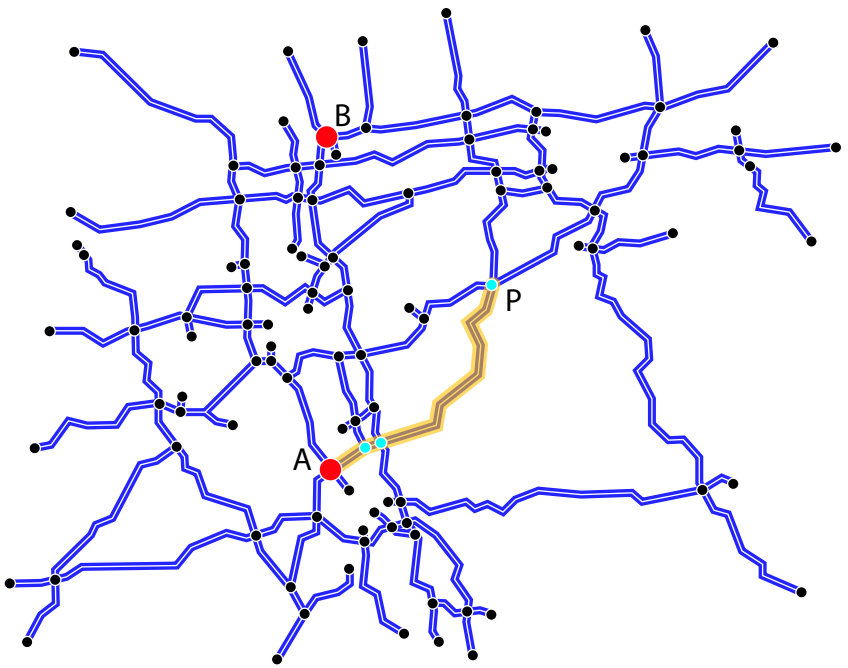


- Nodes represent crossings, junctions, endpoints,...
- Edges represent roads in between.
- Edge labels denote the length of a road between two nodes.

Examples for Search Problems

Road Map Problem (continued)

Search for a shortest path in the graph model:

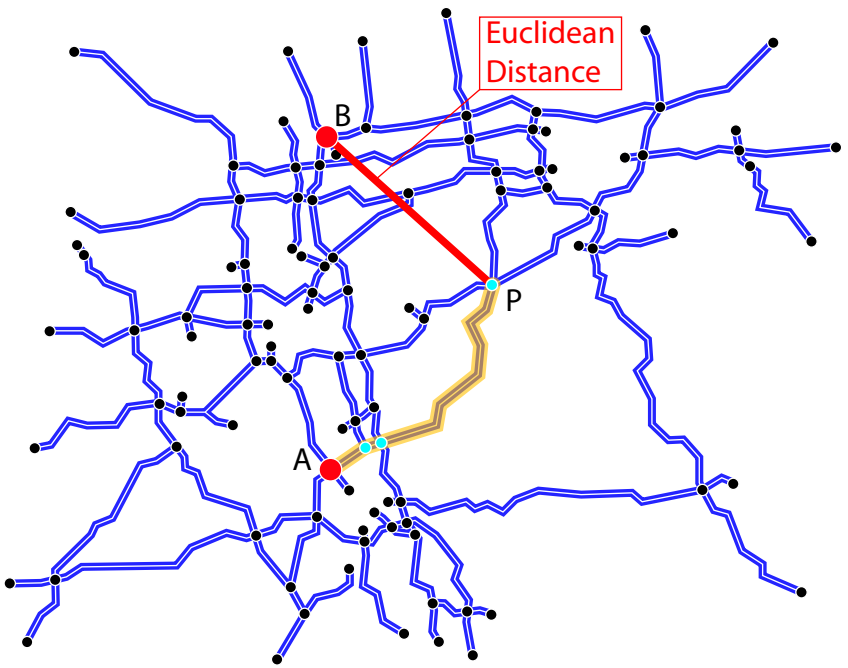


Desired: A heuristic for selection of a promising path.

Examples for Search Problems

Road Map Problem (continued)

Search for a shortest path in the graph model:



Desired: A heuristic for the selection of a promising path.

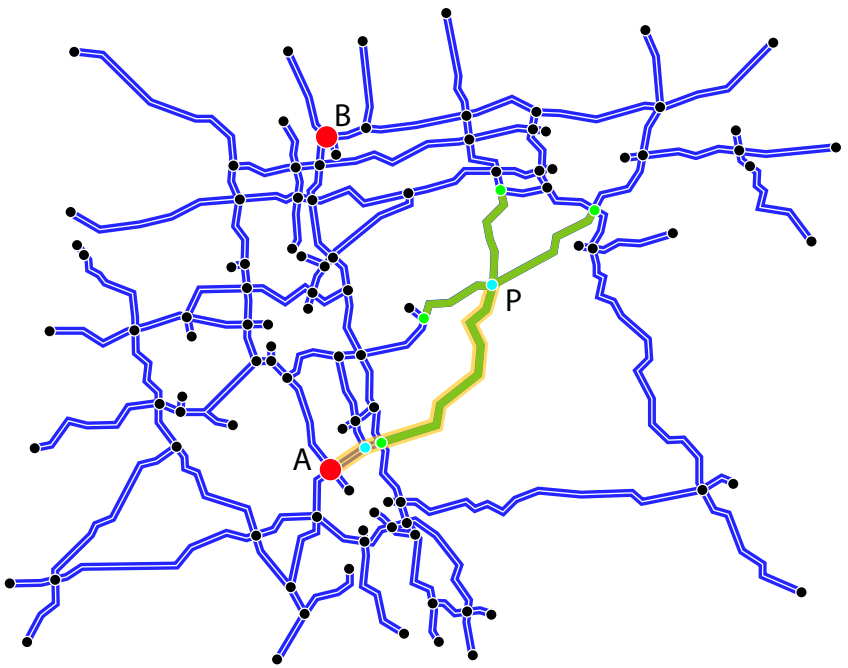
Idea: Which path will take you next to your target?

Heuristic: Use Euclidean distance between current position and target.

Examples for Search Problems

Road Map Problem (continued)

Search for a shortest path in the graph model:



Possible continuations of the selected path will be considered as new candidates.

Examples for Search Problems

Road Map Problem (continued)

Problem: Road Map Problem

Instance: G. A weighted finite graph representing a map with distances.
A, B. Startnode and endnode for the trip.

Solution: A shortest path starting in A and ending in B.

Examples for Search Problems

Road Map Problem (continued)

Problem: Road Map Problem

Instance: G. A weighted finite graph representing a map with distances.
A, B. Startnode and endnode for the trip.

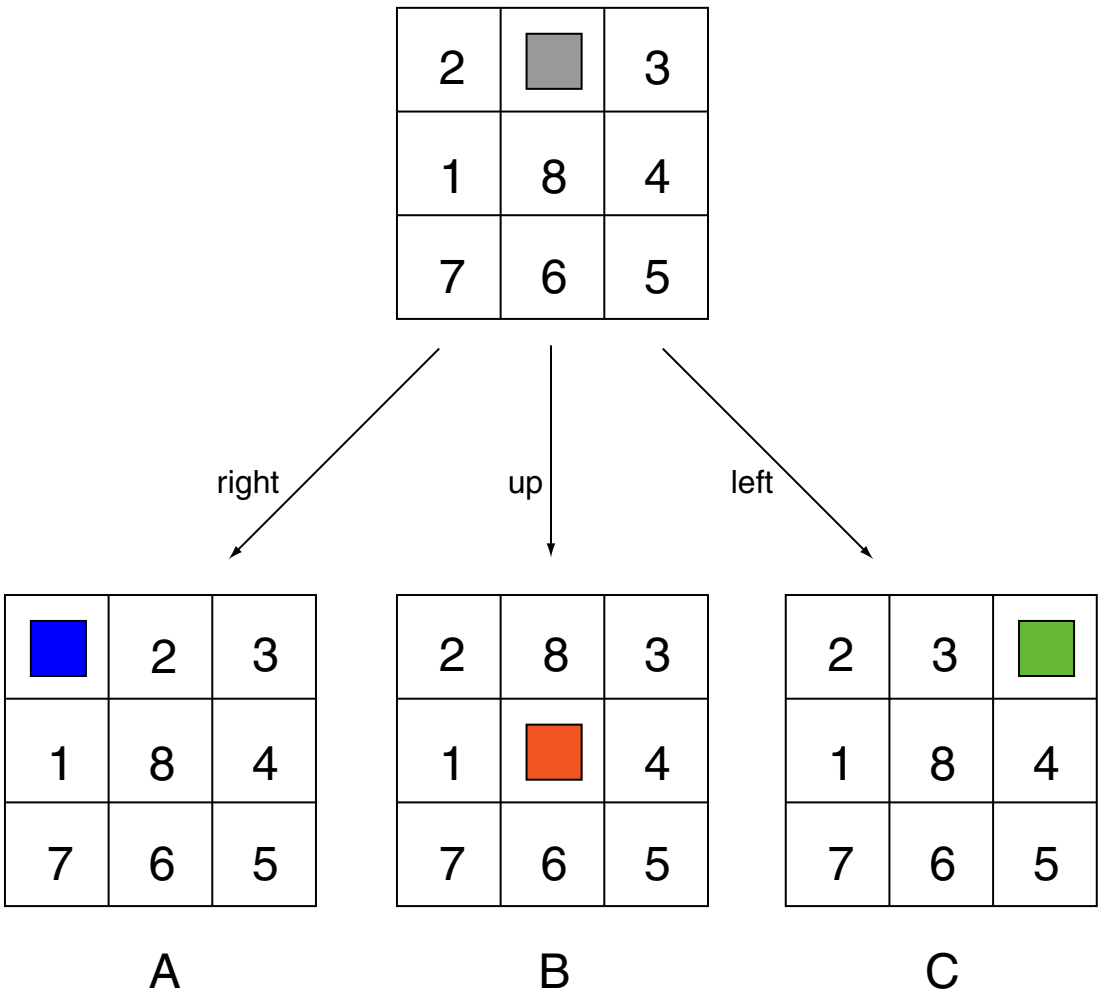
Solution: A shortest path starting in A and ending in B.

Algorithmization

1. Encoding of solution candidates:
   $(v_i)_{i=0}^N$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(v_i)_{i=0}^N$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Deciding the next junction to move to.

Examples for Search Problems

8-Puzzle Problem



Examples for Search Problems

8-Puzzle Problem (continued)

Desired: A heuristic for the next move.

Idea: Which move x minimizes the "distance" to the goal state?

Examples for Search Problems

8-Puzzle Problem (continued)

Desired: A heuristic for the next move.

Idea: Which move x minimizes the "distance" to the goal state?

- Heuristic 1: Minimize $h_1(x)$, with $h_1(x) =$ number of non-matching tiles.

$$\begin{aligned} h_1(A) &= 2 \\ h_1(B) &= 3 \\ h_1(C) &= 4 \end{aligned}$$

Examples for Search Problems

8-Puzzle Problem (continued)

Desired: A heuristic for the next move.

Idea: Which move x minimizes the "distance" to the goal state?

- Heuristic 1: Minimize $h_1(x)$, with $h_1(x) =$ number of non-matching tiles.

$$\begin{aligned} h_1(A) &= 2 \\ h_1(B) &= 3 \\ h_1(C) &= 4 \end{aligned}$$

- Heuristic 2: Minimize $h_2(x)$, with $h_2(x) =$ sum of city block (Manhattan) distances of non-matching tiles.

$$\begin{aligned} h_2(A) &= 2 \\ h_2(B) &= 4 \\ h_2(C) &= 4 \end{aligned}$$

Remarks:

- In the 8-puzzle problem the goal state (final configuration) is known, while in the 8-queens problem the goal state is unknown and has to be determined.
- Both 8-puzzle heuristics assess the unsolved rest problem.
- Both 8-puzzle heuristics compute cost values (not merits).
- “Wrong” decisions can be repaired.
- Infinitely long move sequences are possible.

Examples for Search Problems

8-Puzzle Problem (continued)

Problem: 8-Puzzle Problem

Instance: q_s. Initial board configuration.
q_γ. Final board configuration.

Solution: A sequence of moves that transforms the initial configuration q_s into the final configuration q_γ.

Examples for Search Problems

8-Puzzle Problem (continued)

Problem: 8-Puzzle Problem

Instance: q_s. Initial board configuration.
q_γ. Final board configuration.

Solution: A sequence of moves that transforms the initial configuration q_s into the final configuration q_γ.

Algorithmization

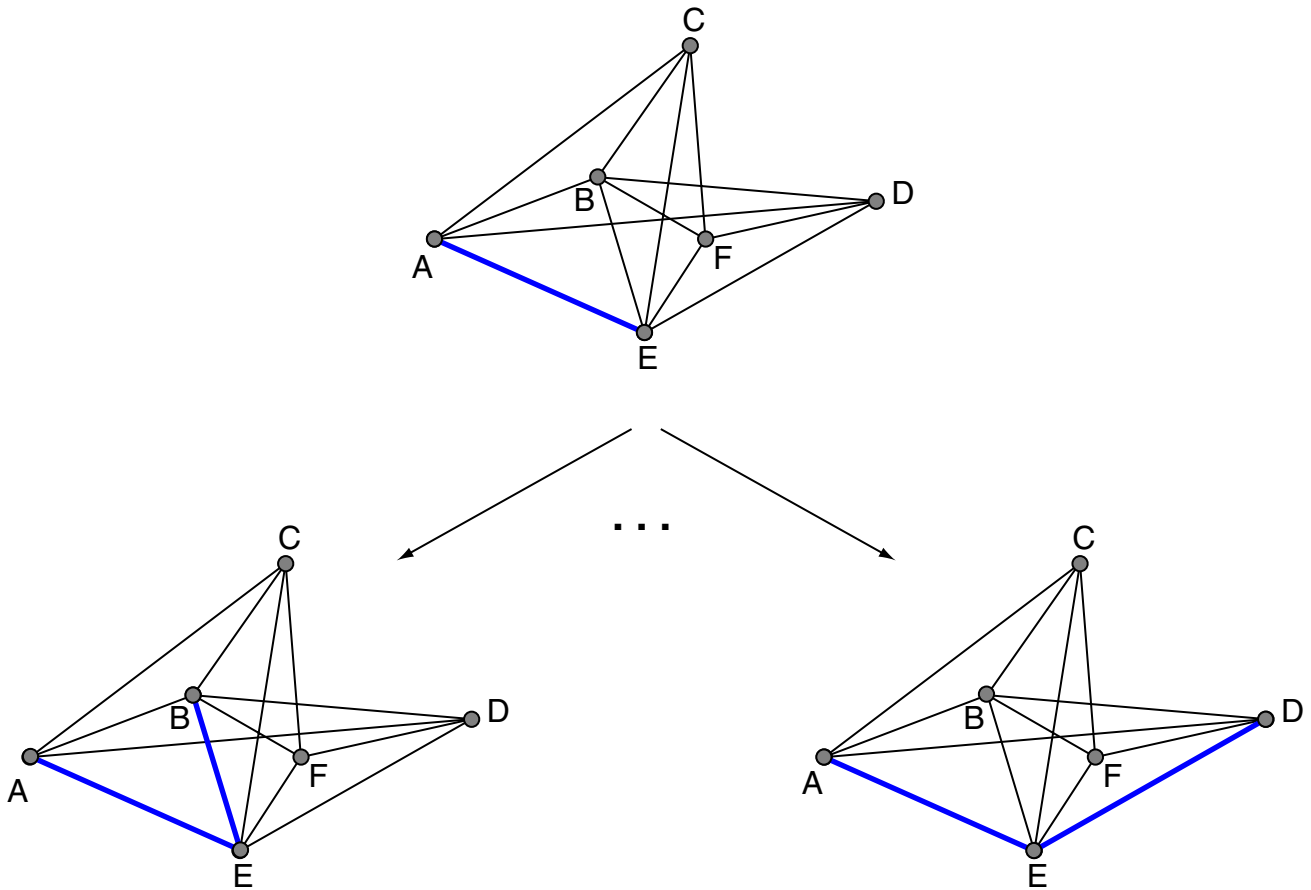
1. Encoding of solution candidates:
   $(m_i)_{i=1}^N$ with $m_i \in \{\text{up}, \text{down}, \text{left}, \text{right}\}$, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(m_i)_{i=1}^N$ with $m_i \in \{\text{up}, \text{down}, \text{left}, \text{right}\}$, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Deciding the next move.

Remarks:

- Obviously, the number of possible sequences of moves is infinite. Therefore, enumerating and testing the positionings is an approach to solving the problem without guarantee of termination.
- The 8-puzzle problem can be generalized by allowing boards of different sizes or shapes. Then, additional parameters are needed in a description of a problem instance, e.g. $m, n \in \mathbf{N}$ for rectangular boards with m rows and n columns.
- Additional restrictions can be placed on solutions to the 8-puzzle problem. The task can be the identification of a shortest sequence or of a sequence that is shorter than a given maximum length.

Examples for Search Problems

Traveling Salesman Problem (TSP)



Examples for Search Problems

Traveling Salesman Problem (continued)

Desired: A heuristic for the most suited next town x.

Idea: What is a lower bound for the shortest tour?

Examples for Search Problems

Traveling Salesman Problem (continued)

Desired: A heuristic for the most suited next town x.

Idea: What is a lower bound for the shortest tour?

- Heuristic 1: For the remaining nodes $V' \subset V$ minimize the edge weight of a subgraph on V' whose degree is ≤ 2.

Examples for Search Problems

Traveling Salesman Problem (continued)

Desired: A heuristic for the most suited next town x.

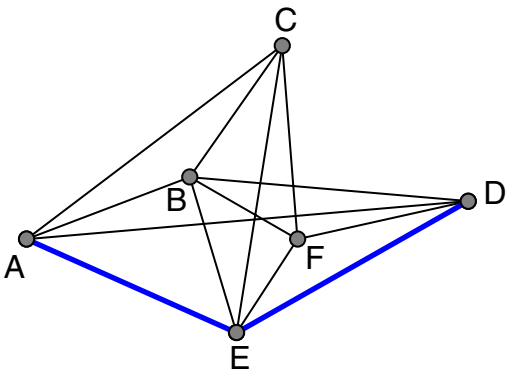
Idea: What is a lower bound for the shortest tour?

- Heuristic 1: For the remaining nodes $V' \subset V$ minimize the edge weight of a subgraph on V' whose degree is ≤ 2.

- Heuristic 2: For the remaining nodes $V' \subset V$ compute the edge weight of a minimum spanning tree.

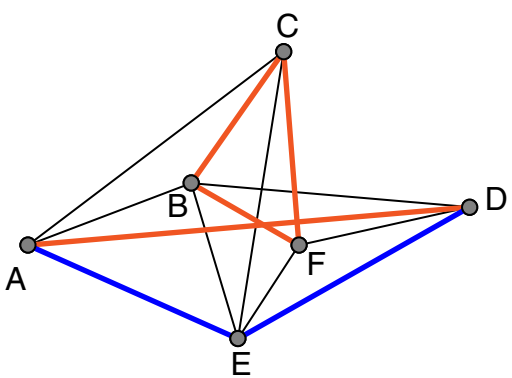
Examples for Search Problems

Traveling Salesman Problem (continued)

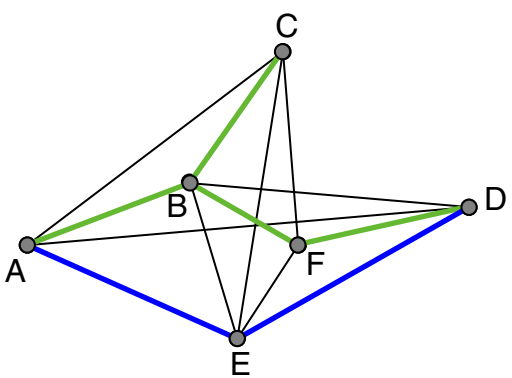


Current State

Estimation of
Completion Cost



Heuristic 1:
cheapest degree-2 graph



Heuristic 2:
minimum spanning tree

Remarks:

- Both TSP heuristics compute cost values (not merits).
- Both TSP heuristics are *optimistic*, i.e., they underestimate the cost of the rest tour.
- The cost of a candidate is computed as the sum of true cost of the completed part of the tour plus the estimated cost of the rest tour.

Examples for Search Problems

Traveling Salesman Problem (continued)

Problem: Traveling Salesman Problem

Instance: G. A weighted finite graph.
A. Start node for the round-trip.

Solution: A shortest path starting and ending in A
that visits each other node of G exactly once.

Examples for Search Problems

Traveling Salesman Problem (continued)

Problem: Traveling Salesman Problem

Instance: G. A weighted finite graph.
A. Start node for the round-trip.

Solution: A shortest path starting and ending in A
that visits each other node of G exactly once.

Algorithmization

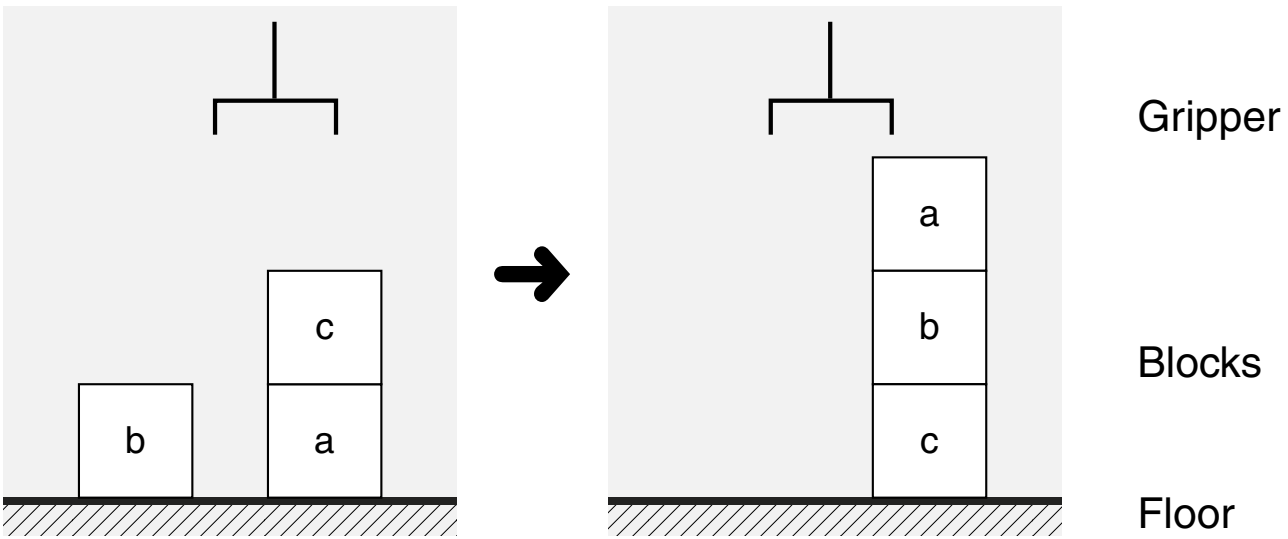
1. Encoding of solution candidates:
   $(v_i)_{i=0}^N$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(v_i)_{i=0}^N$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Deciding the next junction to move to.

Remarks:

- Obviously, there is only a finite number of paths without cycles. Therefore, enumerating and testing these paths is a possible approach to solving the problem.
- Additional restrictions can be placed on solutions to the traveling salesman problem. The task can be the identification of a shortest round-trip or of a round-trip that is shorter than a given maximum length.

Examples for Search Problems

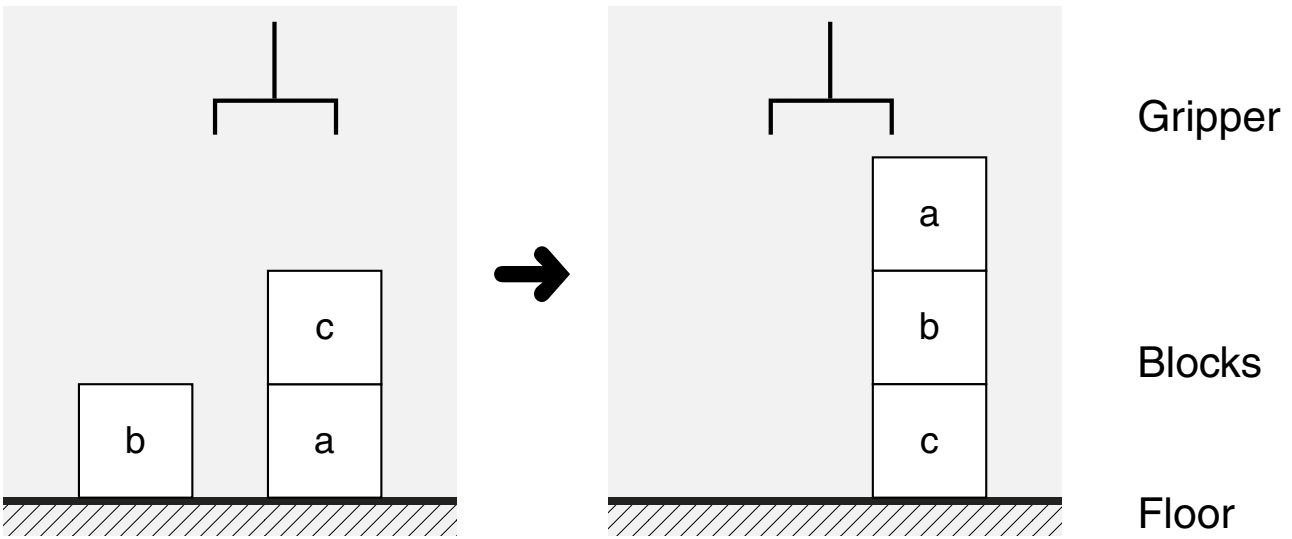
Blocks World Planning



A gripper moves single blocks (free blocks, i.e. no other block on top) to new positions, i.e. either on top of a free block or on the floor.

Examples for Search Problems

Blocks World Planning



A gripper moves single blocks (free blocks, i.e. no other block on top) to new positions, i.e. either on top of a free block or on the floor.

Desired: A heuristic for the next move.

Idea: Which move x minimizes the "distance" to the goal state?

Heuristic: Minimize number of blocks at wrong positions.

Examples for Search Problems

Blocks World Planning (continued)

Problem: Blocks World Planning

Instance: q_s. Initial block configuration.
q_γ. Final block configuration.

Solution: A sequence of moves that transforms the initial configuration q_s into the final configuration q_γ.

Examples for Search Problems

Blocks World Planning (continued)

Problem: Blocks World Planning

Instance: q_s. Initial block configuration.
q_γ. Final block configuration.

Solution: A sequence of moves that transforms the initial configuration q_s into the final configuration q_γ.

Algorithmization

1. Encoding of solution candidates:
   $(op_i)_{i=1}^N$ with op_i a gripper operation on some block, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(op_i)_{i=1}^N$ with op_i a gripper operation on some block, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Deciding the next operation to apply.

Remarks:

- Logical descriptions of Blocks World Planning states use predicates "*ontop*(x, y)" and "*onfloor*(x)" for the position of blocks
- The planning task depicted is called Sussman Anomaly. This anomaly illustrates the weakness of planning algorithms that try to solve a task by appending plans for achieving single properties, e.g. "*ontop*(b, c)" and "*ontop*(a, b)". Optimum plans for these subgoal cannot be combined by simply appending such plans.

Examples for Search Problems

Problem Abstraction: State Transition System

Problem:	State Transition System	
Instance:	S. A set of states	representing (remaining) problems.
	$T \subseteq S \times S$. A transition relation	representing solution steps.
	$s \in S$. A start state	representing the initial problem.
	$F \subseteq S$. A set of final states	representing solved problems.
Solution:	A sequence of transitions leading from state s to some state in F.	

Examples for Search Problems

Problem Abstraction: State Transition System

Problem:	State Transition System	
Instance:	S. A set of states	representing (remaining) problems.
	$T \subseteq S \times S$. A transition relation	representing solution steps.
	$s \in S$. A start state	representing the initial problem.
	$F \subseteq S$. A set of final states	representing solved problems.
Solution:	A sequence of transitions leading from state s to some state in F.	

Algorithmization

1. Encoding of solution candidates:
   $(s_i)_{i=0}^N$ with $s_0 = s$, $s_i \in S$, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(s_i)_{i=0}^N$ with $s_0 = s$, $s_i \in S$, $N \in \mathbf{N}$ (initial sequence to be continued)
3. Operators:
   Deciding the next transition to apply.

Remarks:

- In planning, description languages are used to specify the state transition systems underlying a planning problem. A commonly used example is the Planning Domain Definition Language (PDDL), which was inspired by STRIPS and ADL.
  States are characterized by their properties and transitions (or actions) describe the changes in the properties of a state.
- In all examples a solution is incrementally constructed as sequence of operators (next move, next town,. . .) applications.

Search Problem Abstraction

Questions

- What is the original problem?
- How does the structure of a solution look like?
- Which simplifications of the problem are possible?
- Which rest problems result from problem simplifications?

➜ How to model the original problem by applying problem simplifications?