Chapter S:I

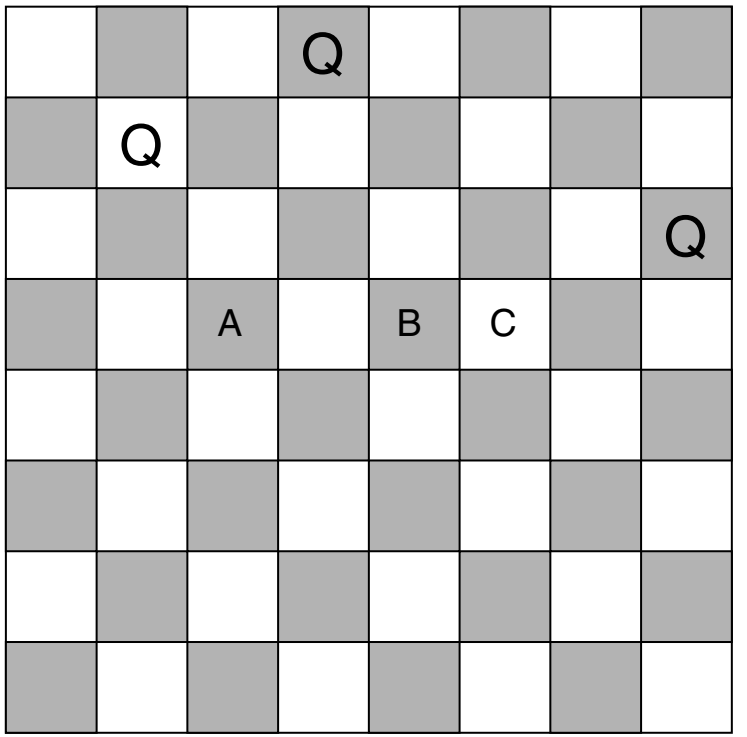
I. Introduction

- Examples for Search Problems
- Search Problem Abstraction

Examples for Search Problems

8-Queens Problem

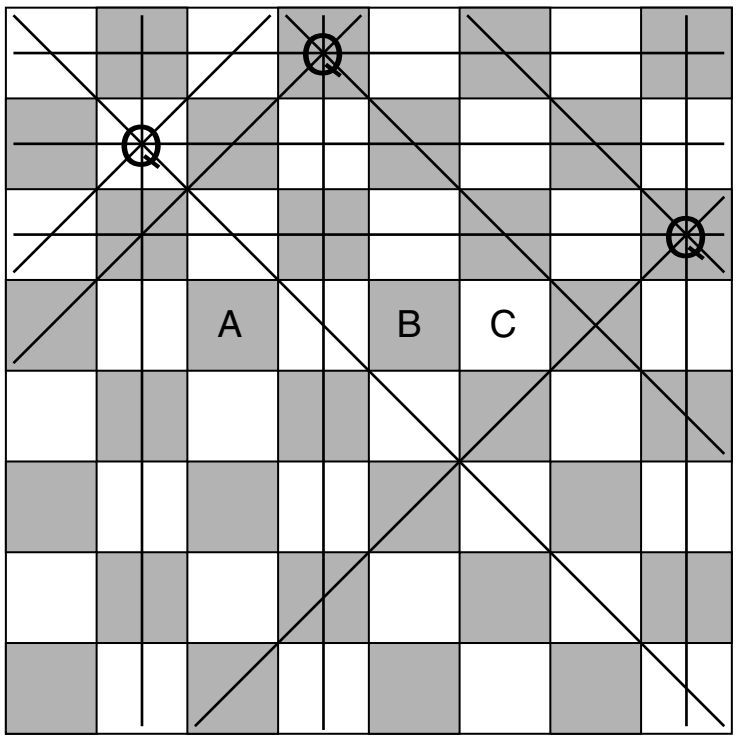
Incremental placement of queens:



Examples for Search Problems

8-Queens Problem (continued)

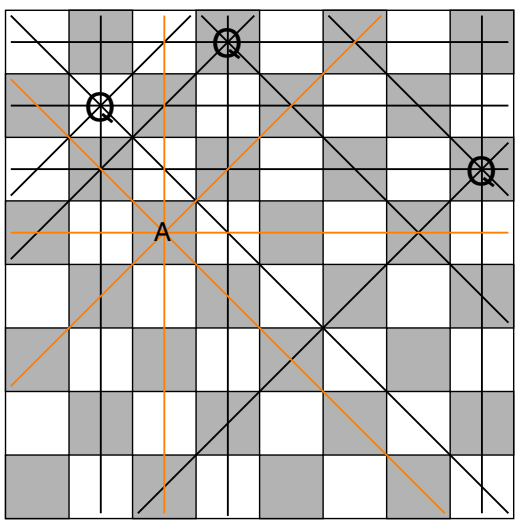
Analysis of the current board:

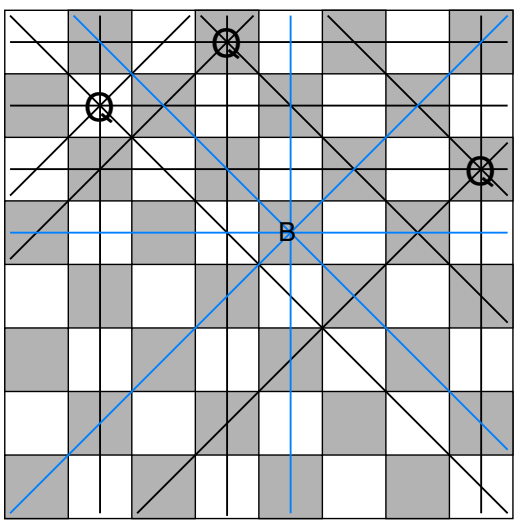


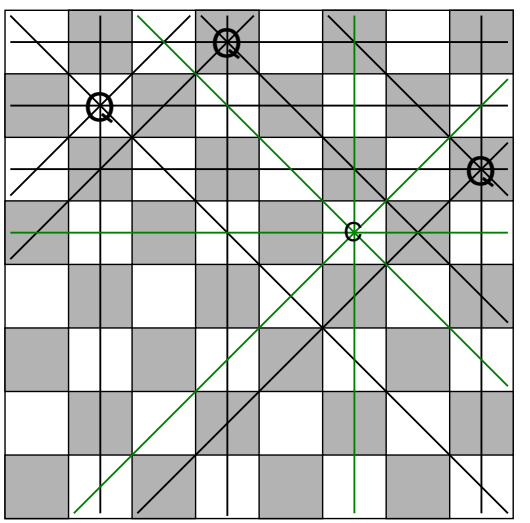
Examples for Search Problems

8-Queens Problem (continued)

Analysis of possible successor boards:



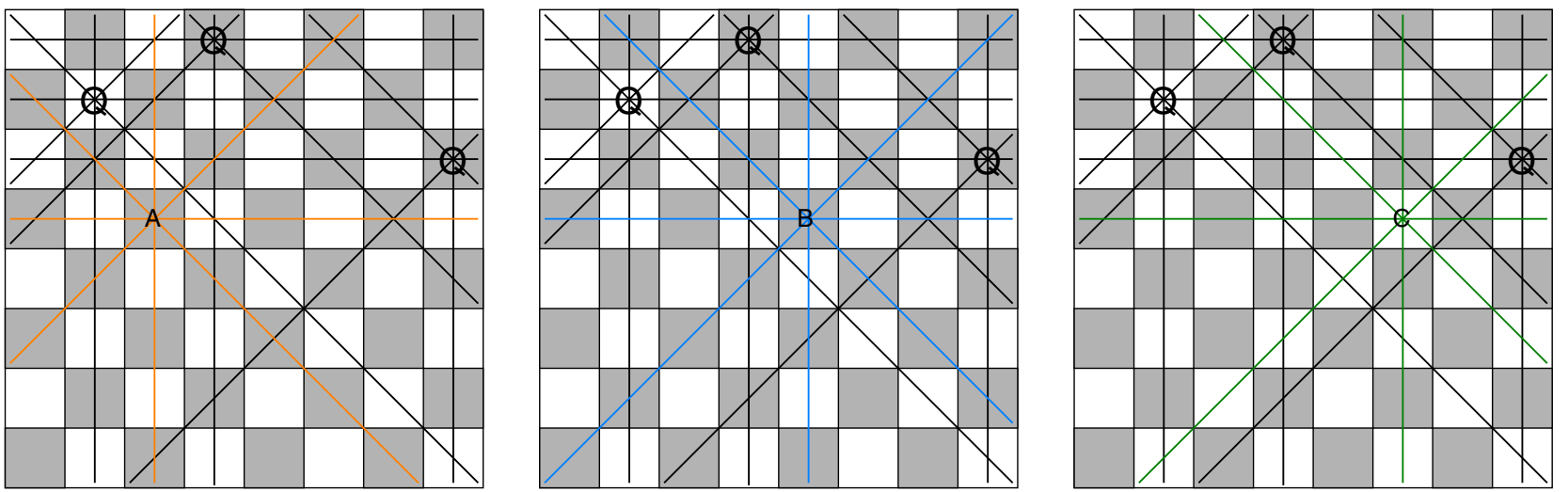




Examples for Search Problems

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 $h_1(x) =$ 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 $h_1(x) =$ number of unattacked cells.

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

- Heuristic 2: Maximize $h_2(x) = \min\{uc(x,r) \mid r \text{ is row without queen}\}$, where $uc(x,r)$ is the number of unattacked cells in row r if queen on x.

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

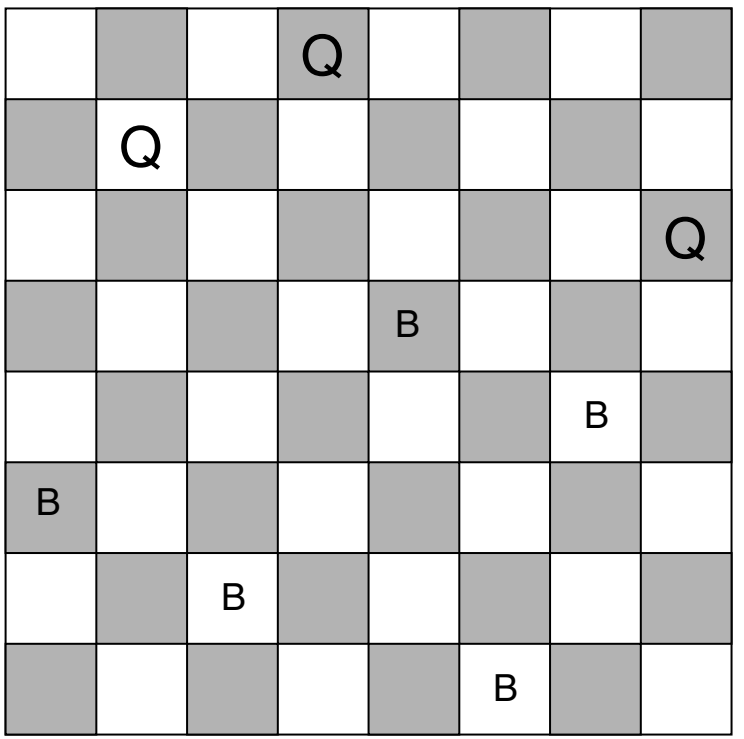
Remarks:

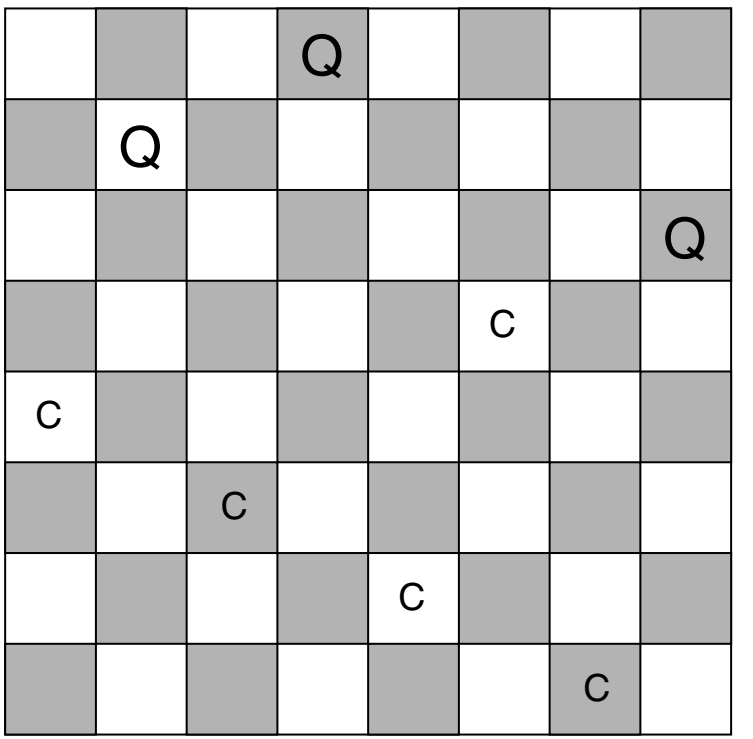
- The described idea became known in the field of Artificial Intelligence as the *Principle of Least Commitment*.
- Both 8-queens heuristics assess the board after the next placement.
- Both 8-queens heuristics compute merit values (not costs).
- These heuristics are helpful when solving the 8-queens problem by depth-first or hill-climbing search: successor nodes are processed in order of their promise which is defined by the heuristic.

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

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

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:
   Positioning of a queen in the next empty row (without resulting threats).
4. Heuristic:
   Try a most promising positioning first.

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 in board configurations. Such a configuration is easily computed from the sequence of positionings, but the sequence contains additional information that is not needed (e.g., an ordering of the positionings). The encoding of solution candidates as lists enables a step-by-step construction of solution candidates.

Examples for Search Problems

SAT Problem

Search for a truth assignment that satisfies all clauses:

Propositional formula in conjunctive normal form (CNF)

$$
\begin{array}{rcrcrcrcr}
\neg x_1 & \vee & x_3 & \vee & x_7 & \vee & \neg x_9 & & \\
x_2 & \vee & \neg x_3 & \vee & \neg x_5 & \vee & x_6 & \vee & \neg x_8 \\
\neg x_2 & \vee & x_7 & \vee & x_8 & & & & \\
x_1 & \vee & \neg x_2 & \vee & \neg x_3 & \vee & x_9 & & \\
\vdots & & & & & & & & \\
\neg x_3 & \vee & x_4 & \vee & \neg x_6 & \vee & x_7 & \vee & \neg x_8
\end{array}
$$

Examples for Search Problems

SAT Problem

Search for a truth assignment that satisfies all clauses:

Propositional formula in conjunctive normal form (CNF)

$$
\begin{array}{rcrcrcrcr}
\neg x_1 & \vee & x_3 & \vee & x_7 & \vee & \neg x_9 & & \\
x_2 & \vee & \neg x_3 & \vee & \neg x_5 & \vee & x_6 & \vee & \neg x_8 \\
\neg x_2 & \vee & x_7 & \vee & x_8 & & & & \\
x_1 & \vee & \neg x_2 & \vee & \neg x_3 & \vee & x_9 & & \\
\vdots & & & & & & & & \\
\neg x_3 & \vee & x_4 & \vee & \neg x_6 & \vee & x_7 & \vee & \neg x_8
\end{array}
$$

Start from an arbitrary truth assignment for the variables in α and iteratively flip truth values assigned to variables.

Heuristics:

GSAT: Flip the truth value of a variable so that the number of satisfied clauses increases the most.

WalkSAT: Flip the truth value of a variable in an unsatisfied clause.

Remarks:

- There is no unique way of constructing an assignment by iterated flipping of truth assignments for variables.
- Restarting the algorithm with a new random assignment can help when getting stuck in a local maximum of numbers of satisfied clauses.
- The optimization version MaxSAT determines for a given CNF formula the maximum number of clauses that can be satisfied by a truth assignment. Since this maximum number is not known in advance, there is no termination criterion. Either all possible truth assignments have to be tested or we terminate with a local maximum.

Examples for Search Problems

SAT Problem (continued)

Problem: SAT

Instance: α. Propositional formula in CNF.

Solution: Truth assignment that satisfies all clauses in α.

Examples for Search Problems

SAT Problem (continued)

Problem: SAT

Instance: α. Propositional formula in CNF.

Solution: Truth assignment that satisfies all clauses in α.

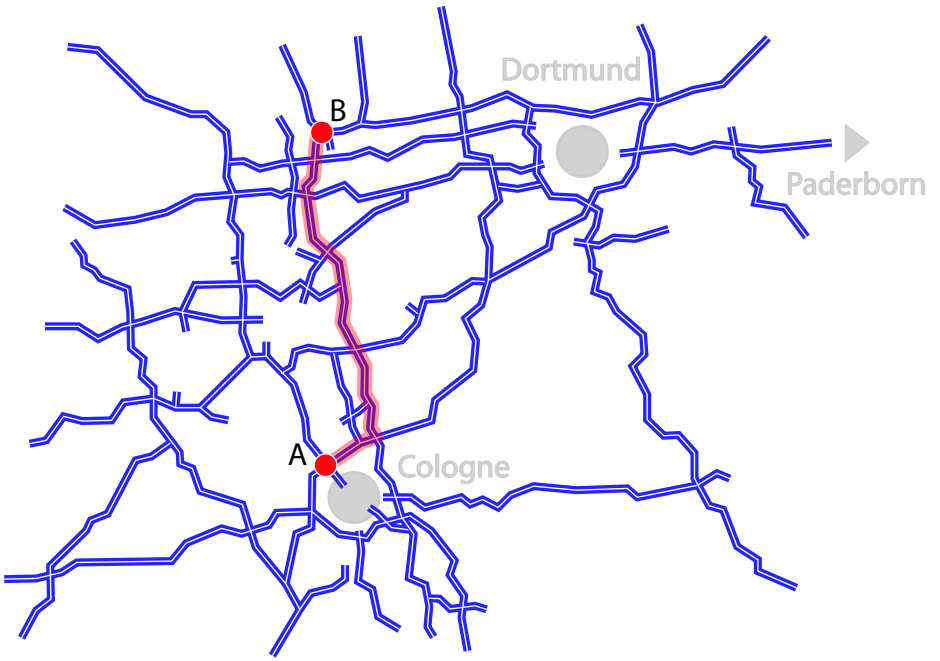
Algorithmization

1. Encoding of solution candidates:
   $(x_1 \mapsto \textit{true}, x_2 \mapsto \textit{false}, \ldots, x_n \mapsto \textit{true})$
2. Encoding of partial solutions:
   $(x_1 \mapsto \textit{true}, x_2 \mapsto \textit{false})$
3. Operators:
   Flip the truth assignment of a variable.
4. Heuristic:
   Try a most promising flipping first.

Examples for Search Problems

Road Map Problem

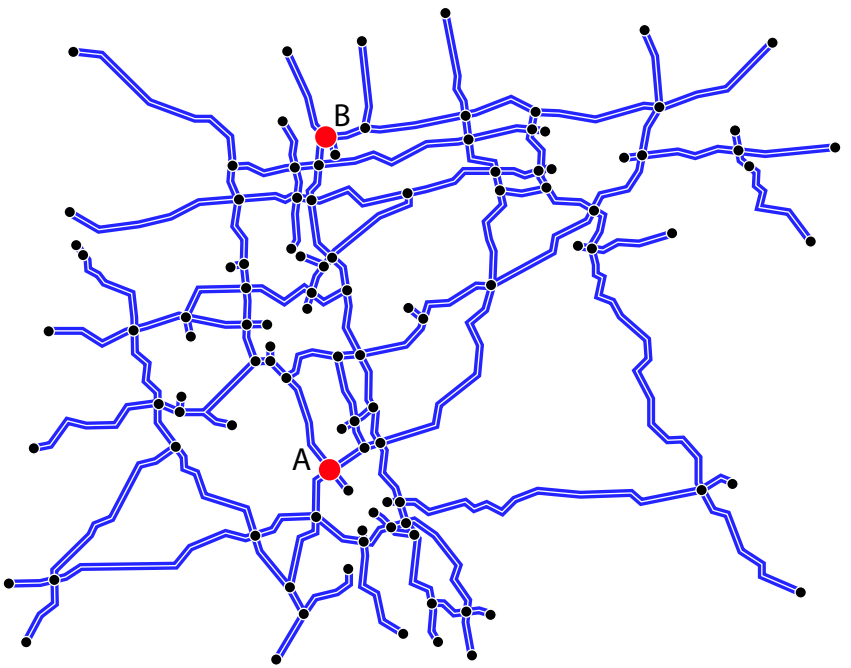
Search for a route from place A to B:



Examples for Search Problems

Road Map Problem (continued)

Using a graph model:

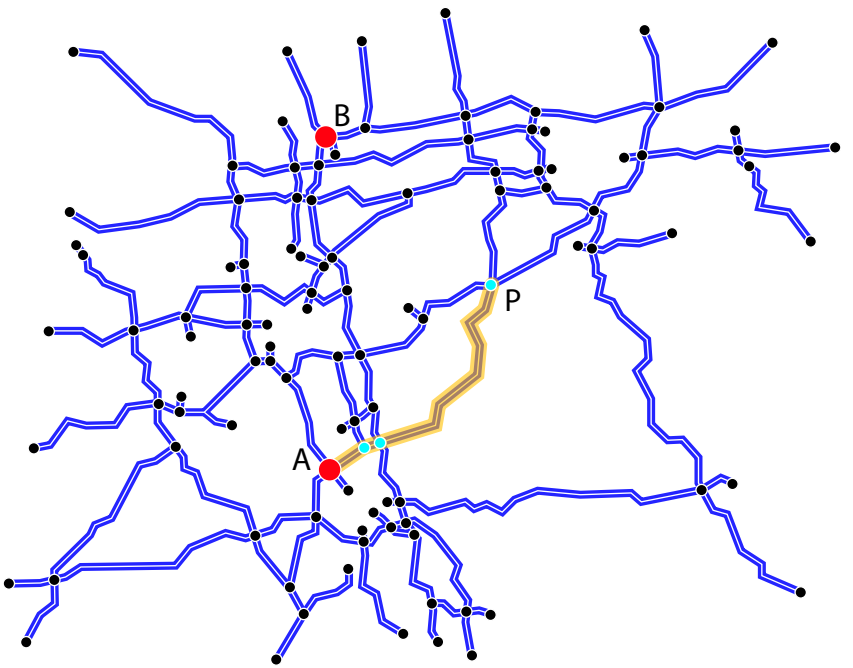


- Nodes represent crossings, junctions, endpoints, . . .
- Edges represent roads in between.
- Edge labels denote the length of a road between two nodes.

Examples for Search Problems

Road Map Problem (continued)

Search for a shortest path in the graph model:

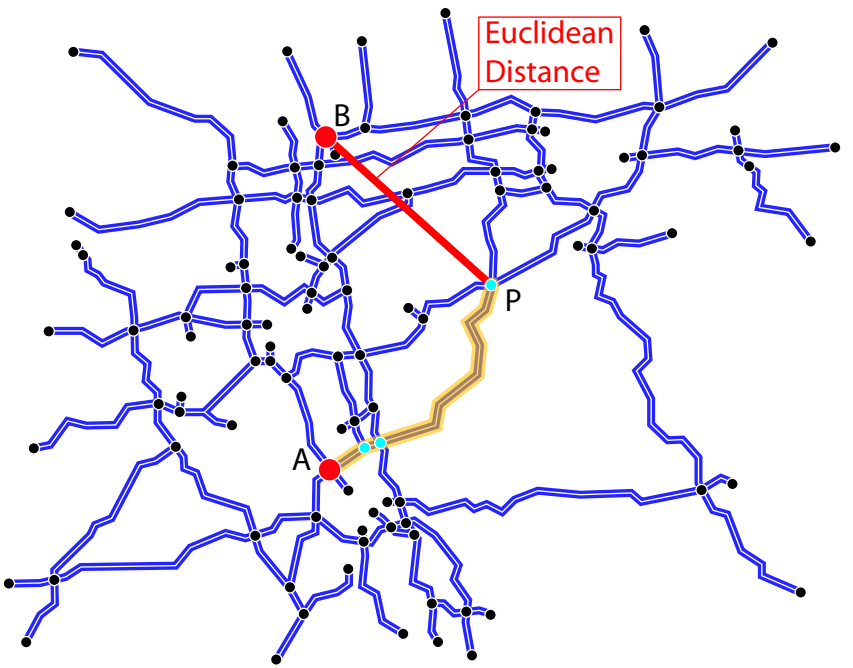


Desired: A heuristic for selection of a promising path.

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.

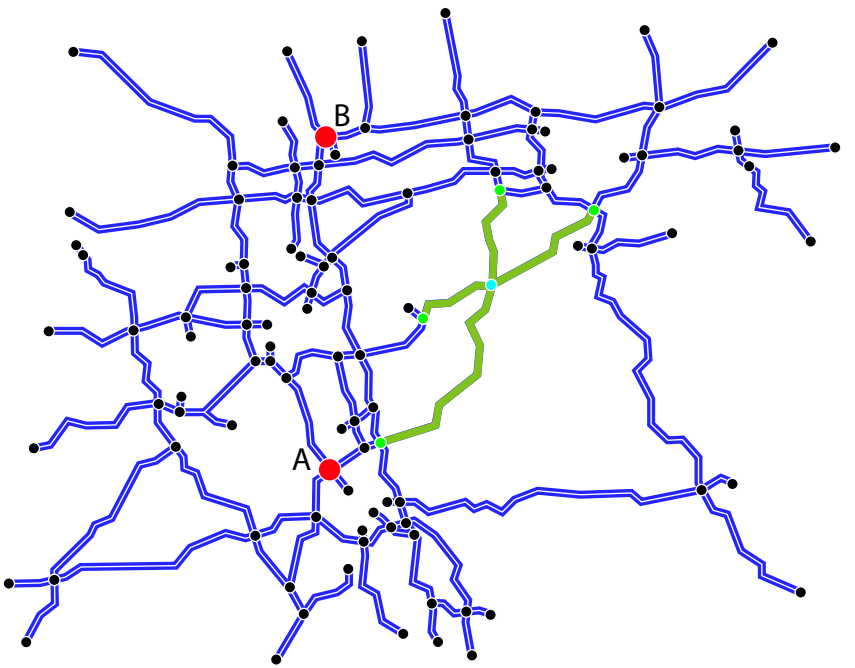
Idea: Which path will take you next to your target?

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.

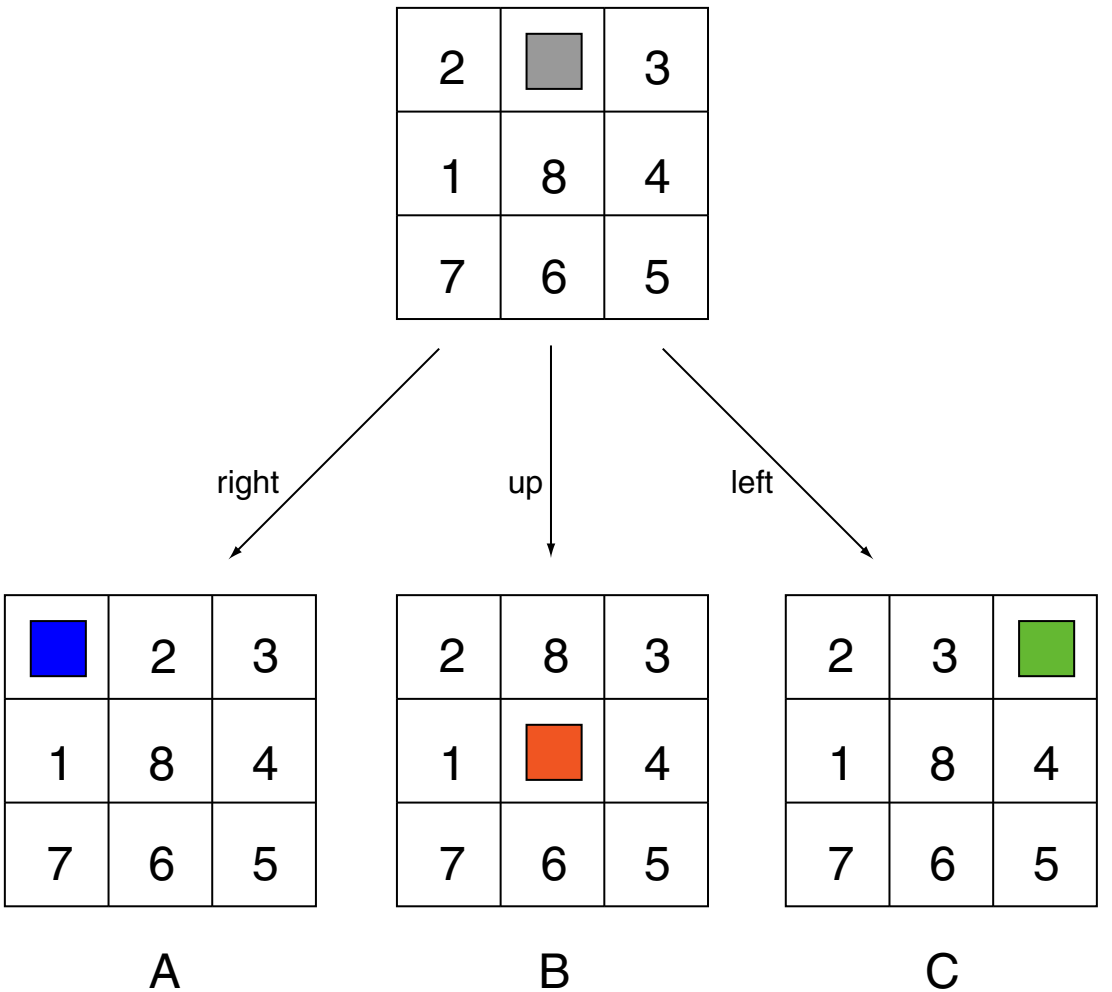
Solution: A shortest path starting in A and ending in B.

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:
   Extend the path by a move to an adjacent junction.
4. Heuristic:
   Try a most promising junction first.

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

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

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

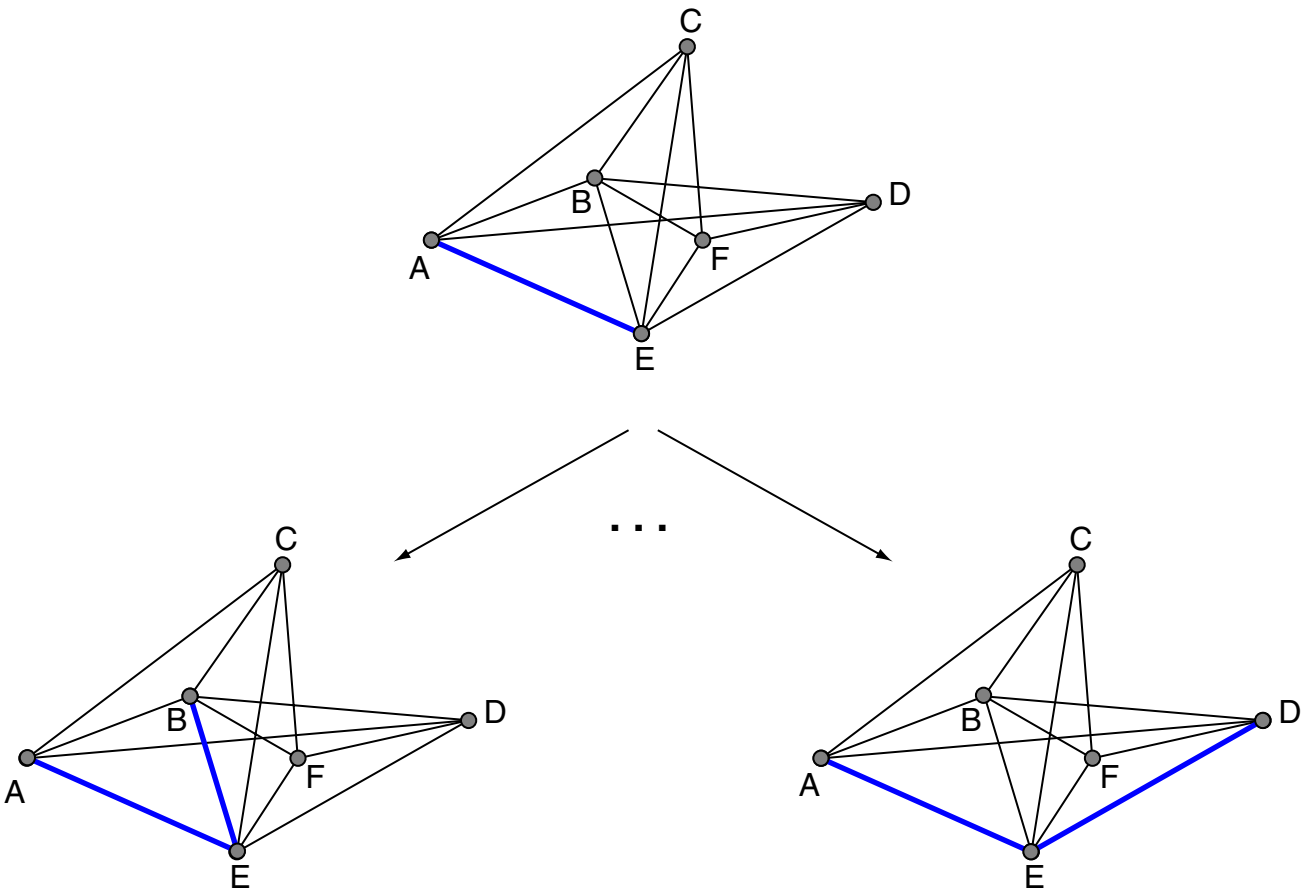
1. Encoding of solution candidates:
   $(m_i)_{i=1}^N$ with $m_i \in \{\text{up}, \text{down}, \text{left}, \text{right}\}$, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(m_i)_{i=1}^N$ with $m_i \in \{\text{up}, \text{down}, \text{left}, \text{right}\}$, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Extend the sequence of moves by a legal move.
4. Heuristic:
   Try a most promising next move first.

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.

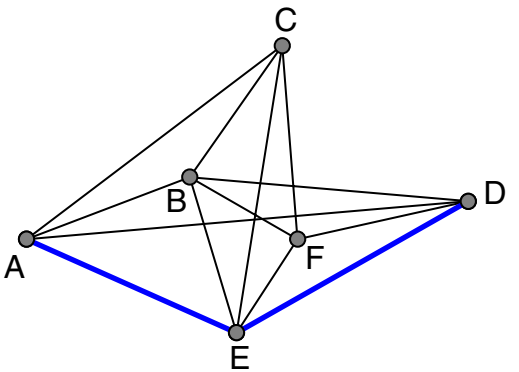
Idea: What is a lower bound for the shortest tour?

- 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.

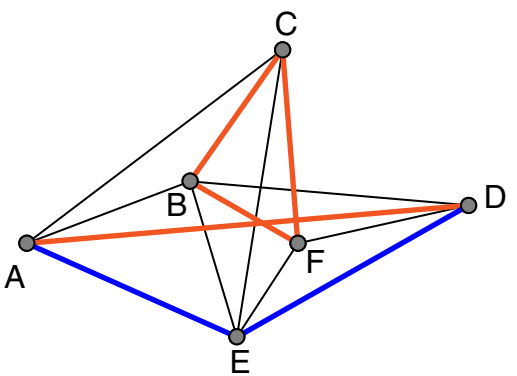
Examples for Search Problems

Traveling Salesman Problem (continued)

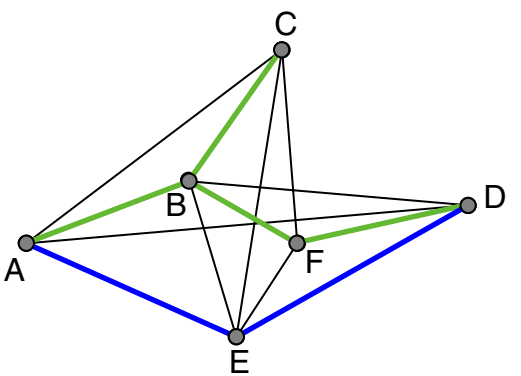


Current State

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*: 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

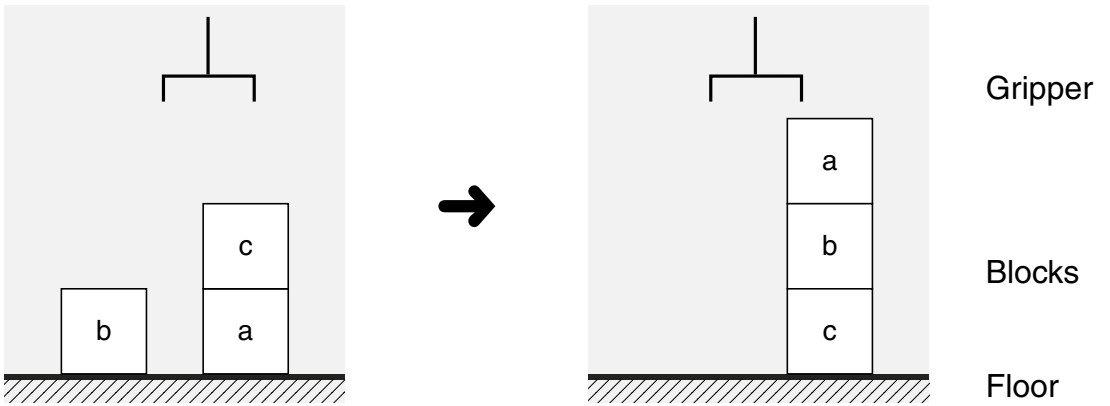
1. Encoding of solution candidates:
   $(v_i)_{i=0}^{N}$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$
2. Encoding of partial solutions:
   $(v_i)_{i=0}^{N}$ with $v_0 = A$, v_i nodes in G, $N \in \mathbf{N}$ (initial part to be continued)
3. Operators:
   Extend a path by appending some adjacent node.
4. Heuristic:
   Try a most promising node first.

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

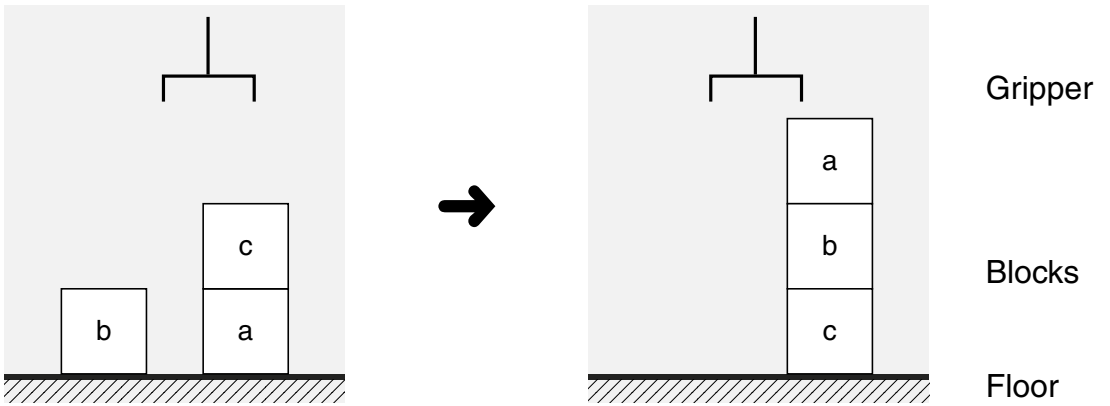
Blocks World Planning



A gripper moves single blocks (clear blocks with no other block on top) to new positions either on top of a clear block or on the floor.

Examples for Search Problems

Blocks World Planning



A gripper moves single blocks (clear blocks with no other block on top) to new positions either on top of a clear 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:
   Extend the sequence of gripper operations by some possible next operation.
4. Heuristic:
   Try a most promising next operation first.

Remarks:

- Logical descriptions of Blocks World Planning states use predicates "*ontop*(x, y)" and "*onfloor*(x)" for the position of blocks.
- The problem instance 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 subgoals cannot be appended to each other to form a solution of the depicted instance.

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:
   Extend the sequence of states by a state reachable via a next transition.
4. Heuristic:
   Try a most promising next transition first.

Remarks:

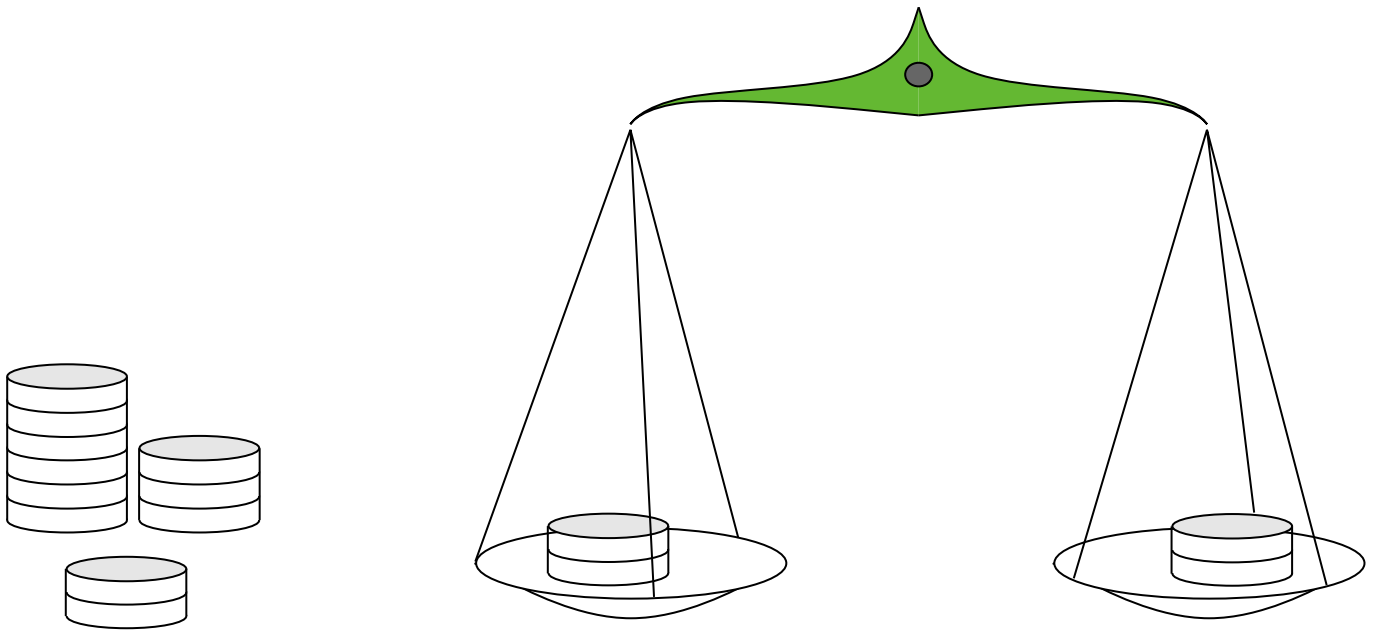
- In planning, description languages are used to specify the state transition systems underlying a planning problem. A commonly used example is the Planning Domain Definition Language (PDDL), which was inspired by STRIPS and ADL.

  States are characterized by their properties and transitions (or actions) describe the changes in the properties of a state.

- In all above examples, a solution is incrementally constructed as a sequence of operator applications (next move, next town, . . .).

Examples for Search Problems

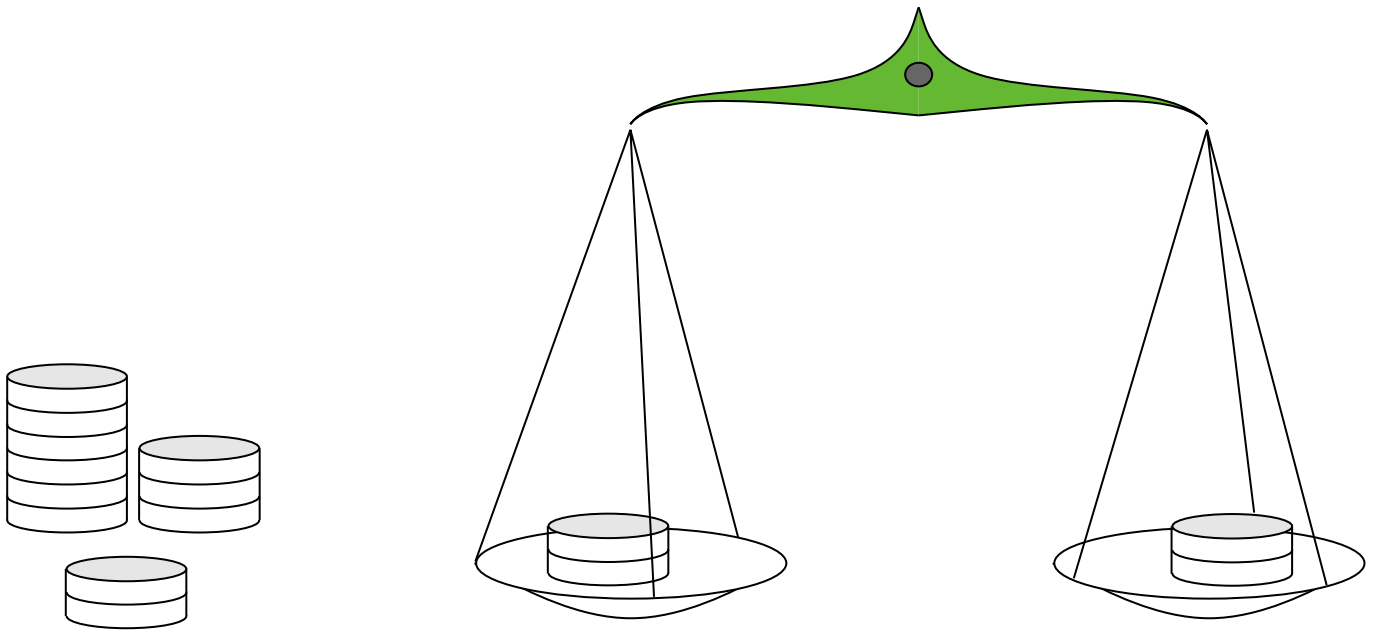
The Counterfeit Coin Problem



- 12 coins are given, one of which is known to be heavier or lighter.
- 3 weighing tests are allowed to find the counterfeit coin.

Examples for Search Problems

The Counterfeit Coin Problem



- Sought: a weighing strategy
  What to weigh first, second, etc.?
  How to process different weighing outcomes?
- Problem decomposition: different weighing outcomes are handled as independent cases.
- The role of heuristics: focus on the most promising weighing strategy.

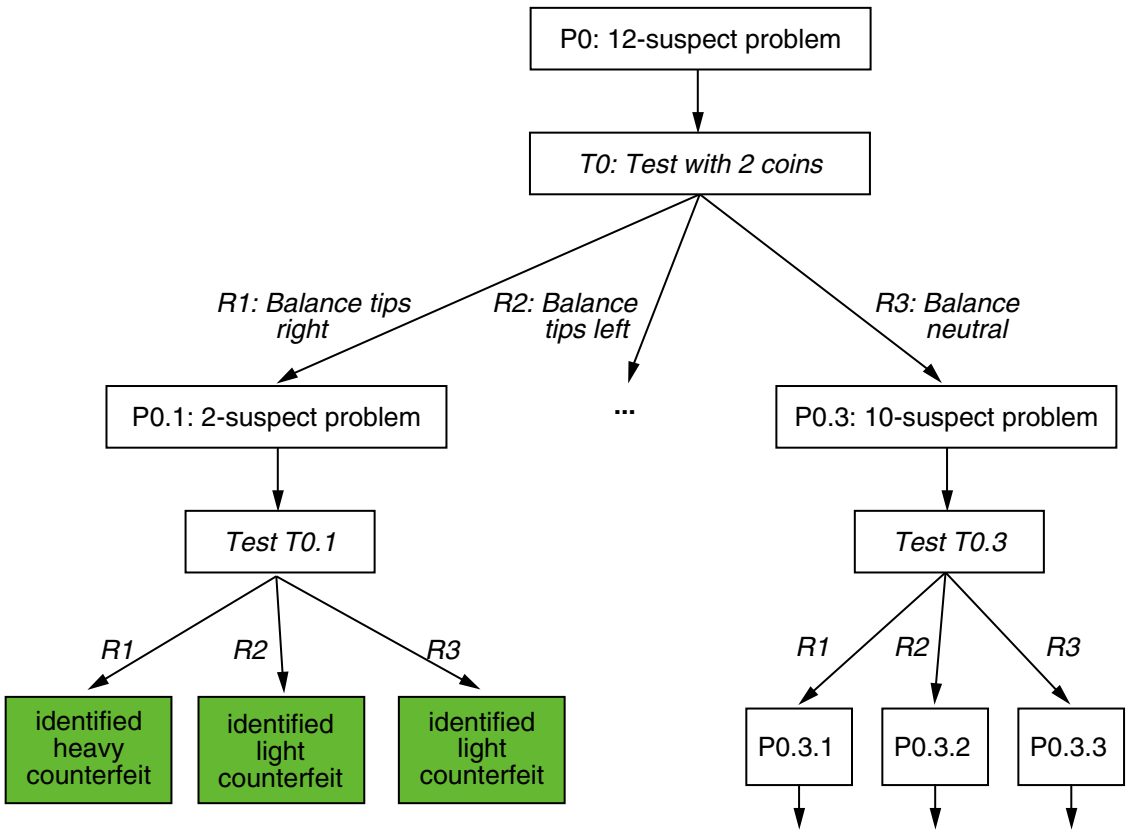
Remarks:

- Each rest problem (specified as state) can be described by knowledge already acquired by the preceding tests. A coin belongs to one of the following four categories: not-heavy, not-light, not-suspect, unknown (= none of the first three categories).
- A weighing action is the selection of a number of coins (for each pan the same number) and to test the selection.
- After each test, the category information of each coin is updated. For instance, if some coin was assigned to “not-light” and from the weighing outcome follows “not-heavy”, its new category becomes “not-suspect”.
- Each weighing action leads to one of three possible outcomes all of which must be dealt with. The most suitable weighing strategy depends on the objective to be optimized:
  1. If the maximum number of weighing actions is to be minimized, treat the most difficult outcome next.
  2. If the expected number of weighing actions is to be minimized, treat the most likely outcome next.

Examples for Search Problems

The Counterfeit Coin Problem (continued)

Representation of a weighing strategy



- Challenge: What is appropriate information about the coins to represent the remaining problems?

Remarks:

- Weighing strategies can be represented as trees (or acyclic directed graphs).
- For a problem at hand, we first constrain the problem by deciding which test to perform and then we consider the possible outcomes of that test (the remaining problems).
- If the balance tips to one side, the counterfeit coin is among the coins just weighed. In the example depicted, test $T0.1$ is then to weigh one of the two "suspect" coins against one of the remaining ten "not-suspect" coins.
- Following a weighing strategy, the subset of "suspect" coins (and "not-light", "not-heavy", "not-suspect") changes over time. For a test, the participants are selected randomly from these sets.
- Of course, more than two coins can be used in a weighing.

Examples for Search Problems

The Counterfeit Coin Problem (continued)

Problem: The Counterfeit Coin Problem

Instance: C. A finite set of coins containing exactly one counterfeit coin.

Solution: A weighing strategy that identifies the counterfeit coin and its failing ("too-light" or "too-heavy").

Examples for Search Problems

The Counterfeit Coin Problem (continued)

Problem: The Counterfeit Coin Problem

Instance: C. A finite set of coins containing exactly one counterfeit coin.

Solution: A weighing strategy that identifies the counterfeit coin and its failing ("too-light" or "too-heavy").

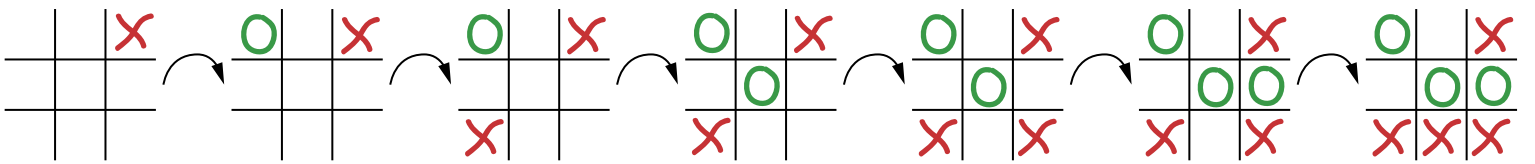
Algorithmization

1. Encoding of solution candidates:
   Finite trees of the above type with tip nodes in which the counterfeit coin and its failing are known ($=$ weighing strategies).
2. Encoding of partial solutions:
   Finite trees of the above type without tip node condition.
3. Operators:
   Extend a branch by a weighing decision node and the branching nodes for its possible outcomes.
4. Heuristic:
   Extend a most promising partial weighing strategy first.

Examples for Search Problems

Tic-Tac-Toe Game [Wikipedia]

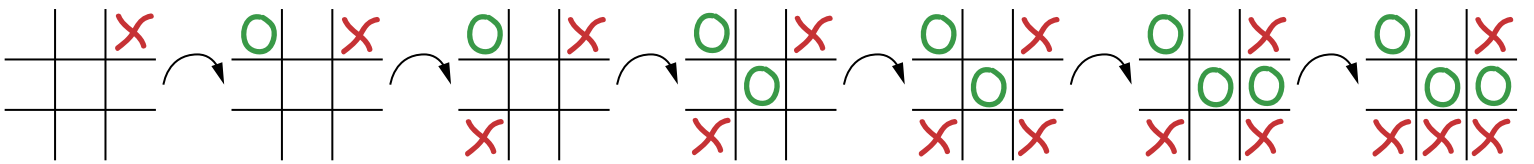
Tic-tac-toe is a game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.



Examples for Search Problems

Tic-Tac-Toe Game [Wikipedia]

Tic-tac-toe is a game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

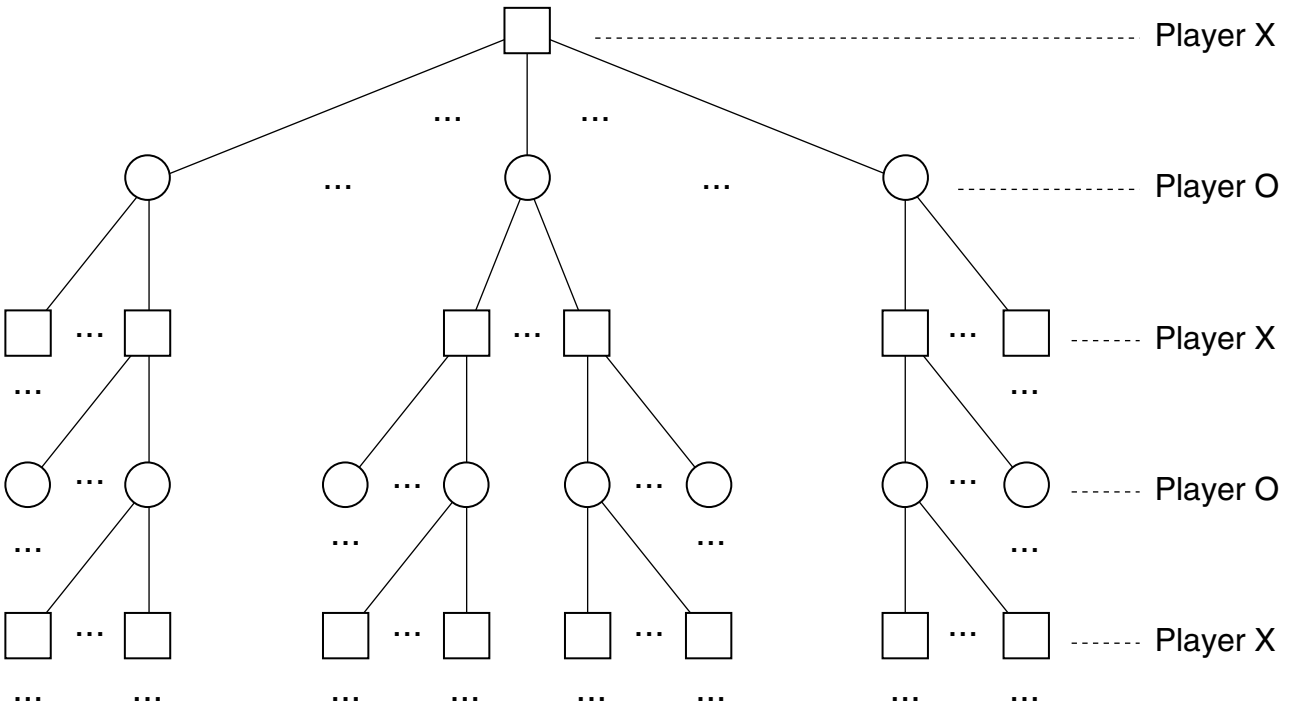


- Sought: a winning strategy for the first player (X)
  Where to put the marks, first, second, etc.?
  How to react to different moves of the opponent?
- Problem decomposition: different possible moves are handled as independent cases.

Examples for Search Problems

Tic-Tac-Toe Game (continued)

Representation of a game tree



- Challenge: Game trees can be huge. Therefore, game trees can usually not be searched completely.

Examples for Search Problems

Tic-Tac-Toe Game (continued)

Problem: Tic-Tac-Toe Game

Instance: Empty 3×3 grid, player X goes first.

Solution: A winning strategy for player X.

Examples for Search Problems

Tic-Tac-Toe Game (continued)

Problem: Tic-Tac-Toe Game

Instance: Empty 3×3 grid, player X goes first.

Solution: A winning strategy for player X.

Algorithmization

1. Encoding of solution candidates:
   Finite trees of the above type with tip nodes in which the outcome of the game (win/draw/loss) is clear.
2. Encoding of partial solutions:
   Finite trees of the above type without tip node condition.
3. Operators:
   Extend a branch by a possible move of the next player.
4. Heuristic:
   Extend a most promising partial game strategy first.

Search Problem Abstraction

Questions

- What is the original problem?
- What is the structure of a solution candidate?
- Which operations (solution steps) are possible to simplify the problem?
- Which rest problems result from such problem simplifications?

➜ How can the original problem be modeled by adequate abstraction and encoding in such a way that efficient search algorithms can be formulated?