Chapter S:IV

IV. Search Space Representation

- Problem Solving
- Systematic Search
- Search Space Encoding
- State-Space Representation

- Problem-Reduction Representation
- Choosing a Representation
- Relation to Dynamic Programming

Problem Solving

Problem Characterization

Problem Types:

> *We call a problem well-defined if there is a test which can be applied to a proposed solution. In case the proposed solution is a solution, the test must confirm this in a finite number of steps.*
>
> [John McCarthy, 1956, Wikipedia]

Assuming a problem implicitly described by discrete states and operators, McCarthy defines two classes of problems:

- Well-defined problems:
  There is complete knowledge of the initial state, goal states, and the operators.

- Ill-defined problems:
  The knowledge of the goal state or the necessary operators is incomplete.
  Example: Paint a lovely picture.

Remarks:

- General speaking, there is a problem continuum whose extremal points are designated as *closed (well-defined) problems* and *open (ill-defined) problems* respectively. [Dörner 1979]
- Assuming a problem implicitly described by discrete states and operators, a solution is a sequence of actions that leads from the initial state to a goal state.
- Consider the following problem P:

  For a given $n \in \mathbf{N}$, is there a continuous sequence of at least n digits "5" in the decimal representation of $\pi = 3.1415...$?

  The characteristic function $\chi_P : \mathbf{N} \rightarrow \{0, 1\}$ of this problem is computable.

  Q. Is P a well-defined problem?

Problem Solving

Problem Characterization

Dörner proposes a classification of problems according to the type of barriers that must be overcome for problem solving [Dörner 1979, Wikipedia] :

- Interpolation barrier:
  There is complete knowledge of the initial state, goal states, and the operators that change states. Searched is a particular operator combination or an operator sequence.

  Example: Traveling Salesman Problem.

- Synthesis barrier:
  The knowledge of available operators is incomplete.

  Example: Find a recursion scheme for a finite number sequence and give its next element.

- Dialectic barrier:
  The criteria that qualify solution candidates as solutions are unclear.

Remarks:

- Interpolation barriers must be overcome when dealing with closed problems; synthesis barriers as well as dialectic barriers must be overcome when dealing with open problems.
- For some problems, both a synthesis barrier and a dialectical barrier have to be overcome if neither the goal nor the means to achieve it are clear.
- Of course, the type of barrier depends on the knowledge available. What is a synthesis barrier or a dialectical barrier for an economist can be an interpolation barrier for a mathematician.

Problem Solving

Problem Characterization

Problem Areas in Artificial Intelligence

> *Along with the development of general-purpose computers, the past few years have seen an increase in effort toward the discovery and mechanization of problem solving processes.* [. . .] *In this article, an attempt will be made to separate out, analyze, and find the relations between some of these problems.* [. . .] *It is convenient to divide the problems into five main areas:*
>
> - *Search,*
> - *Pattern-Recognition,*
> - *Learning,*
> - *Planning, and*
> - *Induction.*
>
> [Marvin Minsky, 1961, Wikipedia]

In the discussion of problem solving, Minsky assumes that, as a rule, all problems to be solved are well-defined.

Problem Solving

Decision Problem: Abstract Definition

A *decision problem* Π consists of a set D_Π of finite objects called *instances* of Π and a set $Y_\Pi \subseteq D_\Pi$.

An algorithm is said to *decide* a decision problem Π if, given as input any instance $x \in D_\Pi$, it returns the answer "yes" whenever $x \in Y_\Pi$ holds and otherwise returns the answer "no".

[Garey, Johnson 1979]

- Usually, Y_Π is defined implicitly: "instances in D_Π that have some property".
- For a given problem instance, a decision algorithm decides whether an instance has the desired property.
- Example: The TSP Decision Problem "Is there a Hamiltonian cycle in a given graph?"

Remarks:

- Algorithms use some encoding of the problem instance. Therefore, the first step of an algorithm is to decide whether its input represents a legal encoding of a problem instance.

  For simplicity, the problem of legal encodings is assumed to be decidable.

Problem Solving

Search Problem: Abstract Definition

A *search problem* Π consists of a set D_Π of instances of Π and, for each instance $x \in D_\Pi$, a set $S_\Pi(x)$ of solutions for x.

An algorithm is said to *solve* a search problem Π if, given as input any instance $x \in D_\Pi$, it returns the answer "no" whenever $S_\Pi(x)$ is empty and otherwise returns some solution y belonging to $S_\Pi(x)$.

[Garey, Johnson 1979]

- It is assumed that for any x and y the question "$y \in S_\Pi(x)$" is decidable. The search space is the set of candidates from which $S_\Pi(x)$ was drawn.
- For a given problem instance, a search algorithm
  1. decides whether a solution exists and,
  2. in case "yes", returns a solution as a witness.
- Example: The TSP Search Problem "Search for a Hamiltonian cycle in a given graph."

Remarks:

- For a problem instance x, the search space is a set of objects that share structural properties with the solutions in set $S_\Pi(x)$. In complexity theory, the search space is finite or at least enumerable.

- Distinguish decision problems and search problems, e.g. for SAT:

Problem:	SAT
Input:	Propositional formula α in CNF
Decision Problem:	Is α satisfiable?
Search Problem:	Find a satisfying assignment to α if one exists, else output “no”.

  Q. Can we solve the search problem for SAT in polynomial time if an oracle for the decision problem is given?

- Consider reachability of a goal state in a state transition system as search problem Π.

  Q. What are the instances in Π and why are they finite objects?

Remarks: (continued)

- In Complexity Theory, search problems are modeled by string relations.

  A *string relation* $R \subseteq \Sigma^+ \times \Sigma^+$ is *realized* by a function $f : \Sigma^* \to \Sigma^*$ if and only if, for each $x \in \Sigma^+$, we have

  $$f(x) = \begin{cases} \varepsilon & \text{whenever there is no } y \in \Sigma^+ \text{ with } (x, y) \in R, \\ y & \text{for some } y \in \Sigma^+ \text{ with } (x, y) \in R \text{ otherwise.} \end{cases}$$

  A Turing machine M *solves* R if the function f_M computed by M realizes R.

  For some encoding e, a search problem Π corresponds to a string relation $R_e[\Pi]$ defined by

  $$R_e[\Pi] = \{(e(x), e(y)) \mid x \in D_\Pi \text{ and } y \in S_\Pi(x)\}$$

  [Garey, Johnson 1979]

- A Turing machine M
  - decides whether a proper encoding of an instance is given and,
  - in case “yes”, decides whether a solution exists and,
  - in case “yes” again, returns an encoding of a solution as a witness.

Problem Solving

Optimization Problem: Abstract Definition

An optimization problem Π consists of a set D_Π of instances and, for each instance $x \in D_\Pi$, a set $S_\Pi(x)$ of solutions for x. A computable function m_Π assigns to x and y a positive rational value $m_\Pi(x, y)$, the *solution value*. The optimization goal is minimization (or maximization) of m_Π values.

An algorithm is said to *solve* an optimization problem Π if, given as input any instance $x \in D_\Pi$, it returns the answer "no" whenever $S_\Pi(x)$ is empty and otherwise returns some solution y belonging to $S_\Pi(x)$ such that

$$m(x, y) = \min\{m(x, y') \mid y' \in S_\Pi(x)\}$$

[Garey, Johnson 1979]

- For a given problem instance, an optimization algorithm
  1. decides whether a solution exists and,
  2. in case "yes", returns a solution with minimum (maximum) m value as witness.
- Example: The TSP Optimization Problem "Search for a Hamiltonian cycle of minimum length in a given graph."

Remarks:

- The function m_Π is also called *objective function*.
- There are three important variants of optimization problems:
  - Constructive Problem Π:
    Input: $x \in D_\Pi$
    Output: $y \in S_\Pi(x)$ such that $m(x, y) = \min\{m(x, y') \mid y' \in S_\Pi(x)\}$ if solutions exist, "no" otherwise.
  - Evaluation Problem Π:
    Input: $x \in D_\Pi$
    Output: $\min\{m(x, y') \mid y' \in S_\Pi(x)\}$ if solutions exist, "no" otherwise.
  - Decision Problem Π:
    Input: $x \in D_\Pi$ and $k \in \mathbf{N}$
    Output: "yes" if $k \geq \min\{m(x, y') \mid y' \in S_\Pi(x)\}$, "no" otherwise.

    (the case $\max$ is analogous)

Chapter S:IV

IV. Search Space Representation

- Problem Solving
- Systematic Search
- Search Space Encoding
- State-Space Representation

- Problem-Reduction Representation
- Choosing a Representation
- Relation to Dynamic Programming

Systematic Search

Search Building Blocks

Given a problem instance and the corresponding search space S with solution candidates. Then *problem solving* means to find or to construct an object with given characteristics in S.

Prerequisites to “algorithmize” problem solving:

1. A symbol structure or code to represent each object in S.
2. Computational tools to transform encodings of some objects in S into encodings of some other objects in S.
3. An effective method to schedule (= order) transformations. Objective is to maximize effectiveness: find the desired object as quickly as possible.

Systematic Search

Search Building Blocks

Given a problem instance and the corresponding search space S with solution candidates. Then *problem solving* means to find or to construct an object with given characteristics in S.

Prerequisites to “algorithmize” problem solving:

1. A symbol structure or code to represent each object in S.
2. Computational tools to transform encodings of some objects in S into encodings of some other objects in S.
3. An effective method to schedule (= order) transformations. Objective is to maximize effectiveness: find the desired object as quickly as possible.

Synonymous terms from the field of Artificial Intelligence (AI) :

1. database or knowledge base,
2. operators or production rules,
3. control strategy.

Systematic Search

Definition 1 (Systematic Control Strategy, Systematic Search)

Given a search space S with solution candidates. A control strategy is called *systematic* if

(a) all objects in S are considered,

(b) each object in S is considered only once.

A search that employs a systematic control strategy is called *systematic search*.

Remarks:

- Condition (a) implies completeness, condition (b) implies efficiency.
- Of course, Condition (a) of a systematic search is particularly important if solutions are rare. No part of the search space should be excluded a priori.

Systematic Search

Categorization of Heuristic [Problem Solving] Methods [Zanakis, 1989]

A. Construction

Construction algorithms generate a solution by adding individual components one at a time until a feasible solution is obtained.

B. Improvement

Improvement heuristics begin with a feasible solution and successively improve it by a sequence of exchanges or mergers in a local search.

C. Mathematical Programming

Mathematical programming approaches use a mathematical optimization model and an exact solution procedure, then modify this solution procedure to obtain an efficient heuristic for the problem.

D. Decomposition

Decomposition refers to solving a sequence of tractable smaller problems, the output of one being the input to the next, and then inductively merging these solutions.

Systematic Search

Categorization of Heuristic [Problem Solving] Methods (continued)

E. Partitioning or Decomposition [Pearl]
   Partitioning algorithms break or 'partition' a problem into smaller subproblems, each of which is solved independently.

F. Solution Space Restriction
   The idea here is to restrict the set of solutions so that a problem becomes easier to solve.

G. Relaxation
   This category refers to expanding the solution space in order to obtain a tractable problem.

Chapter S:IV

IV. Search Space Representation

- Problem Solving
- Systematic Search
- Search Space Encoding
- State-Space Representation
- Problem-Reduction Representation
- Choosing a Representation
- Relation to Dynamic Programming

Search Space Encoding

Encoding of Solution Candidates

Basic Approach: Disregarding common structural properties.

The encoding treats only single solution candidates.

Examples: sequences of legitimate moves for the 8 puzzle, legitimate TSP tours

In the 8-queens problem:

Candidate encoding: (A1, A2, A3, A4, A5, A6, A7, A8) ,
(A1, A2, A3, A4, A5, A6, A7, B8) ,
...
(H1, H2, H3, H4, H5, H6, H7, H8)

Operators: e.g. moving a queen to the left, to the right, up, or down

Search Space Encoding

Encoding of Solution Candidates

Basic Approach: Disregarding common structural properties.

The encoding treats only single solution candidates.

Examples: sequences of legitimate moves for the 8 puzzle, legitimate TSP tours

In the 8-queens problem:

Candidate encoding: (A1, A2, A3, A4, A5, A6, A7, A8) ,
(A1, A2, A3, A4, A5, A6, A7, B8) ,
...
(H1, H2, H3, H4, H5, H6, H7, H8)

Operators: e.g. moving a queen to the left, to the right, up, or down

Improvement: Place only one queen per column and row, apply a lexical sorting.

Candidate encoding: (A1, B2, C3, D4, E5, F6, G7, H8) ,
...
(A8, B7, C6, D5, E4, F3, G2, H1)

Operators: e.g. switching columns of two queens

Remarks:

- Distinguish between the objects (solution candidates) in S and their encodings. An encoding can be understood as a reference to an object or set of objects in S.
- Intermediate improvement for encoding of solution candidates in the 8-queens problem: Start with a board with 8 queens (one per row, lexical sorting) and devise 16 operators (two for each queen) which allow for moving a queen either to the left or to the right.

  Q. Does this encoding allow a systematic search?
- Disregarding structural properties typically looks as follows. We devise an encoding for the representation of solution candidates, along with operators that allow for arbitrary modifications. Then, the search space corresponds to a complete graph, and the control strategy can be realized by an enumeration procedure. [Hooke/Jeeves 1961]
- Genetic algorithms work on a population of individuals. Operators are crossover and mutation for generating new individuals and deletion of individuals from the population.
- If an encoding for solution candidates is given, it is always possible to construct an enumeration of legal encodings (i.e., encodings for solution candidates): add an ordering for symbols, extend this ordering to strings of symbols, generate symbol strings increasing length, test for legal encodings.

  Such an enumeration is complete, because an encoding must be able to represent any solution candidate. But it is efficient only if there is at most one encoding for each solution candidate.

Search Space Encoding

Applicable Heuristic Problem Solving Methods [Zanakis, 1989]

A. Construction

Construction algorithms generate a solution by adding individual components one at a time until a feasible solution is obtained.

B. Improvement

Improvement heuristics begin with a feasible solution and successively improve it by a sequence of exchanges or mergers in a local search.

➜ Local search algorithms like Hill-Climbing

Systematic control strategy: enumeration of candidates.

E. Partitioning

Partitioning algorithms break or 'partition' a problem into smaller subproblems, each of which is solved independently.

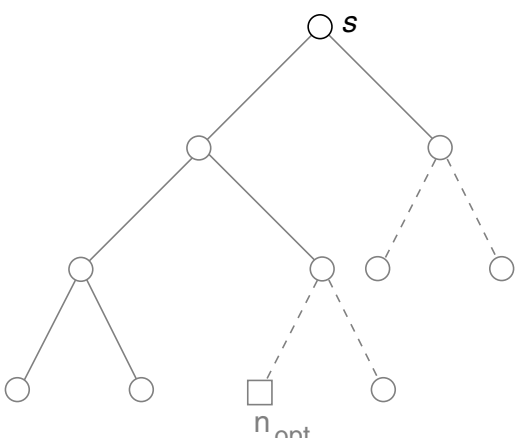
Search Space Encoding

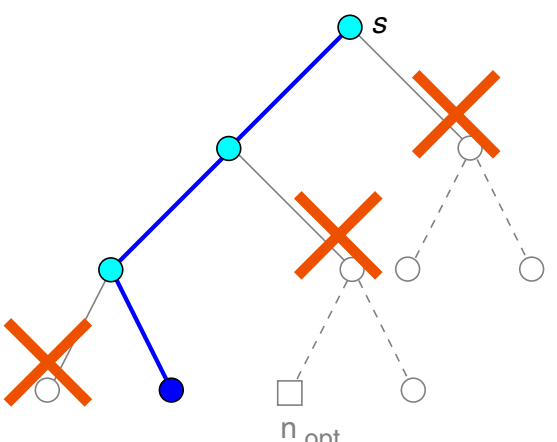
OPEN List Restriction: Hill-Climbing (HC) [BF* Variants: DFS, BT]

Hill-climbing is an informed, irrevocable search strategy.

HC characteristics:

- local or greedy optimization:
  take the direction of steepest ascend (merit) / steepest descend (cost)
- "never look back" :
  alternatives are not remembered ➜ no OPEN/CLOSED lists
- usually low computational effort
- a strategy that is often applied by humans





Search Space Encoding

Hill-Climbing

Algorithm:	HC. Adaption for OR graph search
Input:	s. Start node representing the initial problem. *successors*(n). Returns the successors of node n. $\star(n)$. Predicate that is *True* if n represents a solution path. $h(n)$. Heuristic cost estimation for the remaining problem defined by node n.
Output:	A node γ representing a solution path or the symbol *Fail*.

Search Space Encoding

Hill-Climbing

Algorithm: HC. Adaption for OR graph search

Input: s. Start node representing the initial problem.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n represents a solution path.
$h(n)$. Heuristic cost estimation for the remaining problem defined by node n.

Output: A node γ representing a solution path or the symbol *Fail*.

HC$(s,$ *successors*$, \star, f)$

```
1.  n = s;
2.  n_opt = s;
3.  LOOP
4.    IF ⋆(n) THEN RETURN(n);
5.    FOREACH n' IN successors(n) DO   // Expand n.
        set_backpointer(n', n);
        IF (h(n') < h(n_opt))   // Compare optimum remaining cost estimates.
        THEN n_opt = n';   // Remember optimum successor.
      ENDDO
6.    IF (n_opt = n)
      THEN RETURN(Fail);   // We could not improve.
      ELSE n = n_opt;   // Continue with the best successor.
7.  ENDLOOP
```

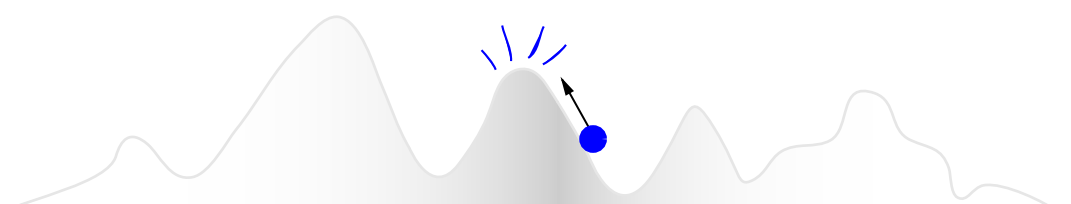
Search Space Encoding

Hill-Climbing: Discussion

Hill-Climbing issue:

The first property of a systematic control strategy, *"Consider all objects in S."*, is violated by hill-climbing if no provisions are made.

- The forecast of the evaluation function (cost function, merit function) may be—at least sometimes—wrong and misguiding the search.
- Search will probably terminate at a local optimum.
- Alternative paths are not considered since each step is irrevocable.



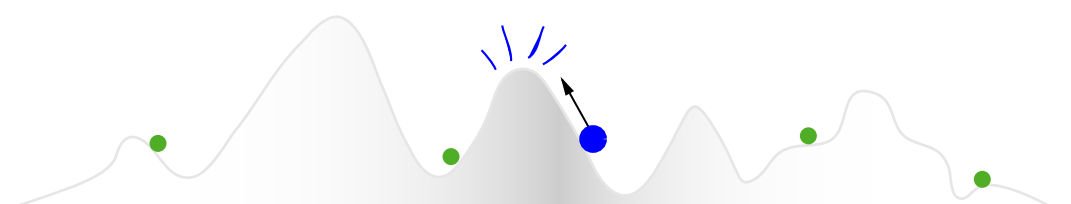
Search Space Encoding

Hill-Climbing: Discussion

Hill-Climbing issue:

The first property of a systematic control strategy, *"Consider all objects in S."*, is violated by hill-climbing if no provisions are made.

- The forecast of the evaluation function (cost function, merit function) may be—at least sometimes—wrong and misguiding the search.
- Search will probably terminate at a local optimum.
- Alternative paths are not considered since each step is irrevocable.



Workaround: Perform multiple restarts (e.g. random-restart hill-climbing).

Workaround issue: The second property of a systematic control strategy, *"Consider each object in S only once."*, is violated if no provisions are made.

Search Space Encoding

Hill-Climbing: Discussion (continued)

Hill-climbing can be the favorite strategy in certain situations:

(a) We are given a highly informative evaluation function to control search.

(b) The operators are commutative. Commutativity is given, if all operators are independent of each other.

➜ The application of an operator will

1. neither prohibit the applicability of any other operator,
2. nor modify the outcome of its application.

Example: Expansion of the nodes in a complete graph.

Remarks:

- Given commutativity, an irrevocable search strategy can be applied without hesitation: finding the optimum may be postponed but is never prohibited. Keywords: *greedy algorithm, greedy strategy, matroid*
- Given commutativity, hill-climbing can be considered a systematic strategy.
- Typically, hill-climbing is operationalized as an *informed strategy*, i.e., information about the goal (or about a concept to reach the goal) is exploited. If such external or look-ahead information is not exploited, hill-climbing must be considered an uninformed strategy.
- Q. What could be a provision to avoid a violation of the second property of a systematic control strategy?

Search Space Encoding

Encoding of Solution Candidates (continued)

Advanced Approach: Exploiting common structural properties. An encoding for solution candidates is used that allows a compact representation of *sets of solution candidates*: subset encoding.

In the 8-queens problem:
Subset encoding: (A2, B5, C3) as shorthand for

{ (A2, B5, C3, D1, E4, F6, G7, H8) ,
(A2, B5, C3, D1, E4, F6, G8, H7) ,
. . . ,
(A2, B5, C3, D8, E7, F6, G4, H1) }

Structural property: Encodings of common parts can represent the set of solution candidates sharing this part.

Search Space Encoding

Encoding of Solution Candidates (continued)

Advanced Approach: Exploiting common structural properties. An encoding for solution candidates is used that allows a compact representation of *sets of solution candidates*: subset encoding.

In the 8-queens problem:
Subset encoding: (A2, B5, C3) as shorthand for

{ (A2, B5, C3, D1, E4, F6, G7, H8) ,
(A2, B5, C3, D1, E4, F6, G8, H7) ,
. . . ,
(A2, B5, C3, D8, E7, F6, G4, H1) }

Structural property: Encodings of common parts can represent the set of solution candidates sharing this part.

Manipulation of subsets:

- Split: Partition a subset by increasing the common part in all possible ways.
- Prune: The entire subset is pruned, because no solution is contained.
  E.g., singleton sets or sets, where some common property already disqualifies all contained solution candidates.

Remarks:

- A slightly less compact encoding (A2, *, C3, D6, *, *, G5, *) is more flexible in representing subsets of solution candidates.
- An encoding should take advantage of the structural properties of a problem domain. Often, structural properties are captured by a kind of incomplete or partial representation of an object—as opposed to its complete or unique representation.

  An encoding of the latter kind should correspond one-to-one to an object (a solution candidate) in S. In the 8-queens problem, for example, such an encoding refers to a board configuration with exactly 8 queens.

Remarks: (continued)

- An encoding for the representation of subsets $S' \subset S$ gives rise to the *principle of refinement*: operators are employed to (large) solution bases towards more specialized solution bases or towards solution candidates.

- Refinement (of the encoding) of solution bases in the 8-queens problem:

  (A2, B5, C3, *, *, *, *, *) $\rightarrow$ (A2, B5, C3, D1, *, *, *, *)

  where (consider the shorthand semantics introduced before)

  (A2, B5, C3, D1, *, *, *, *) $\subset$ (A2, B5, C3, *, *, *, *, *)

- Observe the ambiguous usage and semantics of the term "solution":
  1. Solution candidates may represent a legitimate solution or not. Example: a board with 8 queens which, however, may attack each other.
  2. Legitimate solutions fulfill the problem-specific constraints. Examples: a board with 8 queens which do not attack each other, a legitimate TSP tour.
  3. Optimum solutions fulfill the problem-specific constraints and stand out regarding a distinguished property. Example: a shortest TSP tour.

  Recall that optimization problems may not have a unique solution.

Search Space Encoding

Encoding of Solution Candidates (continued)

Systematic search requirements in the context of subset encoding:

(a) Completeness.

- Each individual solution candidate is included in the collection of subsets that can be considered.
- Let $S'_1, \ldots, S'_k$ be the possible outcomes after a refinement of S' with $S' \subset S$. Then each object (solution candidate) in S' must be a member in some set S'_i, $1 \leq i \leq k$.
- Each individual solution candidate in a subset can be reached (as singleton set) by a finite sequence of refinement operations on that set.

(b) Efficiency.

- If a set $S' \subset S$, has been excluded during search, no solution candidate from S' may be element of those subsets $S'' \subset S$ that are still under consideration.

➜ A search method based on split-and-prune is easily shown to be systematic.

Remarks:

- It is obvious that *subset splittings* are the only operators that can fulfill both conditions for a systematic search. A dead end is reached if after a splitting a generated subset does not contain any solution.
- Systematic search by split-and-prune in the 8-queens problem:
  - Completeness:
    Let S^{Ai} denote (Ai, *, *, *, *, *, *, *), with only one queen per column and row, and under a lexical sorting. Then, for example, $S^{\text{A1}} \cup S^{\text{A2}} \cup \ldots \cup S^{\text{A8}}$ must contain all solution candidates.
  - Efficiency
    A solution candidate with a queen placed on A1 cannot be generated from $S^{\text{A2}} \cup \ldots \cup S^{\text{A8}}$.
- Observe how the encoding of solution bases along with the split-and-prune paradigm ensure search efficiency: no bookkeeping is required to check which of the solution candidates in S have already been considered or not been considered.

Search Space Encoding

Subset Encoding for the 8-Queens Problem

Structural properties of solution candidates:

1. Solution candidates are vectors of queen positions.
2. Positions are sorted.
3. There is at most one queen per row.
4. There is at most one queen per column.

Observations for subset splitting w.r.t. next queen's position:

- ❑ Subset splitting prevents the generation of already considered board configurations.
- ❑ Subset splitting prevents to skip non-considered board configurations.
- ❑ The encoding of solution bases realizes the enumeration of all board configurations.
- ❑ The encoding of solution bases enables both constraint checks and dead end recognition over (large) subsets of solution candidates.

Search Space Encoding

Subset Encoding for the 8-Queens Problem

Structural properties of solution candidates:

1. Solution candidates are vectors of queen positions.
2. Positions are sorted.
3. There is at most one queen per row.
4. There is at most one queen per column.

Observations for subset splitting w.r.t. next queen's position:

- ❑ Subset splitting prevents the generation of already considered board configurations.
- ❑ Subset splitting prevents to skip non-considered board configurations.
- ❑ The encoding of solution bases realizes the enumeration of all board configurations.
- ❑ The encoding of solution bases enables both constraint checks and dead end recognition over (large) subsets of solution candidates.

Consider the search space S and the subset (A2, B5, C3):

Q. What is S and what is a subset splitting of (A2, B5, C3) when exploiting properties 1. - 4. ?

Q. What is S and what is a subset splitting of (A2, B5, C3) when exploiting properties 1. - 3. ?

Search Space Encoding

Subset Encoding for the 8-Puzzle Problem

Structural properties of solution candidates:

1. Solution candidates are finite sequences of moves.
2. The only possible moves are $\{\text{up}, \text{down}, \text{left}, \text{right}\}$.
3. A move is legal for the board configuration resulting from the previous moves.

Observations:

- By appending further moves to the sequence, the subset S' is narrowed.
- The encoding realizes subset splitting, and hence permits to prune hopeless sequences—at least theoretically.
- The number of solution candidates exceeds the number of possible board configurations.

Search Space Encoding

Subset Encoding for the 8-Puzzle Problem

Structural properties of solution candidates:

1. Solution candidates are finite sequences of moves.
2. The only possible moves are $\{\text{up}, \text{down}, \text{left}, \text{right}\}$.
3. A move is legal for the board configuration resulting from the previous moves.

Observations:

- By appending further moves to the sequence, the subset S' is narrowed.
- The encoding realizes subset splitting, and hence permits to prune hopeless sequences—at least theoretically.
- The number of solution candidates exceeds the number of possible board configurations.

Consider the search space S and the subset (left, up, up) for some initial board configuration s.

Q. What is S and what is a subset splitting of (left, up, up) when exploiting properties 1. - 3. ?

Q. What is S and what is a subset splitting of (left, up, up) when exploiting properties 1. - 2. ?

Q. What is the problem with subset pruning?

Remarks:

- By modifying the problem (instead of by modifying the encoding), a dead end handling can be realized in the 8-puzzle problem:
  - The goal state γ must be reached from s within less then K moves.
  - The heuristics $h_1(x)$ and $h_2(x)$ can be applied for pruning, if the sequence length exceeds $K - h_1(x)$ or $K - h_2(x)$. [S:I Introduction]
  - However, dead ends still cannot be predicted earlier. Q. Why not?
- Note that the computation of the heuristics $h_1(x)$ and $h_2(x)$ may be based on both the tile configuration and the move sequence. Obviously the former is much simpler to analyze than the latter.

Chapter S:IV

IV. Search Space Representation

- Problem Solving
- Systematic Search
- Search Space Encoding
- State-Space Representation
- Problem-Reduction Representation
- Choosing a Representation
- Relation to Dynamic Programming

State-Space Representation

Subset Encoding in Search

Framework for algorithms based on subset encodings:

1. Start with the search space S as *initial state* subset.
2. Loop: While no subset contained an easily detectable solution, do:
   (a) Select a subset S' and perform subset splitting on S'.
   (b) Remove S' and store the subsets resulting from the splitting.

Each subset of solution candidates can be seen as the search space for a problem that is related to the initial problem, but that is more restricted.

State-Space Representation

Subset Encoding in Search

Framework for algorithms based on subset encodings:

1. Start with the search space S as *initial state* subset.
2. Loop: While no subset contained an easily detectable solution, do:
   (a) Select a subset S' and perform subset splitting on S'.
   (b) Remove S' and store the subsets resulting from the splitting.

Each subset of solution candidates can be seen as the search space for a problem that is related to the initial problem, but that is more restricted.

Definition 2 (State, State-Space, State-Space Graph)

Given a search space S with solution candidates. The information that explicitly describes the rest problem associated with a subset $S' \subset S$ is called *state*. The set of states is called *state-space*. The graph with state-space as node set and edges defined by subset splittings is called *state-space graph*.

Remarks:

- Special states:
  - The *initial state* represents the set S of all solution candidates.
  - A *goal state* γ represents a solved or trivial rest problem. A goal state specifies a solution.
- We use the term “trivial rest problem” to express the fact that a solution path may not specify a complete solution. Actually, the leaf node γ of a solution path may still stand for a problem whose solution, however, is already known or can be easily looked-up.

State-Space Representation

Subset Encoding in Search (continued)

> *“The idea of viewing problem solving as a process of repetitive splitting (also called branching or refinement) of subsets of potential solutions became popular in operations research and is the basic metaphor behind the celebrated branch-and-bound method. Artificial intelligence literature rarely mentions this metaphor, preferring to describe problem solving as a "generate-and-test" process, creating new objects rather than eliminating objects from preexisting sets.”*
>
> [Pearl, 1984]

From an artificial intelligence point of view, a sequence of subset splittings describes the solution steps taken to solve the initial problem. The further solution steps use the achieved result as a basis and work on the rest problem.

State-Space Representation

Subset Encoding in Search (continued)

> *"The idea of viewing problem solving as a process of repetitive splitting (also called branching or refinement) of subsets of potential solutions became popular in operations research and is the basic metaphor behind the celebrated branch-and-bound method. Artificial intelligence literature rarely mentions this metaphor, preferring to describe problem solving as a "generate-and-test" process, creating new objects rather than eliminating objects from preexisting sets."*
>
> [Pearl, 1984]

From an artificial intelligence point of view, a sequence of subset splittings describes the solution steps taken to solve the initial problem. The further solution steps use the achieved result as a basis and work on the rest problem.

Definition 3 (Solution Base)

Given a search space S with solution candidates. For $S' \subset S$, the information that explicitly describes the sequence of subset splittings that refined S to S' is called *solution base* or *partial solution*.

Remarks:

- Distinguish:
  - The encoding of a solution base $S' \subset S$ tells us how to reach S'.
  - The state associated with a solution base $S' \subset S$ tells us what still has to be done.
- Of course, the information about the rest problem is always derivable from the encoding of a solution base $S' \subset S$. The key question is, how easy this information can be made explicit and exploited.
- In the 8-puzzle problem:
  - The start configuration s along with a sequence of moves $\{\text{up} \mid \text{down} \mid \text{left} \mid \text{right}\}_0^*$ is sufficient to specify the rest problem. However, making the rest problem explicit—and hence its evaluation—is difficult.
  - The encoding of the move sequence (= solution base) *plus* the information about the resulting puzzle configuration (= state) is redundant. However, evaluating the rest problem is easy.
- The encoding and the state-space complement each other:
  - From the perspective of encoding solution bases, solving a search problem means to apply iterative splitting until we reach a subset wherein the solution can be identified easily.
  - From the perspective of a state-space graph, a solution is not given by a goal node (i.e., a node of a solved or trivial rest problem), but by the path from the initial state to that node.

State-Space Representation

Traveling Salesman Problem

$$\underbrace{A \longrightarrow B \longrightarrow C \longrightarrow}_{\text{Encoding of a solution base: all tours that start with } A, B, C, D.} D \underbrace{\longrightarrow \{E, F\} \longrightarrow A}_{\text{State: explicit description of the rest problem } D, \{E, F\}, A.}$$

- Given the initial problem, the encoding of the solution base is sufficient to describe the rest problem. I.e., the state introduces redundant information.
- The state information is not sufficient to describe the complete situation.
- Explicitly maintaining state information simplifies the computation of heuristics.

State-Space Representation

Traveling Salesman Problem (continued)

$$\underbrace{A \longrightarrow C \longrightarrow B \longrightarrow}_{\text{Encoding of a solution base: all tours that start with } A, C, B, D.} D \underbrace{\longrightarrow \{E, F\} \longrightarrow A}_{\text{State: explicit description of the rest problem } D, \{E, F\}, A.}$$

- We are given the same state as before.
- Recognizing (equal) states enables us to solve problems independently.
- Better solution bases invalidate worse solution bases:
  The best solution base among $A, \{B, C\}, D$ is combined with the solution for the state $D, \{E, F\}, A$.

  Keyword: *Pruning by Dominance*

State-Space Representation

Summary: Solving Search Problems by State Space Search

State Space Graph G	Search Problem
Nodes V State space	Rest problems (defined by subsets of solution candidates)
Start node s Initial state	Initial problem (defined by set of all solution candidates)
Edges E State transitions	Solution steps (refinement of subset in subset splittings)
Goal nodes Γ, $\Gamma \subseteq V$ Goal states	Trivial rest problems (subsets with easily identifiable solutions)
Path from s to $\gamma, \gamma \in \Gamma$, with property $*(.)$	Solution

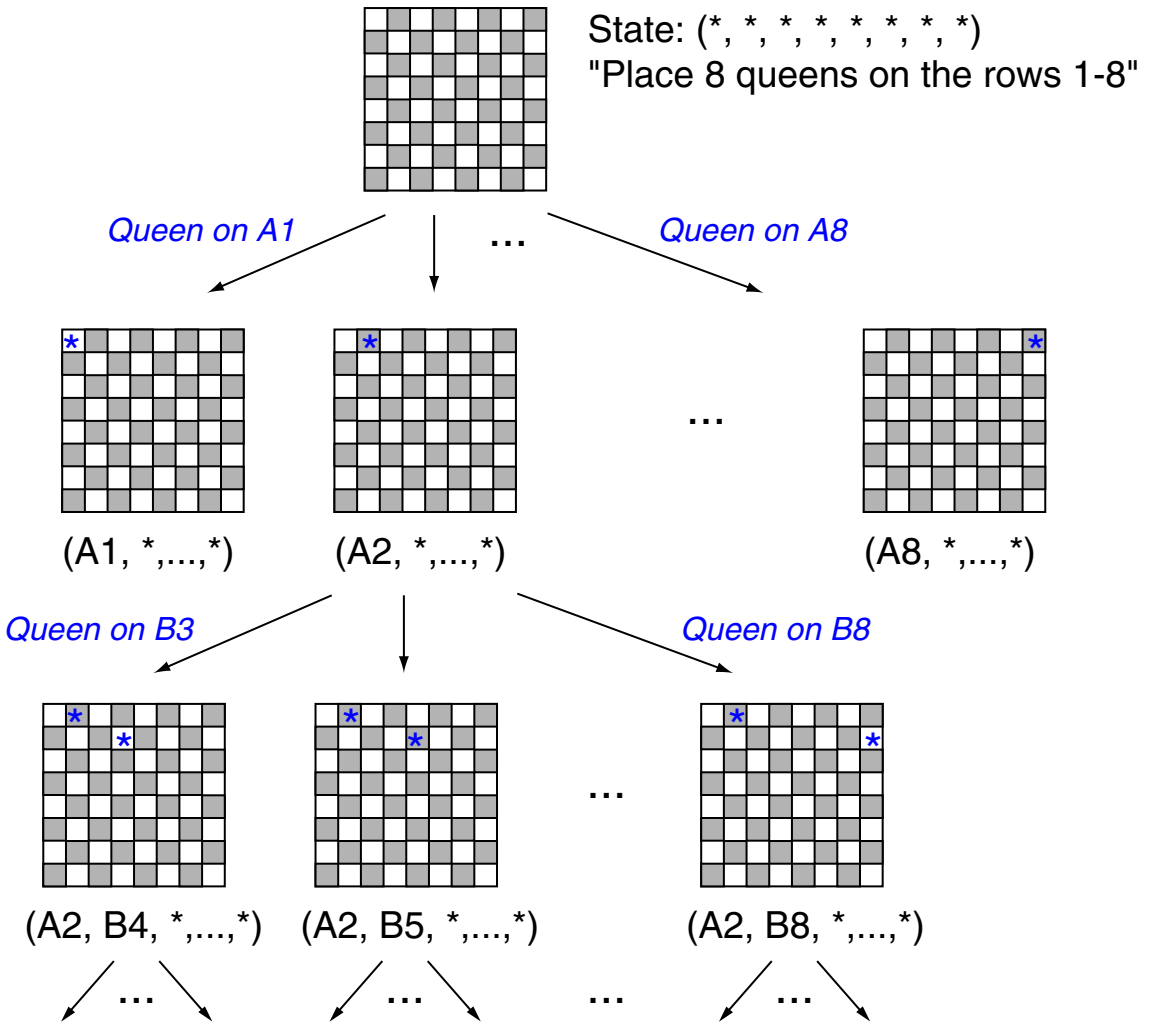
- A solution is a solution candidate that is identified by a sequence of refinement steps in subset splittings, i.e., a path in G from s to some node $\gamma \in \Gamma$.
- A solution base is an initial part of such a path (hopefully).
- Search problem can be solved by searching for specific paths in a state space graph G.
- State space search algorithms maintain a selection of solution bases.

Remarks:

- Since state space graphs are defined via subset splitting, these graphs are directed graphs, but no multigraphs, i.e. for each pair of nodes n, n' there is at most one edge from n to n'.
- The encoding of a solution base $S' \subset S$ of a search space S with solution candidates must be unique.
- The encoding of a solution base must facilitate the computation of control heuristics.
- If P is a solution path for a node n in G and if n' is some node in P, then the subpath P' of P starting in n' and ending in the endnode of P is a solution path for n' in G. This subpath P' is sometimes called the *solution path in P induced by n'*.
- In graph theory, we often distinguish between walks and paths: A path is a walk in which all vertices (except possibly the first and last in order to allow e.g. Hamiltonian cycles) are distinct. Pearl does not make this distinction explicitly, speaking of cyclic paths and acyclic paths. Nevertheless, as best-first algorithms assume cyclic paths to be pruned, solution paths can be assumed to be acyclic for most search problems.
- Given a path in the state-space graph, the identification of cycles requires the identification of those nodes that represent the same state. I.e., we need a function that decides for two encodings whether their associated rest problem is the same. Like the specification of a suited encoding, the specification of such a function belongs to the problem domain.
- If two equal states are not identified as such, the state-space graph degenerates since it contains undetected duplicate nodes and subtrees. This (undetected) redundancy renders an effective pruning impossible.

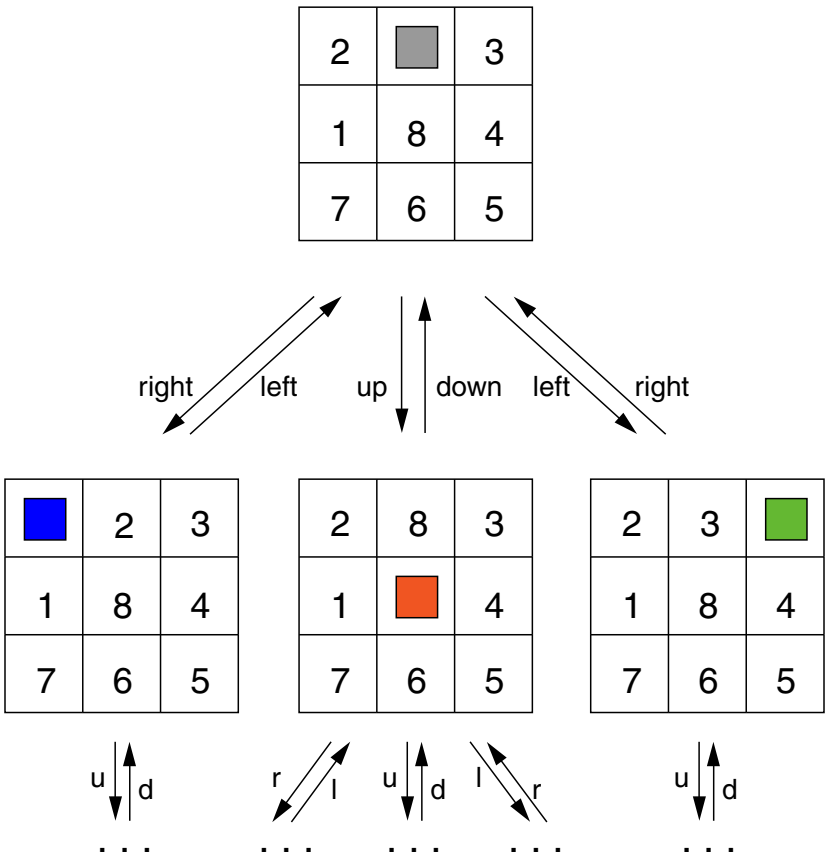
State-Space Representation

8-Queens Problem



State-Space Representation

8-Puzzle Problem



- ❑ Many equal states where the associated solution bases (encodings) are different.
- ➜ Consider only the shortest move sequence. Keyword: *Pruning by Dominance*

Remarks:

- In the 8-queens problem the encoding of solution bases can serve as state description as well. Example: (A2, B4, *, *, *, *, *, *) encodes all board configurations with a queen on A2 and B4. At the same time, we can easily infer the fact that 6 queens still have to placed in the rows 1, 3, 5–8, given the constraints imposed by the queens on A2 and B4.
  parEven more, the formulation of explicit information about the rest problem (“Which cells are attacked or not?”) would be complex and difficult to exploit.
- In the 8-puzzle problem operators can still be applied on a goal node. I.e., goal nodes are not necessarily leaf nodes in a state-space graph.

Search Space Encoding

Applicable Heuristic Problem Solving Methods [Zanakis, 1989]

A. Construction

Construction algorithms generate a solution by adding individual components one at a time until a feasible solution is obtained.

➜ OR Graph Search (= State Space Search)

Systematic control strategy: split and prune.

B. Improvement

Improvement heuristics begin with a feasible solution and successively improve it by a sequence of exchanges or mergers in a local search.

➜ Local search algorithms like Hill-Climbing

E. Partitioning (Decomposition [Pearl])

Partitioning algorithms break or 'partition' a problem into smaller subproblems, each of which is solved independently.

State-Space Representation

Search Problem: Constraint-Satisfaction or Optimization

1. Constraint satisfaction problems.
   The solution candidate has to fulfill constraints and shall be found with minimum search effort.
2. Optimization problems.
   The solution candidate has to fulfill constraints and stands out among all other candidates with respect to a special property.

State-Space Representation

Search Problem: Constraint-Satisfaction or Optimization

1. Constraint satisfaction problems.
   The solution candidate has to fulfill constraints and shall be found with minimum search effort.

2. Optimization problems.
   The solution candidate has to fulfill constraints and stands out among all other candidates with respect to a special property.

   The computation of the optimum is often infeasible.
   ➜ semi-optimization: relaxation of the optimality requirement

   (a) Near optimization.
   A threshold for the maximum (cost) deviation is given.
   ➜ WA*, A^*_{ε}, NRA^*_{ε}

   (b) Approximate optimization.
   Even more relaxed: The deviation threshold (near optimization) needs to be adhered to with a given probability only.
   ➜ R^*_{δ}, $R^*_{\delta,\varepsilon}$

Remarks:

- An important goal of optimality relaxation is scaling: users shall be enabled to control the trade-off between efficiency and the deviation from the optimum solution, ideally with a few hyperparameters.