Configuration variables

Kombinatorická optimalizace (NI-KOP)
Combinatorial Optimization (NIE-KOP)
2024/2025

Dept. of Digital Design
Faculty of Information Technology
CTU in Prague

Configuration

- Recall…
  - Given by the problem
    - no matter what algorithm is used
  - Describes the state of the search
  - Brute force tries all configurations
  - For constructive problems, the configuration typically equals to the output

Configuration variables

- Recall…
  - Given by the problem
    - no matter what algorithm is used
    - *… but for some exceptions*
    - *… and their implementation effectiveness may differ*
  - They encode the configuration
  - Important:
    - finite number
    - finite and discrete domains

    ⇒ It must be possible to evaluate all their values by brute force

Configuration variables

- Recall…
  - For constructive problems, they are typically equal to output variables
  - Every solution must have a corresponding configuration
- The variables must
  - Represent any configuration which is a solution,
  - Together with input variables, they allow to compute the constraints (to verify whether the configuration is a valid solution),
  - Together with input variables, allow to evaluate the optimization criterion for optimization problems.
- The variables should
  - Support any required operation effectively
    - e.g. enumeration for a brute force search
  - Not represent a number of configurations that cannot be solutions

Examples

- Sources:
  - A compendium of NP optimization problems
    - http://www.csc.kth.se/~viggo/wwwcompendium/
  - Wikipedia
    - https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Note that these are optimization problems

- but this doesn't matter – for decision (or enumeration, etc.) problems, the configuration variables will be the same

Maximum Knapsack

- Problem definition
  - Instance:
    - M – knapsack capacity
    - U – a set of items, for each $u \in U$:
      - $w(u)$ – a weight
      - $c(u)$ – a cost
  - Solution:
    - A subset $U' \subseteq U$, so that

$$\sum_{u \in U'} w(u) \leq M$$

  - Optimization criterion (measure)

$$\sum_{u \in U'} c(u)$$ - to be maximized

- Configuration
  - The subset $U' \subseteq U$

    (which items are in the knapsack)

- Configuration vars.
  - Bit vector
  - Set
  - Unordered list
  - …

Maximum Knapsack

- Configuration
  - Subset of U

    (which items are in the knapsack)
- Configuration vars.
  - Bit vector
  - Set
  - Unordered list
  - …

Brute force enumerates all their values

- Can they represent any solution?
- Can we check the constraints?
- Can we evaluate the optimization criterion?
- How can we enumerate them?
- How many possibilities?
- Which configurations are valid solutions?

Maximum Knapsack

- Configuration
  - Subset of U

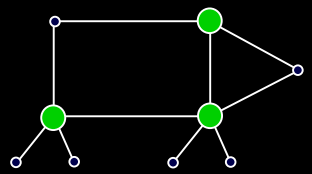
    (which items are in the knapsack)
- Configuration vars.
  - Bit vector
  - Set
  - Unordered list
  - …

Incompetent configuration variables: **the** thing which is in the knapsack

- Can they represent any solution? NO – solutions with more things are missing

Minimum Vertex Cover

- Problem definition
  - Instance:
    - Graph $G = (V, E)$
  - Solution:
    - A vertex cover for G, i.e., a subset $V' \subseteq V$, such that, for each edge $(u, v) \in E$ at least one of u and v belongs to V'
  - Optimization criterion (measure)
    - Cardinality of the vertex cover, i.e., $|V'|$. Should be minimized.



- Configuration
  - The subset $V' \subseteq V$

    (which vertices are in the cover)
- Configuration vars.
  - Bit vector
  - Set
  - Unordered list
  - Subgraph
  - …

Minimum Vertex Cover

- Configuration
  - Subset of V

    (which vertices are in the cover)
- Configuration vars.
  - Bit vector
  - Set
  - Unordered list
  - Subgraph
  - …

Brute force enumerates all their values

- Can they represent any solution?
- Can we check the constraints?
- Can we evaluate the optimization criterion?
- How can we enumerate them?
- How many possibilities?
- Which configurations are valid solutions?

Travelling Salesperson Problem

- Problem definition
  - Instance:
    - Set C of m cities, distances $d(c_i, c_j) \in \mathbb{N}$ for each pair $c_i, c_j \in C$
  - Solution:
    - A tour of C, i.e., a permutation $\pi: [1..m] \rightarrow [1..m]$
  - Optimization criterion (measure)
    - The length of the tour, i.e., the summary distance of the cities in the permutation π. Should be minimized.

- Configuration
  - The permutation π (i.e., the tour)
- Configuration vars.
  - Ordered list of cities
  - A permuted vector
  - ...

Travelling Salesperson Problem

- Configuration
  - The permutation π
    (i.e., the tour)
- Configuration vars.
  - Ordered list of cities
  - Permuted vector
  - …

Brute force enumerates all their values

- Can they represent any solution?
- Can we check the constraints?
- Can we evaluate the optimization criterion?
- How can we enumerate them?
- How many possibilities?
- Which configurations are valid solutions?

Travelling Salesperson Problem

- Configuration
  - The permutation π
    (i.e., the tour)
- Configuration vars.
  - Ordered list of cities
  - Permuted vector
  - …

An ordered list of cities can contain:

- Less cities than m
- More cities than m
- One city twice
- The implementation of configuration variables is correct but inefficient
- A brute force search takes much longer than necessary

Minimum Bin Packing

- Problem definition
  - Instance:
    - Finite set U of items, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$
    - A positive integer bin capacity B
  - Solution:
    - A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$, such that the sum of the sizes of items in each U_i is B or less
  - Optimization criterion (measure)
    - The number of used bins, i.e., the number of disjoint sets, m. Should be minimized.

- Configuration
  - The **partition of** U, i.e the placement of items into the bins

- Configuration vars.
  1. For each item, in which bin it is placed
  2. For each bin, what items it contains

Minimum Bin Packing

Problem definition

- Instance: Finite set U of items, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$, a positive integer bin capacity B
- Solution: A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$, such that the sum of the sizes of items in each U_i is B or less

Configuration vars.

1. For each item, in which bin it is placed

$1 \ldots m$	$1 \ldots m$	$1 \ldots m$	$1 \ldots m$	$1 \ldots m$

$\Rightarrow$ array of $|U|$ integers

How can we enumerate them?
How many possibilities?
Which configurations are valid solutions?

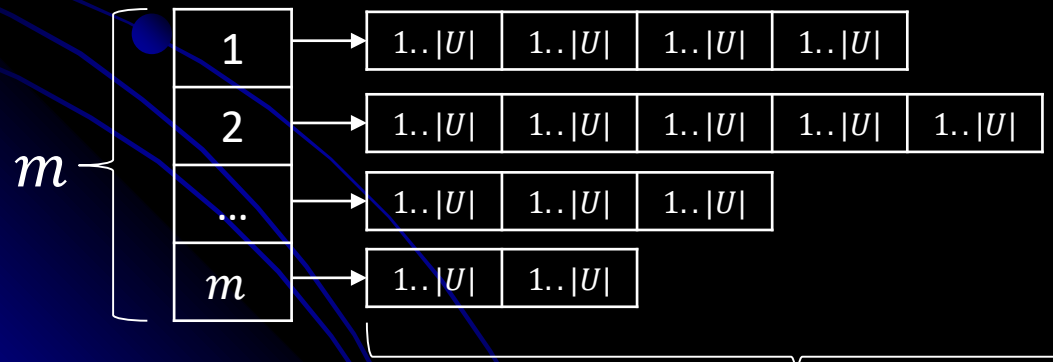
Minimum Bin Packing

Problem definition

- Instance: Finite set U of items, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$, a positive integer bin capacity B
- Solution: A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$, such that the sum of the sizes of items in each U_i is B or less

Configuration vars.

2. For each bin, what items it contains



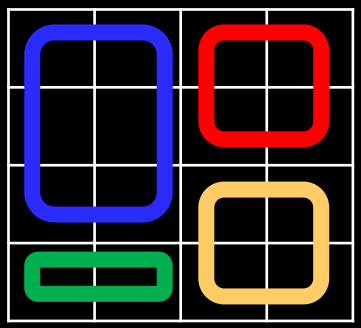
How can we enumerate them? How many possibilities? Which configurations are valid solutions?

$\Rightarrow m$-dimensional array of unordered lists (sets) of integers

Minimum Rectangle Tiling

- Problem definition
  - Instance:
    - An $n \times n$ array A of non-negative numbers, positive integer p.
  - Solution:
    - A partition of A into p non-overlapping rectangular subarrays.
  - Optimization criterion (measure)
    - The maximum weight of any rectangle in the partition. The weight of a rectangle is the sum of its elements. To be minimized.

- Configuration
  - The **partition of** A, i.e. the rectangular subarrays



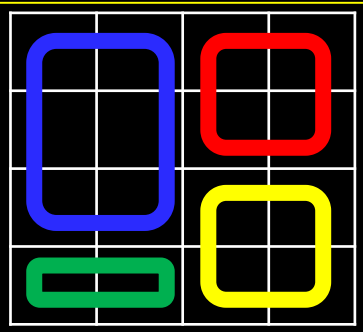
- Configuration vars.

  How to encode the rectangular subarrays?

Minimum Rectangle Tiling

- Problem definition
  - Instance: An $n \times n$ array A of non-negative numbers, positive integer p.
  - Solution: A partition of A into p non-overlapping rectangular subarrays.

- Configuration
  - The rectangular subarrays



- Configuration vars.
  1. Set of rectangles specified by coordinates

$\{((0,0),(1,2)); ((2,0),(3,1)); ((0,3),(1,3)); ((2,2),(3,3))\}$

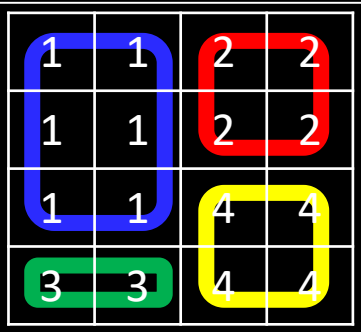
How can we enumerate them? How many possibilities? Which configurations are valid solutions?

Minimum Rectangle Tiling

- Problem definition
  - Instance: An $n \times n$ array A of non-negative numbers, positive integer p.
  - Solution: A partition of A into p non-overlapping rectangular subarrays.

- Configuration
  - The rectangular subarrays
- Configuration vars.
  2. Which cell belongs to which rectangle

  $\Rightarrow n \times n$ array of integers



How can we enumerate them? How many possibilities? Which configurations are valid solutions?

Minimum Consistent Finite Automaton

- Problem definition
  - Instance:
    - Two finite sets P, N of binary strings.
  - Solution:
    - A determinist… $(Q, \{0,1\}, \delta, q$ … strings in P a… in N.
  - Optimization c…
    - The number o… automaton. Sh…

How can we enumerate them? How many possibilities? Which configurations are valid solutions?

- Configuration
  - The automaton
- Configuration vars.

How to encode an automaton?

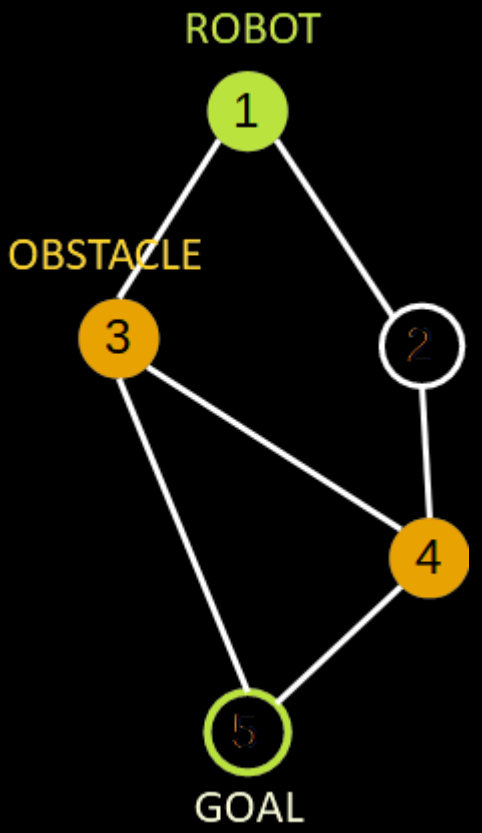
Transition table:

Q_t	$Q_{t+1}/0$	$Q_{t+1}/1$	$Q_t \in F$
q_0	q_1	q_2	y/n
q_1	q_3	q_2	y/n
q_2	…	…	y/n
…	…	…	y/n
q_n	…	…	y/n

Minimum Graph Motion Planning

Problem definition

- Instance:
  - Graph $G = (V, E)$, start vertex $s \in V$, where the robot is initially placed, goal vertex $t \in V$, and a subset $W \subseteq V$ of vertices where the obstacles are initially placed.
- Solution:
  - A motion scheme for the robot and the obstacles. In each time step either the robot or one obstacle may be moved to a neighboring vertex that is not occupied by the robot or an obstacle.
- Optimization criterion (measure)
  - The number of steps until the robot has been moved to the goal vertex t.



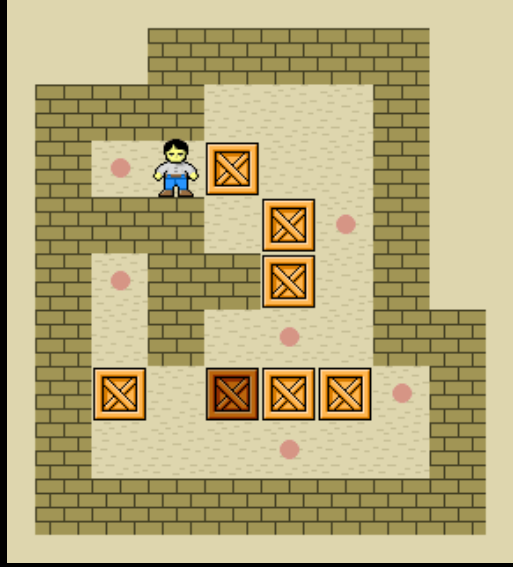
Minimum Graph Motion Planning

- Problem definition
  - Instance:
    - Graph $G = (V, E)$, start vertex $s \in V$, where the robot is initially placed, goal vertex $t \in V$, and a subset $W \subseteq V$ of vertices where the obstacles are initially placed.
  - Solution:
    - A motion scheme for the robot and the obstacles. In each time step either the robot or one obstacle may be moved to a neighboring vertex that is not occupied by the robot or an obstacle.
  - Optimization criterion (measure)
    - The number of steps until the robot has been moved to the goal vertex t.

- Configuration
  - The motion scheme
- Configuration vars.
  - Sequence of moves of the robot/obstacles

How can we enumerate them? How many possibilities? Which configurations are valid solutions?

Sokoban



- What is Sokoban?
  - A game from 1980
  - But also an important combinatorial problem
    - It is proven to be PSPACE-complete
- Problem definition (informal)
  - Instance:
    - A maze (warehouse) with boxes and a warehouseman
      - The warehouseman can
        - move to unoccupied places in the maze
        - push the boxes to unoccupied places in the maze
    - Target places of boxes
  - Solution
    - A motion scheme for the warehouseman
  - Optimization criterion (measure)
    - The number of warehouseman steps

Sokoban

- What is Sokoban?
  - A game from 1980
  - But also an important combinatorial problem
    - It is proven to be PSPACE-complete
- Problem definition (informal)
  - Instance:
    - A maze (warehouse) with boxes and a warehouseman
      - The warehouseman can
        - move to unoccupied places in the maze
        - push the boxes to unoccupied places in the maze
    - Target places of boxes
  - Solution
    - A motion scheme for the warehouseman
  - Optimization criterion (measure)
    - The number of warehouseman steps

- What does it mean?
  - One of the hardest problems solvable in polynomial space
  - $\Downarrow$
  - SAT $\leq_P$ Sokoban
  - $QBF_k \leq_P$ Sokoban
  - ~~Sokoban $\leq_P$ SAT~~

Sokoban

- Problem definition (informal)
  - Instance:
    - A maze (warehouse) with boxes and a warehouseman
      - The warehouseman can
        - move to unoccupied places in the maze
        - push the boxes to unoccupied places in the maze
    - Target places of boxes
  - Solution
    - A motion scheme for the warehouseman
  - Optimization criterion (measure)
    - The number of warehouseman steps

- Configuration
  - A motion scheme for the warehouseman
- Configuration vars.
  - Sequence of moves of the warehouseman
  - Or sequence of valid moves of boxes
    - It is “shorter”
    - The movement of the warehouseman can be computed (shortest path in a graph)

Minimum Equivalent Boolean Formula

Problem definition

- Instance:
  - Boolean formula consisting of literals, operators, parentheses. E.g.,
    $F = \left(\left(a + \bar{b}\right) \cdot c + b \cdot (a + \bar{c})\right) \cdot \bar{b}$
- Solution:
  - Equivalent Boolean formula consisting of literals, operators, parentheses. E.g.,
    $F = \bar{b}c$
- Optimization criterion (measure)
  - The number of literals. Should be minimized.
- Note
  - This problem is more complex than NP. Why?

Configuration

- The formula

Configuration vars.

How to encode a formula?

- String

How can we enumerate them? How many possibilities? Which configurations are valid solutions?

Rough problems classification

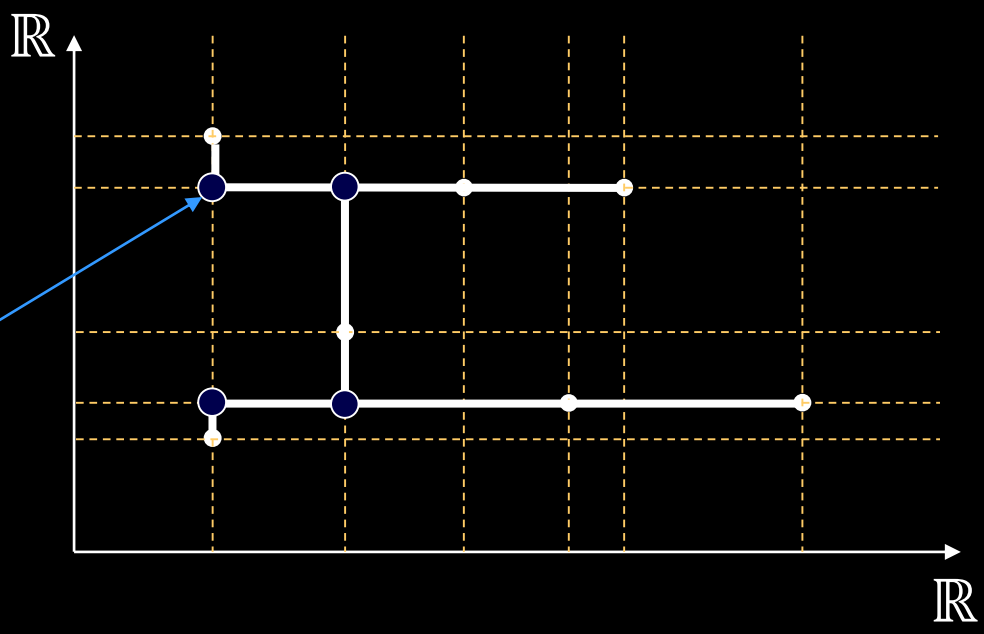
- What have we found?
  - Optimum subset problems
    - Knapsack
    - Vertex cover
    - SAT (0/1 evaluation)
  - Optimum permutation
    - TSP
  - Optimum assignment
    - Minimum bin packing
  - Optimum sequence
    - Minimum graph motion planning
  - Other
    - Minimum consistent automaton
    - …

Minimum Steiner tree in Manhattan metric

Problem definition

- Instance:
  - A set $P \subseteq \mathbb{R} \times \mathbb{R}$ points in a plane
- Solution:
  - A finite set of Steiner points $Q \subseteq \mathbb{R} \times \mathbb{R}$
- Optimization criterion
  - The total weight of the minimum spanning tree for the vertex set $P \cup Q$, where the weight of an edge $\langle (x_1, y_1), (x_2, y_2) \rangle$ is the length in Manhattan metric, i.e.,

$$|x_1 - x_2| + |y_1 - y_2|$$



⇒ Discrete coordinates, even though in real numbers

⇒ Discretization

⇒ Still a combinatorial problem

The Buckets problem

- Problem definition
  - There are n buckets, a tap, and a sink. Given:
    - Capacities of the buckets
    - Initial amounts of water in each bucket
    - Target amounts of water in each bucket
  - In each step you can:
    - Fill any bucket up to the brim (to its maximal capacity)
    - Empty any bucket
    - Pour the water from one bucket to another, either emptying one or fully filling the second bucket
  - The goal is to start from the initial state and reach the target amounts of water in the respective buckets
  - Optimization criterion
    - The number of operations (pouring) with buckets

The Buckets problem

- Discussion
  - What is the instance?

    - n buckets
    - Capacities of the buckets
    - Initial amounts of water in each bucket
    - Target amounts of water in each bucket
  - What is the configuration?

    - Sequence of operations (pouring)

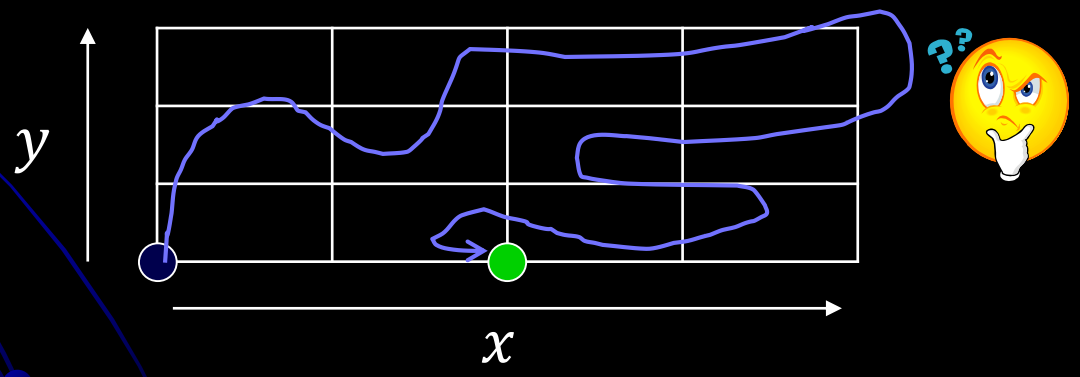
The Buckets problem

- Given example instance
  - Buckets: 2
  - Bucket capacities: (4, 3)
  - Initial amounts: (0, 0)
  - Target amounts: (2, 0)
- Whiteboard exercise

Current amounts of water in buckets: (x, y)

$x \in \{0, 1, 2, 3, 4\}$

$y \in \{0, 1, 2, 3\}$



The Buckets problem

- The bucket space

  Current amounts of water in buckets: (x, y)

  $x \in \{0, 1, 2, 3, 4\}$

  $y \in \{0, 1, 2, 3\}$

- Discrete amounts
- Finite number of them
  - How many?

⇒ Discretization

⇒ Still a combinatorial problem