DISCRETE OPTIMIZATION

Optimization problems
Heuristic algorithms
Examples of practical problems

Jakub Wróblewski, jakubw@qed.pl

OPTIMIZATION TASKS

Many problems solvable by computers have the form of **optimization tasks**:
„Given the space of all possible solutions, find the most appropriate one”

Problems:

- How to specify an „optimization task”?
- How to specify degree of complexity of a method (algorithm) for searching for the optimal solution?
- Which tasks should be regarded as „easy” and which – as „hard” (solvable only approximately)?
- How to solve „hard” problems in approximate (but satisfactory) way?

FORMAL DEFINITION

Discrete optimization task

Let X be an arbitrary finite set (*the space of states*)

Let f : X⟶R be an arbitrary real function over X (*objective function*)

Optimization task is related to finding such element x_0 of X that:

$$f(x_0) = \max(f(x)), \quad x \in X$$

or

$$f(x_0) = \min(f(x)), \quad x \in X$$

EXAMPLES (1)

Sort *n* names according to alphabetical order

The space of states: all possible orderings of *n* names
Cardinality of the space of states: *n!* (**not *n***).

Objective function: e.g. number of pairs of properly ordered adjacent names (maximum: *n-1*, what corresponds to the totally proper ordering).

One can solve each optimization task by examining all possibilities (all elements of the space of states). Often, however, there are far more efficient algorithms (like, e.g., in case of sorting)

EXAMPLES (2)
- THE TRAVELING SALESMAN PROBLEM

Given graph G, with labeled edges of a fixed length, find the shortest route coming through each of nodes exactly once (if it exists)

The space of states: all routes coming through each of nodes exactly once. Cardinality of the space of states is at most $n!$, where n denotes the number of nodes in G

Objective function: aggregated length of a route

EXAMPLES (3)
- THE MATRIX COVERING PROBLEM

Given matrix $A=\{a_{ik}\}$ with values in {0,1}, find the smallest subset of columns B, such that for each i-th row of A exists column $k(i)$ belonging to B, satisfying equality $a_{ik(i)}=1$

The space of states: all possible coverings of matrix A. Cardinality of the space of states: less than 2^n, where n denotes the number of columns

Objective functions: Cardinality of B

EASY AND HARD TASKS

"Easy"

- Sorting
- Solving polynomial equations
- Searching for maxima of continuous and derivable functions
- Multiplying matrices
- Checking existence of the Euler cycle in graphs
- ...

We know effective algorithms providing accurate solutions

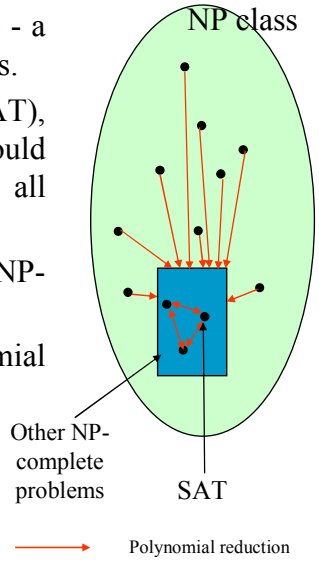
"Hard"

- Searching for maxima of functions not continuous, not derivable functions, with noisy values, changing in time
- Searching for the shortest notation for logical formulas
- Decomposing numbers onto their prime coefficients
- Checking existence of the Hamilton cycle in graphs

Known algorithms have high (e.g. exponential) time complexity
We must search for approximate methods

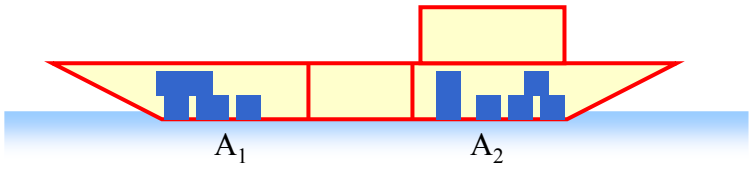
P & NP

- P - class of polynomial (*easy*) problems.
- NP (*nondeterministically polynomial*) - a wide class of hard (practical) problems.
- There exists "universal" problem (SAT), such that its polynomial solution would provide polynomial solutions of all problems in NP.
- Many practical problems are NP-complete (*universal for NP class*).
- However, we don't know a polynomial algorithm for any of them.
- Open question: P = NP ?



Set partition

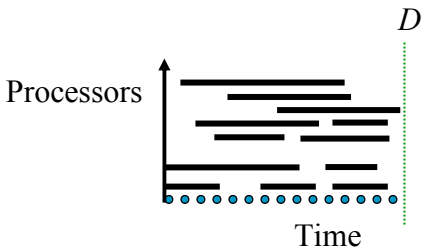
Let n real numbers $\{a_1, \ldots, a_n\}$ be given.
Is it possible to divide them onto two disjoint subsets A_1 and A_2 such that the sum of elements of A_1 is equal to the sum of elements of A_2?



NP-complete problem

Task scheduling

Let the set of tasks of a specified duration be given.
Assume that we have *m* processors and the time limit D.
Is it possible to divide the tasks among processors in such a way that all the tasks will be finished before *D*?



NP-complete problem

Matrix covering

- Let 0-1 matrix $A=\{a_{ij}\}$ of rank $n*n$ be given. Find the smallest subset of columns B such that in each row we have at least one „1” over a column belonging to B
  - Let the list of books available in each of some set of libraries be given. Find minimal subset of libraries offerring together all the books available
  - Consider information about the need for particular goods in particular regions, as well as the list of suppliers of some groups of those goods. Sign contracts with minimal number of suppliers providing all necessary goods in all regions

GREEDY ALGORITHM (1)

General principle:
build a solution “step by step”; at each level of the construction choose a continuation, which locally provides the largest increase of quality

GREEDY ALGORITHM (2)

Example: scheduling at school:
Let n lectures, with specified time intervals (in hours), are given. Find such the largest possible subset of lectures which do not overlap in time

Algorithm: First we choose the lecture which ends the most early. In the foregoing steps, we choose lectures which do not overlap with the previous ones and end the most early

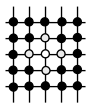
Such a greedy algorithm always provides optimal solution

NEIGHBORHOOD (1)

We assume that it's possible to define over set X the notion of "neighborhood": if $x \in X$, then N(x) denotes the (finite) set of its neighbors

Such a definition provides us with unbounded possibilities of specifying „neighbors". In practice, that notion should somehow correspond to a particular task (neighbors should correspond to similar solutions)

Example: X – a discrete plane. By "neighbors" we regard points which are sufficiently close to each other

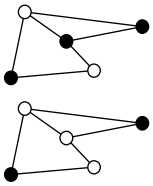


NEIGHBORHOOD (2)

Example: The Vertex Covering Problem

Given graph G, find the smallest subset of vertices B, such that each edge is adjusent to at least one element of B

The space of states X: the set of all potentialy possible coverings (actually, the set of all subsets of vertices)



We regard two subsets as neighbors, if and only if they differ with respect to only one element

HEURISTICS: TWO GENERAL SCHEMES

Greedy principle:
build a solution “step by step”; at each level of the construction choose a continuation, which locally provides the largest increase of quality

Search by neighborhood:
define a notion of neighborhood in state space, perform a local search on complete solutions
(neighbors should correspond to similar solutions)

THE HILL CLIMBING ALGORITHM

General scheme: we begin with a random point, search through all its neighbors, and choose the one with the largest quality value ("we climb up"). We repeat this step until reaching (local) maximum

```
x0 = Random(X)
do
  max=x0
  for(x∈N(x0))
    if(f(x)>f(max)) max=x
  end for
  if(max=x0) break
  x0=max
while(1)
```

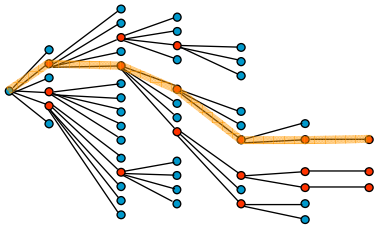
- Advantages: simple implementation, fast search
- Disadvantages: the lack of robustness with respect to local maxima, strong dependency on the starting point

Neighborhood-based method

BEAM SEARCH (1)

General scheme: we construct solution step by step (like in greedy approach), however, we always keep in memory k currently best partial solutions and continue the construction process for each of them

BEAM SEARCH



1. Begin with a randomly chosen element of solution.

2. Check all possible continuations and select k „most promising” ones.

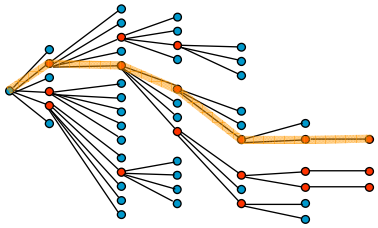
3. Check all possible cotinuations of the selected partial solutions, at each level restrict to k „most promising” ones.

4. Continue step 3 until complete solutions are created. Choose the best of them.

Greedy-like method

BEAM SEARCH (2)

Example: The TSP for 7 towns (*k*=*3*)



1. Begin with a randomly chosen town

2. Find *k* nearest towns

3. Beginning with these towns, calculate distances to the towns not visited yet. Then choose *k* towns, which correspond to the shortest aggregate route so far

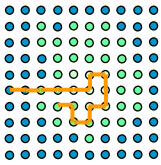
4. Repeat point 3 keeping in memory *k* currently shortest partial routes

5. After reaching the last town, choose the best of *k* solutions

TABU SEARCH

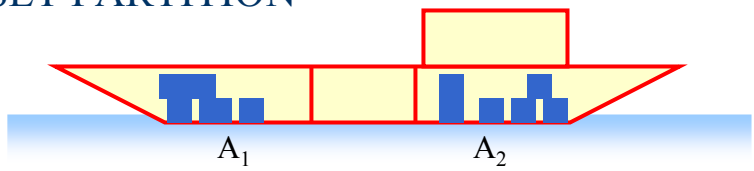
*General scheme: Look through the space of states, by continuing always with the neighbor providing the best quality (just like in the hill-climbing algorithm), unless it occurs at the list of **forbidden states**. At such a list, store k recently visited states (for, e.g., k=1000).*

Example: maximization of two-dimensional function (over a grid).



Neighborhood-based method

SET PARTITION



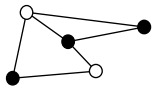
Let n real numbers $\{a_1, \ldots, a_n\}$ be given.
Find two disjoint subsets A_1 and A_2 such that the sum of elements of A_1 is close to the sum of elements of A_2.

Greedy scheme:
one step = add one element or a pair of them
optimization by the current difference of sums

Neighborhood scheme:
define neighbors as two partitions differing by only one element or a pair of them

VERTEX COVERING

Given a graph G = (V, E). Find a minimal set of vertices such that every edge is adjacent to at least one of them.



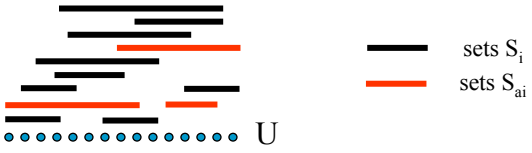
Greedy scheme:
one step = one unused vertex
optimization by the number of newly covered edges

Neighborhood scheme:
define neighbors as two sets differing by only one element

SET COVERING

Given a set U and a family $\{S_1, \ldots, S_n\}$ of its subsets (sum = U).Find possibly small $\{S_{a1}, \ldots, S_{ak}\}$ such that:

$$S_{a1} \vee \ldots \vee S_{ak} = S_1 \vee \ldots \vee S_n = U$$



Greedy scheme:
one step = one unused subset
optimization: no. of newly covered elements

Neighborhood scheme:
define neighbors as two families differing by only one subset

MATRIX COVERING

Given matrix A=$\{a_{ik}\}$ with values in {0,1}, find the smallest subset of columns B, such that for each *i*-th row of A exists column *k(i)* belonging to B, satisfying equality $a_{ik(i)}=1$

Greedy scheme:
one step = one unused column
optimization: no. of newly covered rows

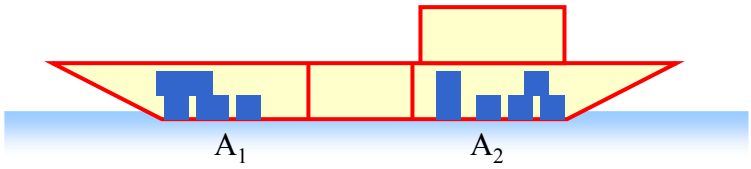
Neighborhood scheme:
define neighbors as two potential coverings differing by only one column

REMARKS ON NEIGHBORHOOD

In general, we can use any definition of neighborhood. However, it is reasonable to assure some properties:

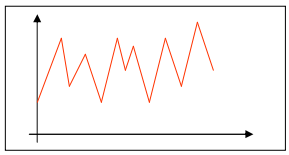
- **There should not be too many neighbors.**
  A loop checking all neighbors of a solution is common part of many neighborhood-based methods. The more neighbors we declare, the slower search is.
- **Relation of neighboorhood should be connected.**
  We should be able to reach any solution by finite steps, starting from any other solution.
- **Neighbors should typically have similar value of goal function.**
  I.e. we assume some kind of continuity. Most of heuristics base on a hidden assumption, that „similar causes yield similar effects".

NEIGHBORHOOD – AN EXAMPLE



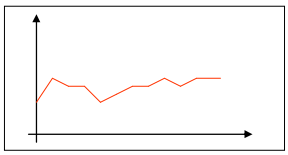
Set partition problem.

Goal function: absolute difference between sums of subsets.



Neighbors:
Move one element.

Hard to optimize.



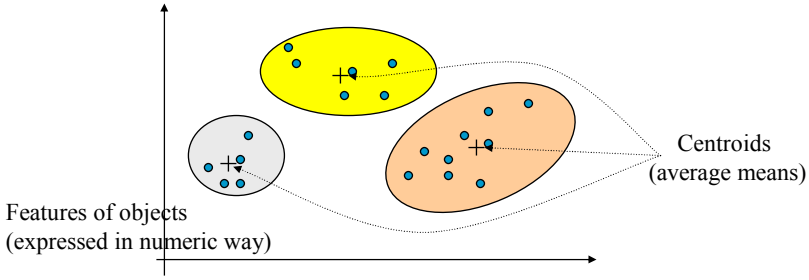
Neighbors:
Switch a pair of elements.

Some solutions unreachable.
More neighbors.

Better solution: try to exploit both methods.

Other optimization problems – grouping

Algorithms gathering objects within larger groups at the basis of their mutual similarity (unsupervised learning)

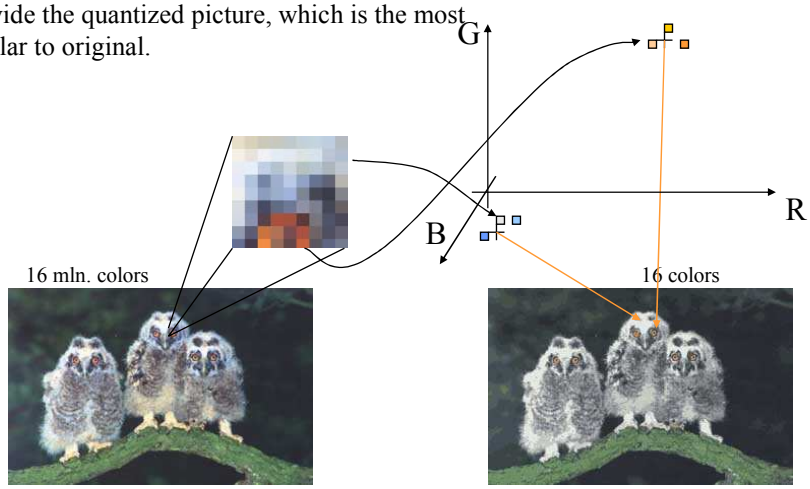


Criterion of „similarity" of objects is based on mutual distance.

Optimization task: Find such partition that distances between objects within the same groups are minimal and distances between objects from different groups are maximal

Gropuing algorithms – example of application

Quantization of colors: find such 16 colors that provide the quantized picture, which is the most similar to original.



Usually applied algorithms: close to greedy approach (e.g. k-means) or hill-climbing (e.g. centroids).

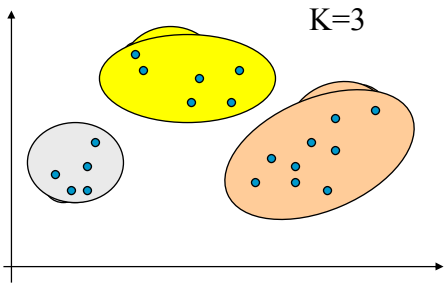
GROUPING - K-MEANS (EXAMPLE OF ALGORITHM)

Assumption: we are supposed to divide the set of objects onto K disjoint groups

1. Find K points which are the most distant to each other and set up K groups

2. Find object which is the closest to a given group and add it there (**greedy strategy**)

3. Repeat step 2 until considering all objects



GROUPING - CENTROIDS (EXAMPLE OF ALGORITHM)

K=2

1. Divide the set onto K groups randomly
2. Calculate the middle (centroid) of each group
3. Divide objects again by assigning them to the group with the closest middle point
4. Repeat steps 2,3 until stabilization