Evolutionary Computation

2024/25

Master Artificial Intelligence

Arno Formella

Departamento de Informática
Escola Superior de Enxeñaría Informática
Universidade de Vigo

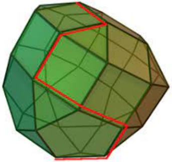
24/25

Local search methods

- Local search methods explore the search space by inspecting (close or far) neighbor solutions.
- They stop at a local minimum, i.e., all neighbors are greater (remind: we are searching for a minimum).

Local search methods

- a classical example is the simplex method for linear programming



- or the Newton method (or Newton-Raphson method) applied to optimization (here formulated as maximization)

```
while GradientFobj(xi) > tolerance:
  xi=xi-GradientFobj(xi)/SecondDerivativeFobj(xi)
```

(Take care: should check if eventually really maximum, and not minimum.)

Local search methods

Further classic iterative optimization methods for the minimization of real-valued functions that need gradient information, are:

- Gauss-Newton method
  (variation of the Newton-method by using the Jacobean-matrix instead of the Hessian-matrix)
  second derivatives are not required here
- gradient descent (or, similarly, steepest decent)
- Levenberg-Marquardt methods
  (interpolation between Gauss-Newton methods and gradiant descent)
- Nesterov's method for convex optimization
  (with much faster convergence rate compared to gradient decent)

Local search methods

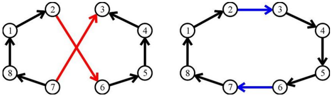
Further classic iterative optimization methods for minimization of real-valued functions that don’t need gradient information, are:

- Nelder-Mead method (heuristic), converges to a stationary point (minimum, maximum, or saddle, i.e., gradient is zero)
- idea: shrink, reflect, and expand a simplex (triangle in 2D), by evaluating the objective function on corners and faces (edges in 2D)
- García-Palomares method, converges to a local minimum
- idea: explore the neighorhood according a random local spanning coordinate system and proceed at a point that has been found with a sufficiently steep descent (otherwise iterate with smaller tolerance)

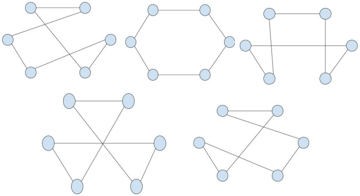
Local search for traveling salesperson problem

We have seen already the idea: define a local operation that changes a given tour into another tour

- 2-opt move:



- 3-opt move:



- Lin-Kernighan-heuristics (and efficient LKH by Helsgaun) is a combination of 2-opt, 3-opt, and rare k-opt moves (recall, still state-of-the-art to solve TSP)

Use of local search

Observe: local search methods can be used in any other optimization algorithm in order to (try to) converge to a local minimum.

That is exactly what LKH (Lin-Kernighan-Helsgaun, the very good implementation) does. It starts with a tour based on a minimal spanning tree. However:

- Can all tours be reached with 2-opt moves? (when starting with a certain initial tour) ...*still an open question*
- There are worst case scenarios where the 2-opt heuristics has exponential runtime until convergence.
- What about 3-opt, or k-opt, moves? ...*still an open question*

Reactive Tabu Search

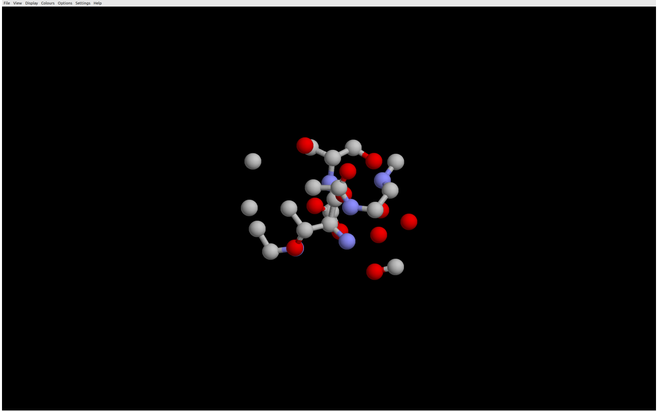
- start with a feasible solution (e.g., with some heuristics)
- search for possibilities to improve the current solution (e.g., search in the neighborhood)
- if we can improve: choose the best one or a random one
- if we cannot improve (i.e., trapped at a minimum):
  - search for possibilities to worsen the current solution
  - if we can escape: try again improvements
  - if we cannot escape: jump to another feasible solution

The Tabu criterion

- avoid repetitive movements taking advantage of a memory that stores forbidden intermediate solutions
(or forbidden specific features of the current neighborhood search)
- i.e., mark certain movements as tabu for a certain number of iterations, i.e., the memory is volatile!
- reactive means that the tabu period is dynamically adapted

psm: point set match for proteins

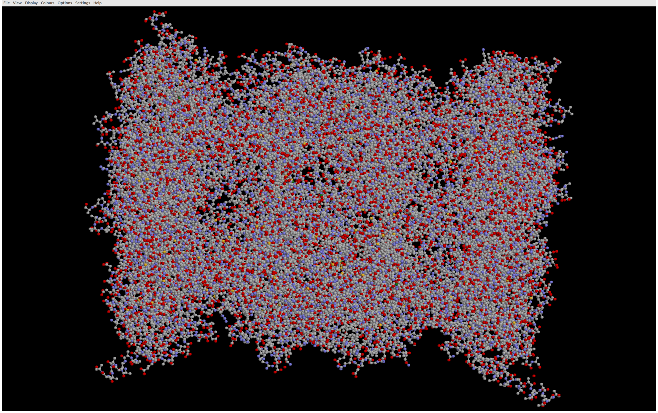
A template and graph based method with local search and use of domain specific knowledge for approximate match.



searching a 3D-structure (34 atoms) in a protein with certain admitted tolerance

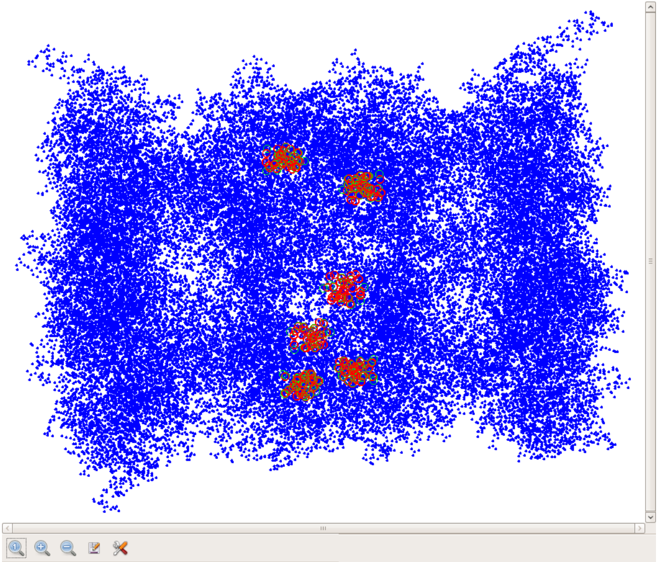
psm: point set match for proteins

A template and graph based method with local search and use of domain specific knowledge for approximate match.



the protein has 50000 atoms

psm found, for instance, 6 locations:



things to take into account

the search space and/or the objective function can be:

discrete	or	continous
total	or	partial
simple	or	complex
explicite	or	implicite
modelado	or	experimental
linear	or	non-linear
convex	or	non-convex
differentiable	or	non−differentiable
single-objective	or	multi-objective
constrained	or	unconstrained
static	or	dynamic

We have seen already a lot of examples of all kind.

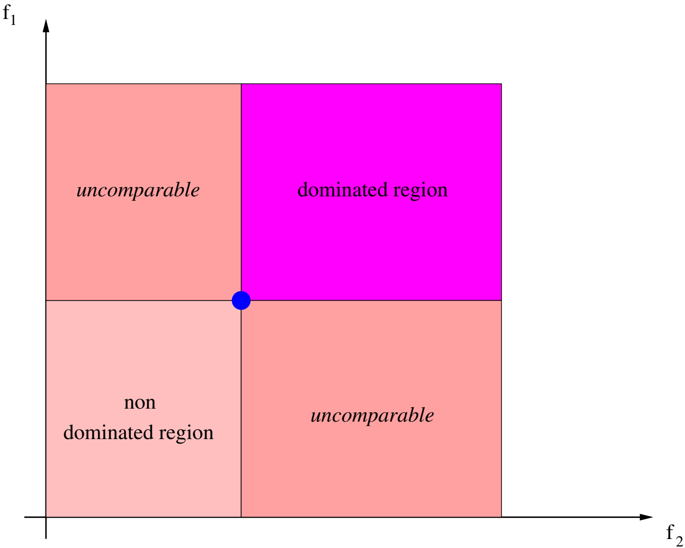
multi-objective optimization

- given a search space $\mathbb{X}$ (called as well *search domain* or *problem space*) and
- a set of k functions f_i (bounded from below) from the search space to the real numbers (or at least a totally ordered set) e.g. $f_i : \mathbb{X} \longrightarrow \mathbb{R}$,
- find an element $x^\star \in \mathbb{X}$ such that $f_i(x^\star) \leq f_i(x)$ for all $x \in \mathbb{X}$ and all k functions f_i
- maybe there is no point in $\mathbb{X}$ that minimizes all the k functions simultaneously, then we look for Pareto-optimal solutions, i.e., solutions that cannot be improved without worsening at least one of the other objective functions

Pareto front

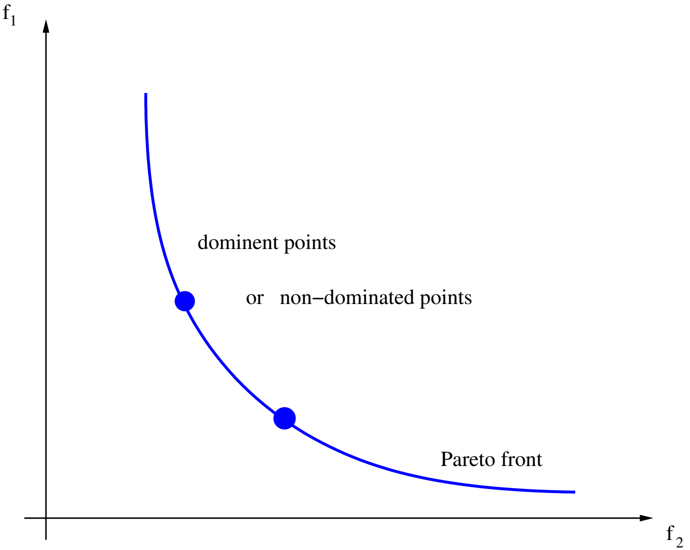
- remind we have more than one independent objective function
- Pareto optimal (global): every other component for all other solutions is worse (or equal)
  (other names are: efficient points or non-interior points)
- Pareto optimal (local): every other component for all other solutions in a local neighborhood is worse (or equal)
- hence, the Pareto front describes the trade-off between the different objectives
- the Pareto front consist of the points in the search space that are not dominated by any other point

Dominant points and regions



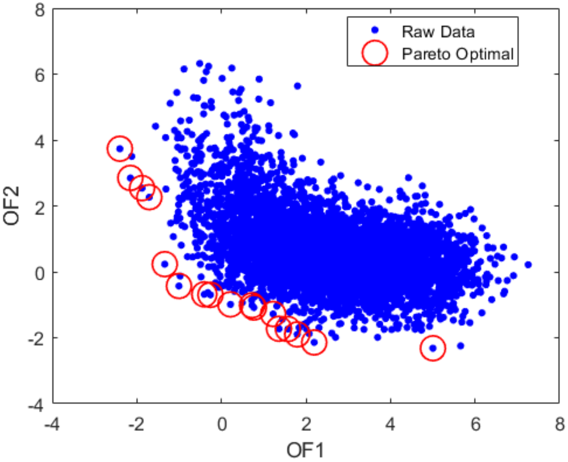
Pareto front

Example of a Pareto front with 2 marked non-dominated points:



Pareto front

Example of a Pareto front for 2 objectives (on measured data):



Multi–objective optimization

How can we find a somewhat good solution to the optimization problem with more than one objective function?

- convex combination of the objectives
- homotopic techniques, i.e., compute the entire Pareto front, for instance with a population based algorithm, and select later...
  (to obtain the Pareto front one might explore the coefficient space of the convex combination)
- goal programming, i.e., fixed values for all objectives and minimize the distance of all objectives to the predefined goals (according to some convenient distance metric)

Multi−objective optimization (continued)

- priority optimization, i.e., fix thresholds for all but one objective function beforehand and optimize above the threshold according to the most important one
- priorization (multi-level) programming, i.e., optimize according to a predefined ordering of the objective functions.
- fixed trade-off, i.e., find the point in the Pareto front that is tangent to a certain hyperplane (especially usefull when Pareto front is convex and low dimensional).

Optimization with restrictions (or constraints)

In many application there are restrictions (or constraints) that limit the optimization process:

- find an element $x^\star \in \mathbb{X}$ such that $f_i(x^\star) \leq f_i(x)$ for all $x \in \mathbb{X}$ and all k functions f_i and
- a certain number L of inequality constraints $g_j(x) \geq 0$ (for all $j \in [1:L]$) are fulfilled, and
- a certain number E of equality constraints $h_n(x) = 0$ (for all $n \in [1:N]$) are fulfilled.

Simple constraints are for instance so-called box-constraints, i.e., the search space is confined in each dimension by an interval.

Such box constraints are often handled separately in the optimization packages.

Optimization with restrictions (or constraints)

- The inequality and equality constraints again might be linear or non-linear functions.
- Due to the restrictions there might arise points (elements of the search space) during the optimization process which are unfeasible, i.e., no valid objective function values can be computed (or even the objective function cannot be computed at all).
- Sometimes even trying to find some feasible solution is already a very complex task (for instance: for TSP with time windows the problem is already NP-hard).

Optimization with restrictions (or constraints)

To tackle constraints there are two main classical approaches:

- use of penalties, i.e., assign *sufficiently large* value(s) to the objective function(s)
- use of interior methods, i.e., make sure not to leave the feasible region
  (main idea: use the fulfillment of the constraints as additional objective function building a so-called *barrier function* with an additional parameter μ that is continuously shrinked to reach zero)

Multi−objective optimization with evolutionary methods

- Evolutionary methods can approximate the Pareto front in parallel (with the help of the diversity among the individuals).
- For instance particle swarm systems varying the weights of a convex combination periodically during the iterations.
- For instance a genetic algorithm can hold a population that tries to converge (in some sense uniformly) towards the Pareto front.