Evolutionary Computation

2023/24

Master Artificial Intelligence

Arno Formella

Departamento de Informática
Escola Superior de Enxeñaría Informática
Universidade de Vigo

23/24

PSO: different versions

binary version: the variables are interpreted as binary values according to a distribution or threshold

discret version: the variables are interpreted as integer values (for instance with simple rounding)

dynamic version: the search space is reinitialized and/or the local variables are reset (type of outer Monte Carlo loop)

PSO: convergence

- the individuals should exhibit certain diversity
  (recall the similarity measures)
- diversity can be forced dynamically by adapting the parameters alongside the simulation time
- or one might use the lack of diversity as a stopping condition

ant colony optimization (ACO)

The idea stems from stigmergy: exercise indirect communication and coordination through the environment
(leave a trace and act on findings).

- The inspiration stems from ants, bees, termites, wasps, etc.
- The individuals of a population leave information (pheromones) in the search space.
- The decisions are based on individual information or behavior and on the pheromones encountered.
- The information (pheromones) is volatile and can evaporate.
- The pheromones or a statistical evaluation of the individuals define the solution.
- Initially invented to deal with combinatorial problems (like TSP).

ACO: principal loop

An ant colony optimization can be summarized in the following principal loop:

```
InitializePheromoneValues()
while not Stopping():
  for individuals in range(n):
    ConstructSolution(individual)
    UpdatePhermoneValues()
    UpdateIndividuals()
```

ACO: how TSP can be approached

- The ant colony optimization takes place on the graph of the underlying problem (e.g., the complete graph among all cities).
- The ants are placed at the cities.
- The initial pheromones are placed on the edges (either constant value or inversely proportional to the distance).
- The ants (in an appropriate iteration) run along a path in the graph (excluding already visited cities) and draw at each city a decision in which direction to continue.
- The decision is based on: pheromones on each possible edge, maybe on some own information stored at the individual, and on a random value.
- Once the tour is completed for all ants, all of them deposit their pheromone on their tracks.
- The general evaporation process is applied to all/changed edges.
- The currently best tour is memorized.
- The iteration is repeated until a certain stopping condition is met.

ACO: when to use?

ACO approaches are especially possible when the underlying problem allows for a constructive solution (as seen with the nearest-neighbor heuristic for the TSP).

Simon gives the example that an ACO approach found a tour with 3% deficit on the Berlin52 problem.

The no-free-lunch theorem

The no-free-lunch theorem states that the performance of all optimization (search) algorithms, amortized over the set of all possible functions, is equivalent. The implications of this theorem are far reaching, since it implies that no general algorithm can be designed so that it will be superior to a linear enumeration of the search space (exhaustive search).

What are practical implications of the no-free-lunch theorem?

- Each problem (or each type/class of problem) might need its own and proper optimization method.
- Maybe for interesting problems we find good optimization algorithms (we are not interested in *all* problems).
- Benchmarking optimization algorithms is a challenge, as general benchmarks might just provide *average* data, but our algorithm might be special for a niche of problems.
- There is a need to categorize problems and algorithms to obtain some insight on which type of problem a certain type of algorithm performs well.

How to compare different approaches?

In order to compare different algorithms one might take into account:

- wall clock runtime on comparable systems
- (average) number of objective functions evaluations
  (but the rest of the inverted time must not be neglected)
  difficult to be used when comparing constructing algorithms
- the result as distance to optimium or to some known lower bound
- mean best fitness
- properties of the solution histogram (fitness of all solutions found)
- scaling properties with problem size (applied to any measure above)

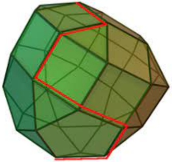
Practical aspects to be considered

One has to decide what is really needed:

- need a good (or best) solution independent of runtime
  (e.g. controler for space telescope or the evolved antenna)
- need a moderate solution fast
  (e.g., daily TSP with time windows, where finding a feasible solution is already NP-hard)

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 searching for a minimum)
- 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 steepest decent)
- Levenberg-Marquardt methods (interpolation between Gauss-Newton methods and gradiant descent)

Local search methods

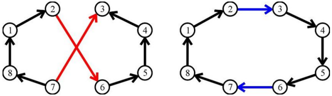
Further classic iterative optimization methods for minimization of real-valued functions that don’t need gradient information, are:

- 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)

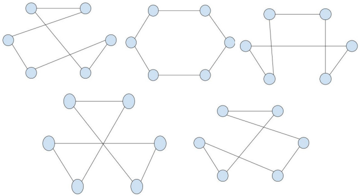
Local search for traveling salesperson problem

Idea: define a local operation that changes a given tour into another tour:

- 2-opt move:



- 3-opt move:



- *k*-opt move
- Lin-Kernighan-heuristics (LKH) 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. 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

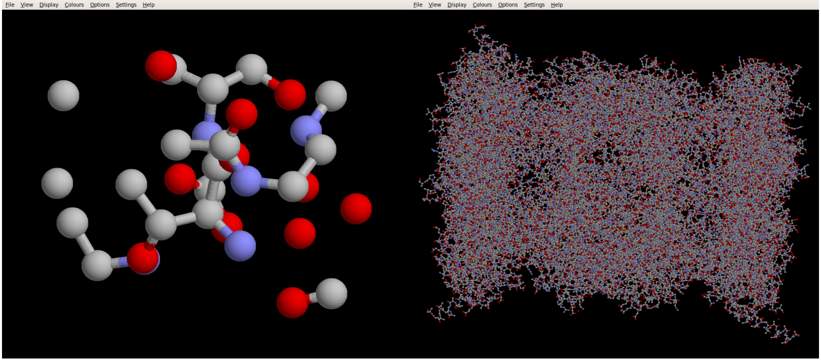
- start with a feasible solution (e.g., with some heuristics)
- search for possibilities to improve the current solution (e.g., search in the neighborhood)
- if we can improve: choose one, the best or a random one.
- if we cannot improve (i.e., trapped at a minimum):
  - search for possibilities to worsen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

The Tabu criterion

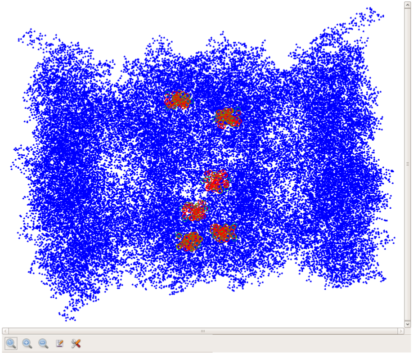
- avoid repetitive movements taking advantage of a memory that stores forbidden intermediate solutions
  (or forbidden specific features of the current neighborhood search)
- i.e., mark certain movements as tabu for a certain number of iterations, i.e.,the memory is volatile!
- reactive means that the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

psm: point set match for proteins

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



searching a 3D-structure (34 atoms) in a protein (50000 atoms) psm finds, for instance, six locations:



things to take into account

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

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

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