Evolutionary Computation

2023/24

Master Artificial Intelligence

Arno Formella

Departamento de Informática
Escola Superior de Enxeñaría Informática
Universidade de Vigo

23/24

Simulated Annealing cooling schemes

- It seems that any monotonely decreasing function, that is neither too steep at the beginning and neither to slow decreasing in the end, might serve.
- Cooling can be implemented differently in each dimension in multi-dimensional problems.
- The entire process can be restarted from another location in search space (Monte Carlo approach around inner simulated annealing process).

Simulated Annealing vs. Tabu Search

- Both methods belong to the random path methods:
  - there is only one individual
  - the moves in the search space explore the neighborhood in a random manner
  - a move is accepted whenever
    - TS: a random, but not tabu, move among the improving once in first place and non-improving onces in second place.
    - SA: a random move that fulfills the annealing condition
- Tabu search is, as improvements are prefered, a local search method that finds local minima;
  which might not be the case for simulated annealing (you need to add some final local search approach).

memetic algorithms

The population based algorithms usually do not perform local optimization of the individuals. Including such an approach is called a memetic algorithm:

- use an adequate local search strategy (e.g. steepest decent, iterated local search etc.) to improve the fitness of the individuals
- this improvement of an individual is also called *individual learning*
- the result of the local optimization:
  - might change the genotype of the individual (Lamarckian learning), i.e., the changed individual participates in the genetic algorithm, or
  - might not change the genotype of the individual (Baldwinian learning), i.e., the population is not changed
- the local search can be restricted to a certain part of the population, for instance, the best ranked ones
- the local search can be restricted to be performed only after a certain amount of iterations in the overall genetic algorithm

biogeography-based optimization

- The biogeography-based optimization uses the idea of populations evolving at islands that once in a while exchange individuals via migration.
- Implemented as further meta-heuristic in the framework of genetic algorithms (especially for separable objective function, where optimization of their convex combination is an option).
- Can be applied to particle swarm optimization as well (I haved found implementation examples, e.g. from 2017).

much more nature-inspired work done

Without going into details, there is a huge bunch of other nature-inpired optimization heuristics (see introduction as well):

- ant lion optimization
- artificial bee colonies
- bat algorithm
- dragonfly algorithm
- firefly optimization
- grasshopper optimization
- grey wolf optimizer
- whale optimization
- invasive weed optimization
- ...
- cristalization of materials
- great deluge algorithm
- gravitational search algorithm
- etc. etc. etc.

issues to be considered

The following issues should be considered when implementing and using a certain optimization approach for a given problem:

- implementation difficulty
- number of parameters to be adjusted
- population size
- number of objective function evaluation
- convergence velocity (convergence profile)
- handling of local optima (stucking)
- handling of restrictions
- statistically correct evaluation on benchmark problems and MonteCarlo runs

efficiency issues to be considered

The following efficiency aspect might be useful in a certain implementation of an optimization approach on a given platform:

- use of parallelization
- use of caching
- use of efficient data structures
- use of adequate precision in calculation
- use of approximation algorithms (especially in early phases for the objective function computation)
- use good random number generators

hyper-parameter tuning

Many of the different heuristics seen so far have a certain number of free parameters that must be fixed when starting the optimization, but

- we might use another optimization algorithm that tries to find a good parameter set for the original algorithm
- for instance, we run the algorithm on a suitable benchmark suite to obtain good free parameters, and then run the optimization with these settings on the real problems we are interested to solve
- this process can be iterated, i.e., each use of the optimizer might adapt to a certain degree its free parameters
- eventually we arrive at an online-algorithm which adapts its parameters with each problem it is faced to

extension: an optimization problem

Given for a undirected graph certain properties:

- number of nodes
- number of edges
- distribution of node degrees
- assortativity of the interconnexions

$$a = \frac{\sum_{(i,j)}(d_i - \overline{d}_i) \cdot (d_j - \overline{d}_j)}{\sqrt{\sum_i (d_i - \overline{d}_i)^2} \cdot \sqrt{\sum_j (d_j - \overline{d}_j)^2}} \quad (1)$$

- distribution of length of edges
  (according some coordinates of the nodes in the plane)
- must be connected

Find a (large) random graph that fulfills (more or less) all properties.

example

- number of nodes: 1.000.000
- number of edges: 149.000.000
- distribution of node degrees: facebook distribution
- assortativity of the interconnexions: -0.189 (facebook)
- distribution of length of edges: population in Germany
- must be connected

current state-of-solution

Input:

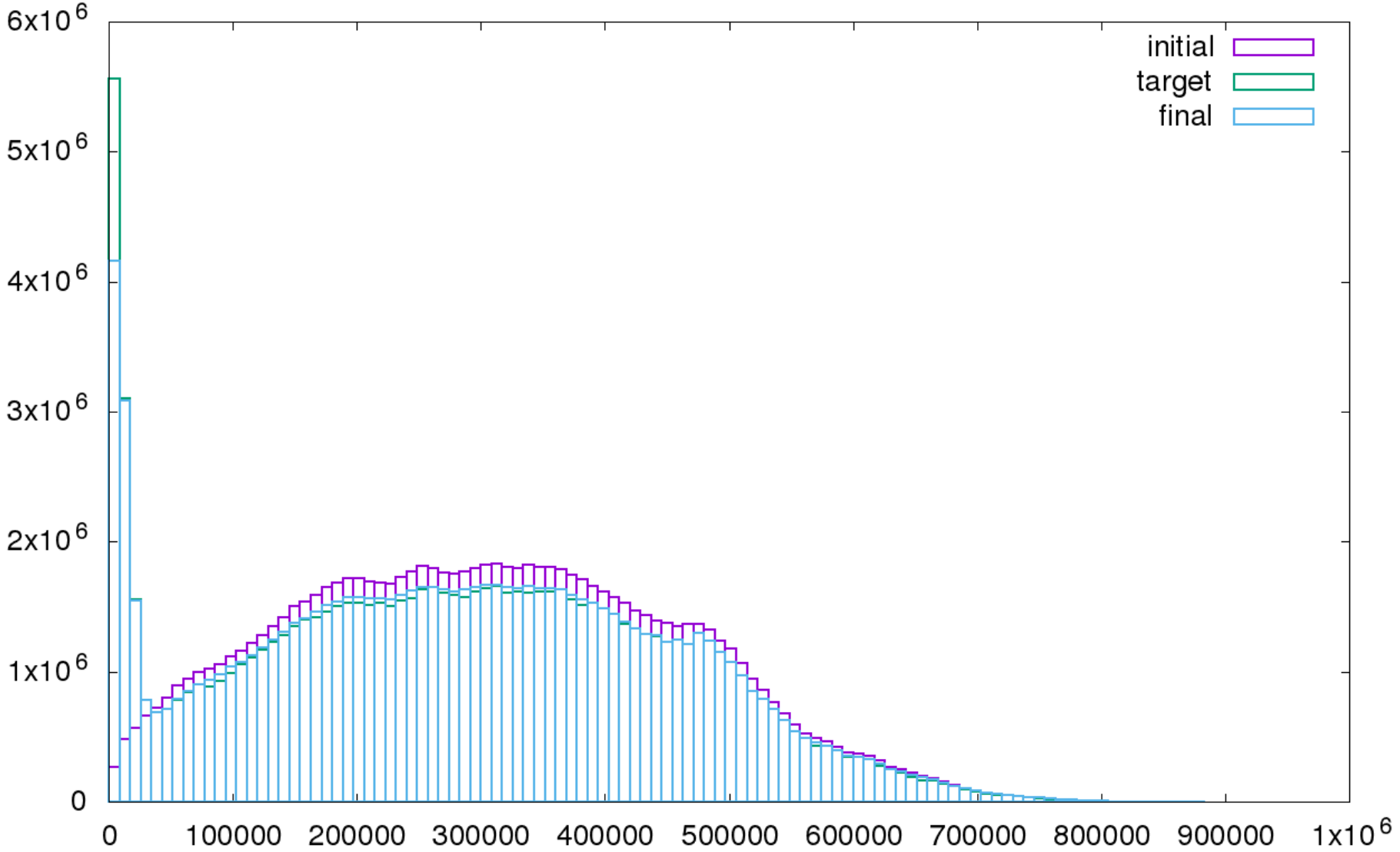
- fix number of nodes
- fix degree distribution
- scale degree distribution, hence, get number of edges
- fix target assortativity
- fix target length distribution
- must be connected

current state-of-solution

Algorithm:

- randomized Havel-Hakimi-Horvat-algorithm generates graph with exact degree distribution
- alternating annealing algorithm using some type of 2-opt operation on graph
- connectivity tested and corrected once in a while

result



length distribution

result

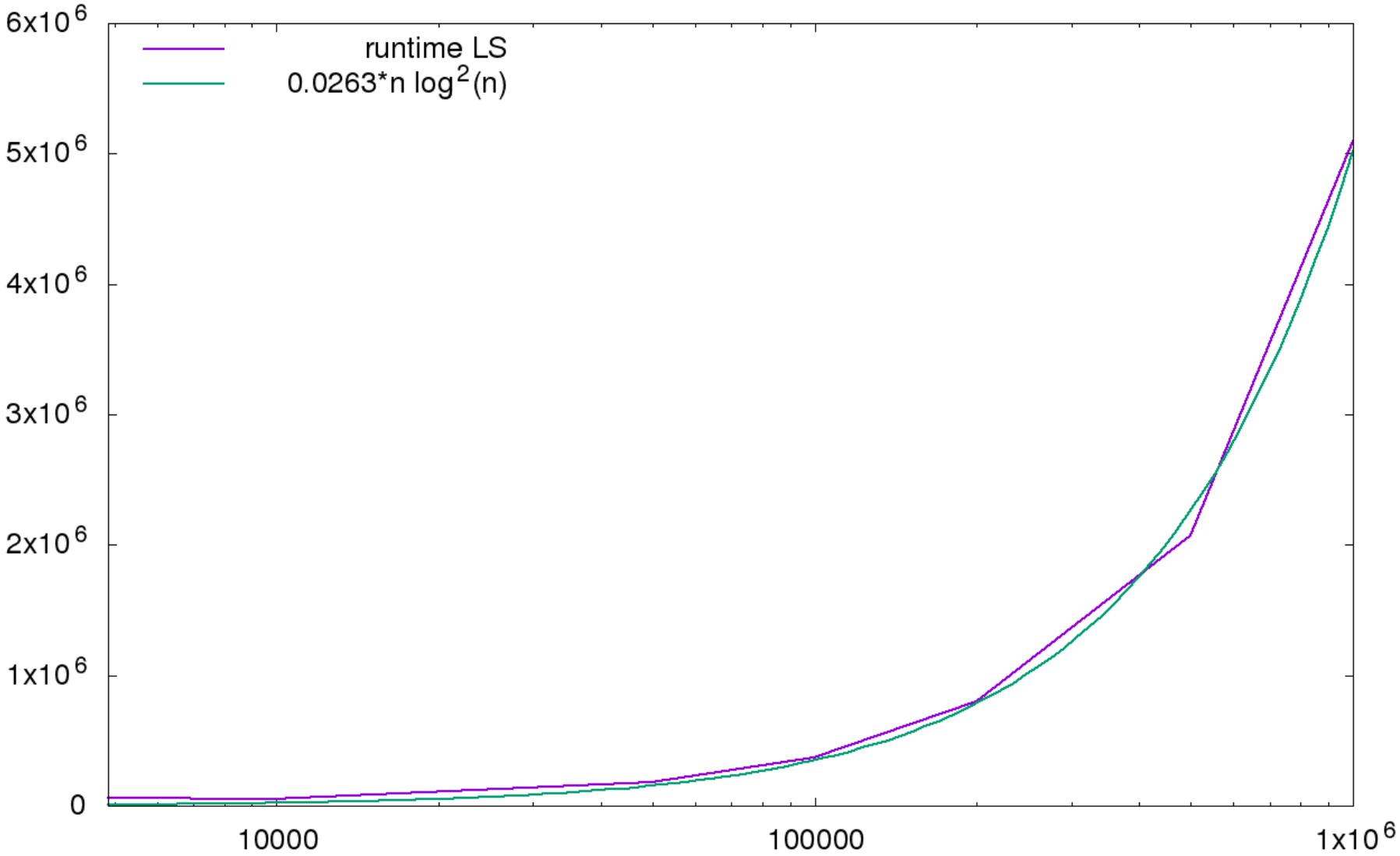
The previous slide shows:

- magenta: the initial length distribution
  (after having run the randomized HHH-algorithm),
- green: the target length distribution,
- blue: the generated length distribution
  (after having run the alternating annealing algorithm)

where the assortativity is below 0.1% of the target assortativity.

Hence, we found a connected graph with 1.000.000 nodes and 149.000.000 edges that has exactly the given degree distribution and very close assortativity and length distribution regarding the given target values.

runtime of graph generating tool



runtime versus number of nodes in [ms]

last but not least

that's it, folks ... thanks. questions?