Genetic Algorithms for Map Labeling

Steven van Dijk

10th December 2001
Outline of talk

- What is the map-labeling problem?
- What are genetic algorithms (GAs)?
- A GA for the basic map-labeling problem.
- Does the algorithm scale well?
- Application of techniques to other problems.
- A GA for labeling a map with cities and rivers.
- Conclusion.
What is the map-labeling problem?

- A map contains point-, line- and area features. The depiction of the feature name on the map is called its label.

- The map-labeling problem: place the labels of the features on the map.
What is the map-labeling problem?

Why is this a difficult problem?

1. Even basic instances are NP-hard. (Exhaustive search has exponential scale-up.)

2. There exist numerous cartographic rules which need to be considered.
What is the map-labeling problem?

The *basic* map-labeling problem: given is a set of points in the plane. Each point has a rectangular label of fixed dimensions which can be placed in one of four positions. Find a *labeling* which assigns a position to the label of each point such that the number of free labels is maximized.
What are genetic algorithms (GAs)?

*Genetic algorithms* are heuristic solvers for combinatorial problems, based on the theory of Darwinian evolution.

Outline of algorithm:

1. initialize population of solutions
2. repeat
3. select parents from population
4. with probability $Pr_c$ perform crossover and generate children
5. with probability $Pr_m$ perform mutation on children
6. replace members of population with children
7. until termination criterion satisfied
8. return best individual
What are genetic algorithms (GAs)?

Why do GAs work?

- Schema theorem: partial solutions which contribute much to the fitness and are unlikely to be disrupted will propagate through the population. (Result of selection.)

- Building block hypothesis: (close-to) optimal solutions are assembled from partial solutions by the GA. (Result of crossover.)

Disruptive crossover:

Perfect crossover:

Shaded parts are building blocks.
What are genetic algorithms (GAs)?

Key concepts:

- Linkage: what is are the building blocks and how can they be preserved from disruption?

- Mixing: assure that parts are exchanged quickly enough to allow assembly.

Disruptive crossover:

Perfect crossover:
A GA for the basic map-labeling problem.

Encoding:  

**Initialization:** assign a random position to each label.

**Selection:** elitist recombination (see below).

**Crossover:** rival crossover (see later).

**Mutation:** no traditional mutation.

Best two of family replace parents.
A GA for the basic map-labeling problem.

Crossover is done by repeatedly choosing rival groups. Two points are *rivals* if their labels can overlap. A point together with its rivals is called a *rival group*.

Crossover is *complementary*: half of a parent is copied to a child and the other half is copied from the other parent.
A GA for the basic map-labeling problem.

After crossover the geometrically local optimizer is applied to points which may have a conflict.

The geometrically local optimizer for the map-labeling problem is slot filling:
A GA for the basic map-labeling problem.

Results: comparison against best algorithm at the time (based on simulated annealing).
Does the algorithm scale well?

Design of GA allows an analysis of its *scale-up behavior*: what happens to run time when input is doubled?

\[ RT = e_{fit} \cdot n^* \cdot t^*, \]

where

\( RT \) = run time,

\( e_{fit} \) = time needed for a single fitness evaluation,

\( n^* \) = *critical* population size, and

\( t^* \) = number of generations when \( n = n^* \).

Every term is dependent on \( l \), the input size (number of cities).
Does the algorithm scale well?

How do the terms of the formula scale?

- \( e_{fit} = O(l) \), since each city can be checked in constant time.

- \( n^* \): If the gambler’s-ruin model can be applied, prediction is \( O(\sqrt{l}) \).

- \( t^* \): If convergence models can be applied, prediction is \( O(\sqrt{l}) \).

Therefore, run time is quadratic: \( RT = O(l^2) \).

Double input \( \Rightarrow \) four times the computation time. Compare with exponential scale-up of exhaustive search.

Question: can the models be applied?
Does the algorithm scale well?

Question: can the models be applied?

Short answer: YES.
Does the algorithm scale well?

Assumptions of models can be satisfied because:

- Fitness function can be kept simple (uniformly scaled, semi-separable, and additively decomposable).

- Crossover is linkage-respecting and mixes well.

- Disruption is minimized by the geometrically local optimizer.

Bottom line: theoretical insights can be used to design efficient genetic algorithms for real-world problems.