Centrum Wiskunde & Informatica

Genetic Algorithms (I)

WISM454 Laboratory Class Scientific Computing, Jan-Willem Buurlage

April 24, 2019

Genatic Algorithms

Optimization Problems

- Optimization problem: find the best solution from a set of candidates.
- Optimality is with respect to some *objective function*

$$f : \mathcal{D} \to \mathbb{R}.$$

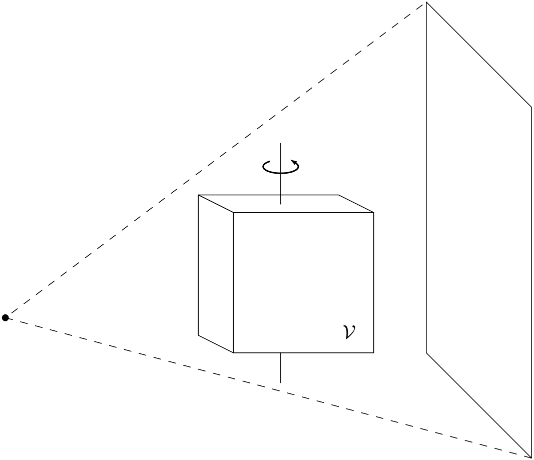
  where $\mathcal{D}$ is a set of *candidate solutions*.
- An optimization problem is expressed as:

$$\operatorname{argmax}_{x \in \mathcal{D}} f(x).$$

- Equivalently, we can consider minimization:

$$\operatorname{argmin}_{x \in \mathcal{D}} \left(-f(x)\right).$$

Example (I): Linear systems



- Tomography: $A\mathbf{x} = \mathbf{b}$
- A the physics, $\mathbf{x}$ the object, $\mathbf{b}$ the noisy measurements

Example (I): Linear systems

- Let $A : \mathbb{R}^{m \times n}$ be some matrix, and let $\mathcal{D}$ be $\mathbb{R}^n$.
- For some $\mathbf{b} \in \mathbb{R}^m$, finding a least-squares solution:

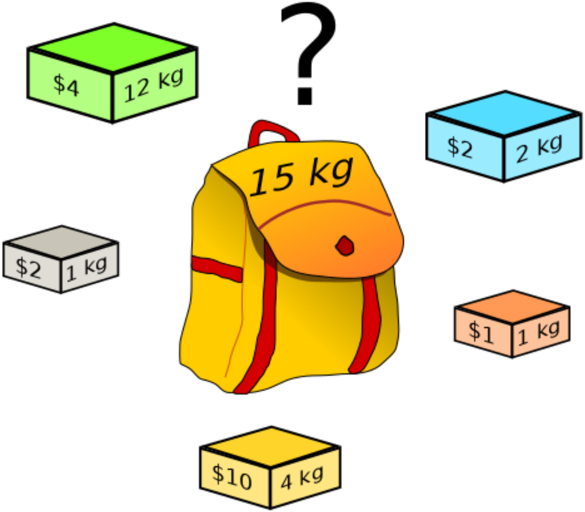
$$\text{argmin}_{\mathbf{x} \in \mathbb{R}^n} \; ||A\mathbf{x} - \mathbf{b}||_2^2,$$

  is an example of an optimization problem.
- Can add *regularization*:

$$\text{argmin}_{\mathbf{x} \in \mathbb{R}^n} \; ||A\mathbf{x} - \mathbf{b}||_2^2 + \lambda ||\mathbf{x}||_2,$$

- Linear problems like this have a lot of structure.

Example (II): Knapsack

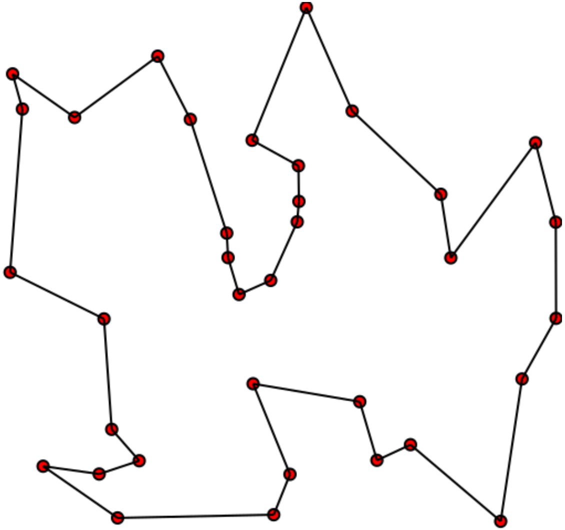


Example (II): Knapsack

- We are given a collection of m objects with weights $w_0, \ldots, w_{m-1}$, and a knapsack which can carry a weight of M.
- Which objects should we take to be as close to the maximum weight as possible?
- A bitstring $\mathbf{b}$ of length m can encode which objects we are taking.
- Here, the total weight W of a collection of objects is the objective function, and $\mathcal{D}' = \{\mathbf{b} \mid W(\mathbf{b}) \leq M\}$:

$$\text{argmax}_{\mathbf{b} \in \mathcal{D}'} \; W(\mathbf{b}).$$

Example (III): TSP

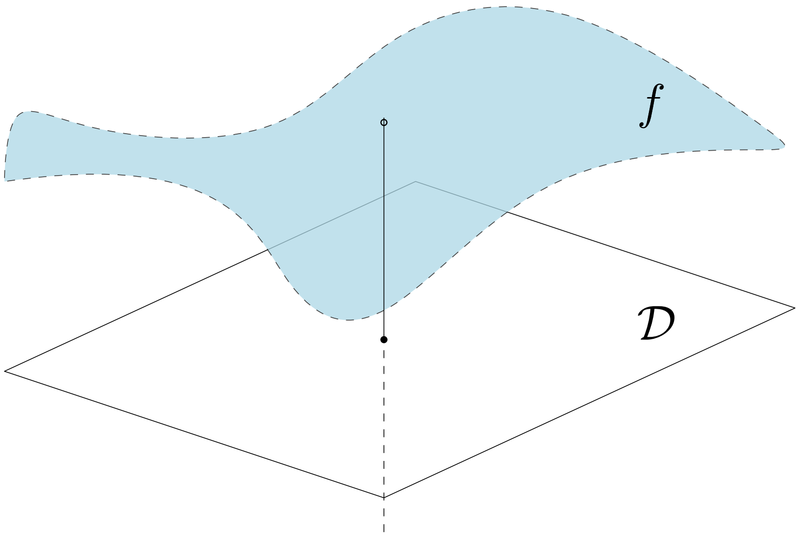


Example (III): TSP

- A path in a graph (i.e. sequence of edges) is called Hamiltonian if it visits every vertex exactly once.
- The *traveling salesman problem* (TSP) is to find the shortest Hamiltonian path of a complete graph.
- Note that if there are n vertices, the set of permutations of $\{1, \ldots, n\}$ is in 1-1 correspondence with the set of Hamiltonian paths.
- Let $D(\pi)$ be the total length of a path π. The TSP can be expressed as:

$$\operatorname{argmin}_{\pi \in \mathrm{Aut}(\{1,\ldots,n\})} D(\pi).$$

Fitness landscape



Genetic algorithms

- Genetic algorithms (GAs) are a general way to solve optimization problems.
- The main advantage: virtually no restrictions on f! (e.g. continuous, differentiable, ...)
- GAs mimic evolution as it happens in nature. A finite subset of $\mathcal{D}$, the candidate solutions, is evolved through several generations.
- *Good* current candidates survive, and combine to hopefully create even better candidates.

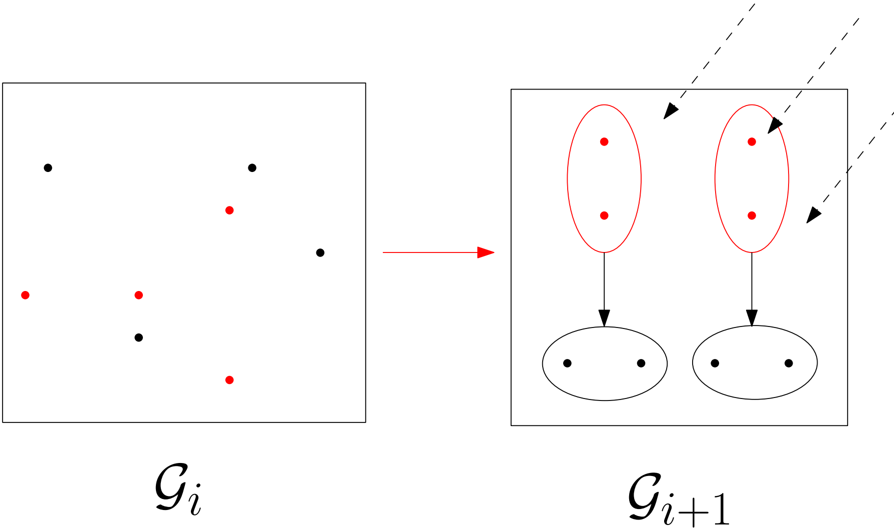
GA notions

- We denote a generation by

$$\mathcal{G} = \{x_1, \ldots, x_n\} \subset \mathcal{D}.$$

- $x_j \in \mathcal{G}$ are the *members* or *chromosomes* of the generation.
- Three main operations: selection, combination and mutation.
- Starting from a usually random initial generation, these three steps define the evolution.
- Short overview today (but enough information to start designing our software!), more details next week.

GA overview



1. Selection

- To *select* the best candidates of the current generation, we can simply evaluate the objective function.
- However, the best and worst members can have very similar objective values!
- Instead ranking, or scaling is a better metric for defining the *fitness* of a solution.
- Typically, the members that are selected to survive in each generation are chosen randomly, but biased to the *fittest* members.
- For example, using a discrete distribution (with pdf of fitness divided by total fitness)!

2. Combination

- After a number of members have been selected to survive, a number of these survivors will be selected for reproduction.
- Pairs of survivors, e.g. x_0 and x_1, generate offspring using some combination operator $\mathcal{C}$:

$$(y_0, y_1) = \mathcal{C}(x_0, x_1).$$

- Many choices for $\mathcal{C}$, e.g. crossover.

3. Mutation

- The survivors and their offspring together make up the next generation.
- They are also subjected to *mutation*, which can be seen as small changes to the solutions. For example, low probability flips if the solution is represented as a bitstring.
- This keeps the current generation 'diverse'.

Summary of GAs

- Optimization problems are very *common* in applied mathematics.
- Genetic algorithms are a *strategy for solving these problems*, without requiring any structure.
- They are very *general*, but because they do not use the structure of e.g. the objective function, they can be less efficient than tailored methods.

Exercises (designing a GA library)

First, read the lecture notes up to and including 4.1.2.

(12.1) Make a list (on a piece of paper) of all the different concepts that are relevant for GAs. What would be a good class structure for a GA library? What are the customization points?

(12.2) Many candidate solutions can be represented as a bitstring. Describe how subsets, permutations and different numerical values can be represented. Design and implement a ‘bitstring‘ type. What methods should it support?

(12.3) Make a mock implementation of Algorithm 4.1 in C++. Use the user-defined types that you have designed in (11.1). Define the signature of the auxiliary functions that you will need.