DM812
METAHEURISTICS

Lecture 6
Evolutionary Algorithms

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
<marco@imada.sdu.dk>

Outline

Outline

1. Evolutionary Algorithms

Evolutionary Algorithms

Key idea (Inspired by Darwinian model of biological evolution):
Maintain a population of individuals that compete for survival, and generate new individuals, which in turn again compete for survival

Iteratively apply genetic operators mutation, recombination, selection to a population of candidate solutions.

- Mutation introduces random variation in the genetic material of individuals (unary operator)
- Recombination of genetic material during reproduction produces offspring that combines features inherited from both parents (N-ary operator)
- Differences in evolutionary fitness lead selection of genetic traits ('survival of the fittest').

Original Streams

- Evolutionary Programming [Fogel et al. 1966]:
  - mainly used in continuous optimization
  - typically does not make use of recombination and uses stochastic selection based on tournament mechanisms.
  - often seek to adapt the program to the problem rather than the solutions

- Evolution Strategies [Rechenberg, 1973; Schwefel, 1981]:
  - similar to Evolution Strategies (developed independently)
  - originally developed for (continuous) numerical optimization problems;
  - operate on more natural representations of candidate solutions;
  - use self-adaptation of perturbation strength achieved by mutation;
  - typically use elitist deterministic selection.

- Genetic Algorithms (GAs) [Holland, 1975; Goldberg, 1989]:
  - mostly for discrete optimization;
  - often encode candidate solutions as bit strings of fixed length, (which is now known to be disadvantageous for combinatorial problems such as the TSP).

Terminology

Individual	$\Longleftrightarrow$	Solution to a problem
Genotype space	$\Longleftrightarrow$	Set of all possible individuals determined by the solution encoding (representation)
Phenotype space	$\Longleftrightarrow$	Search space
Population	$\Longleftrightarrow$	Set of candidate solutions
Chromosome	$\Longleftrightarrow$	Representation for a solution (*e.g.*, set of parameters)
Fitness	$\Longleftrightarrow$	Quality of a solution
Gene and Allele	$\Longleftrightarrow$	Part and value of the representation of a solution (*e.g.*, parameter or degree of freedom)
Crossover Mutation	$\Longleftrightarrow$	Search Operators
Natural Selection	$\Longleftrightarrow$	Promoting the reuse of good solutions

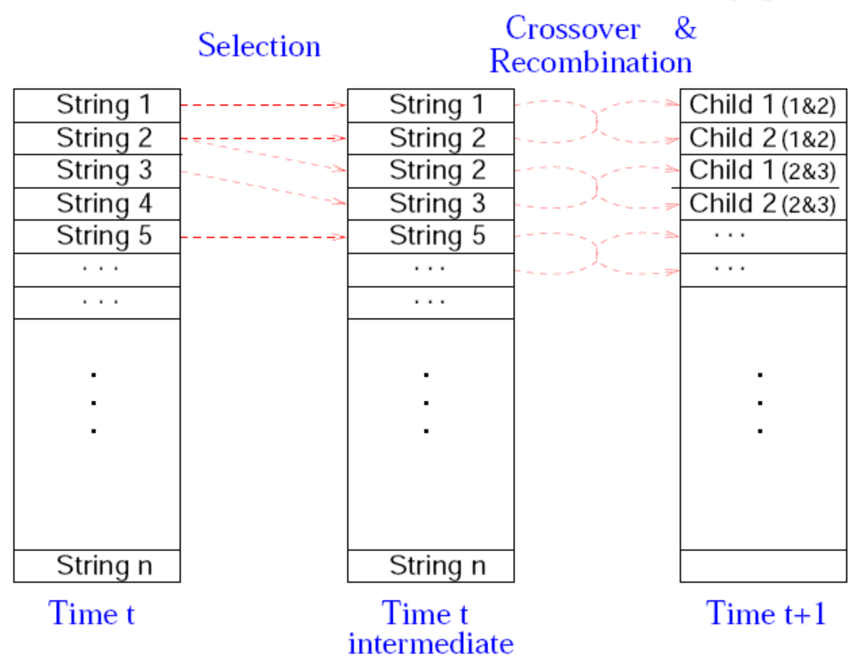
Evolutionary Algorithm (EA):
determine initial population sp

while termination criterion is not satisfied: **do**

    generate set spr of new candidate solutions
        by recombination

    generate set spm of new candidate solutions
        from spr and sp by mutation

    select new population sp from
        candidate solutions in sp, spr, and spm



Problem: Pure evolutionary algorithms often lack capability of sufficient search intensification.

Solution: Apply subsidiary local search after initialization, mutation and recombination.

Memetic Algorithms [Dawkins, 1997, Moscato, 1989]

- transmission of memes, mimicking cultural evaluation which is supposed to be direct and Lamarckian
- (aka Evolutionary Local Search, or Hybrid Evolutionary Algorithms)

Memetic Algorithm (MA):
determine initial population `sp`
perform subsidiary local search on `sp`
while termination criterion is not satisfied: **do**
  generate set `spr` of new candidate solutions
    by recombination
  perform subsidiary local search on `spr`
  generate set `spm` of new candidate solutions
    from `spr` and `sp` by mutation
  perform subsidiary local search on `spm`
  select new population `sp` from
    candidate solutions in `sp`, `spr`, and `spm`

Solution representation

Separation between solution encode/representation (genotype) from actual solution (phenotype)

Example

- genotype set made of strings of length l whose elements are symbols from an alphabet $\mathcal{A} \Rightarrow$ set of all individuals $\mathcal{A}^l$
  - the elements of strings are the genes
  - the values that each element can take are the alleles
- the search space is $\mathcal{X} \subseteq \mathcal{A}^l$,
- if the strings are member of a population they are called chromosomes and their recombination crossover
- an expression maps individual to solutions (phenotypes) $c : \mathcal{A}^l \mapsto \mathcal{S}$
- strings are evaluated by $f(c(x)) = g(x)$ which gives them a fitness

Example

1001010 1101100 0111010 1010010 1000010

0101110 0111101 0110110 1101000 1010101

10010 101101100011101 010100101000010

01011 100111101011011 011010001010101

Which Produces the Offspring

01011101101100011101011010001010101

10010100111101011011010100101000010

Note: binary representation is appealing but not always good (in constrained problems binary crossovers might not be good)

Conjectures on the goodness of EA

schema: subset of $\mathcal{A}^l$ where strings have a set of variables fixed.
Ex.: 1 * * 1

- exploit intrinsic parallelism of schemata
- Schema Theorem:

$$E[N(S,t+1)] \geq \frac{F(S,t)}{\bar{F}(S)} N(s,t)[1-\epsilon(S,t)]$$

- a method for solving all problems $\Rightarrow$ disproved by No Free Lunch Theorems
- building block hypothesis

Initial Population

- Which size? Trade-off
- Minimum size: connectivity by recombination is achieved if at least one instance of every allele is guaranteed to be present at each gene. Ex: if binary:

$$P_2^* = (1-(0.5)^{M-1})^l$$

  for $l = 50$, it is sufficient $M = 17$ to guarantee $P_2^* > 99.9\%$.
- Generation: often, independent, uninformed random picking from given search space.
- Attempt to cover at best the search space, eg, Latin hypercube, Quasi-random (low-discrepancy) methods (Quasi-Monte Carlo method).
- But: can also use multiple runs of construction heuristic.

Selection

Main idea: selection should be related to fitness

- Fitness proportionate selection (Roulette-wheel method)

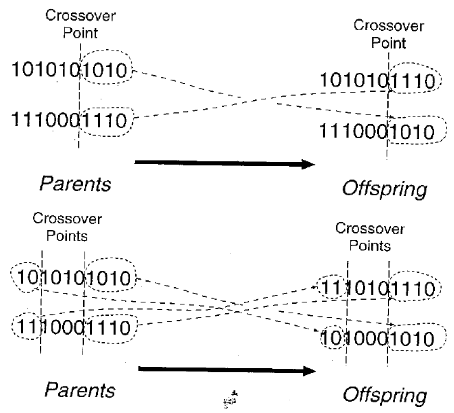
$$p_i = \frac{f_i}{\sum_j f_j}$$

- Tournament selection: a set of chromosomes is chosen and compared and the best chromosomes chosen.
- Rank based and selection pressure
- Fitness sharing (aka niching): probability of selection proportional to the number of other individuals in the same region of the search space.

Recombination (Crossover)

- Binary or assignment representations
  - one-point, two-point, m-point (preference to positional bias w.r.t. distributional bias)
  - uniform cross over (through a mask controlled by a Bernoulli parameter p)
- Permutations
  - Partially mapped crossover (PMX)
  - Mask based crossover
  - Order crossover (OX)
  - Cycle crossover (CX)
- Sets
  - greedy partition crossover (GPX)
- Real vectors
  - arithmetic crossovers
  - k-point crossover

Example: crossovers for binary representations



- Crossovers appear to be a crucial feature of success
- Therefore, more commonly: ad hoc crossovers
- Two off-springs are generally generated
- Crossover rate controls the application of the crossover. May be adaptive: high at the start and low when convergence

Mutation

- Goal: Introduce relatively small perturbations in candidate solutions in current population + offsprings obtained from recombination
- Typically, perturbations are applied stochastically and independently to each candidate solution
- Mutation rate controls the application of bit-wise mutations. It may be adaptive: low at the start and high when convergence
- Possible implementation through Poisson variable which determines the m genes which are likely to change allele.
- Can also use subsidiary selection function to determine subset of candidate solutions to which mutation is applied.
- With real vector representation: Gaussian mutation

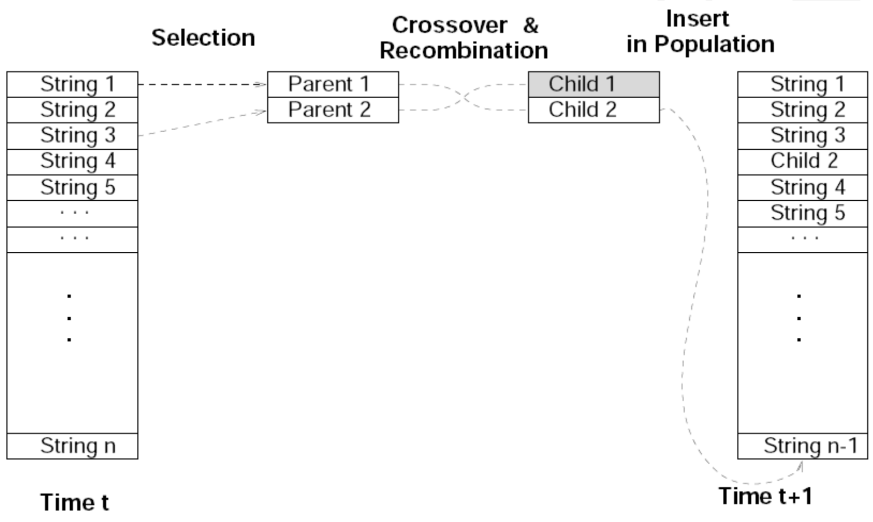
Subsidiary local search

- Often useful and necessary for obtaining high-quality candidate solutions.
- Typically consists of selecting some or all individuals in the given population and applying an iterative improvement procedure to each element of this set independently.

New Population

- Determines population for next cycle (generation) of the algorithm by selecting individual candidate solutions from
  - current population +
  - new candidate solutions from recombination, mutation (and subsidiary local search).
- Generational Replacement (λ, μ): $\lambda \leftarrow \mu$
- Elitist strategy $(\lambda + \mu)$ the best candidates are always selected
- Steady state (most common) only a small number of least fit individuals is replaced
- Goal: Obtain population of high-quality solutions while maintaining population diversity.

  Survival of the fittest and maintenance of diversity (duplicates avoided)



Example

A memetic algorithm for TSP

- **Search space:** set of Hamiltonian cycles
  Tours represented as permutations of vertex indexes.
- **Initialization:** by randomized greedy heuristic (partial tour of $n/4$ vertices constructed randomly before completing with greedy).
- **Recombination:** greedy recombination operator GX applied to $n/2$ pairs of tours chosen randomly:
  1) copy common edges (param. p_e)
  2) add new short edges (param. p_n)
  3) copy edges from parents ordered by increasing length (param. p_c)
  4) complete using randomized greedy.
- **Subsidiary local search:** LK variant.
- **Mutation:** apply double-bridge to tours chosen uniformly at random.
- **Selection:** Selects the μ best tours from current population of $\mu + \lambda$ tours (=simple elitist selection mechanism).
- **Restart operator:** whenever average bond distance in the population falls below 10.