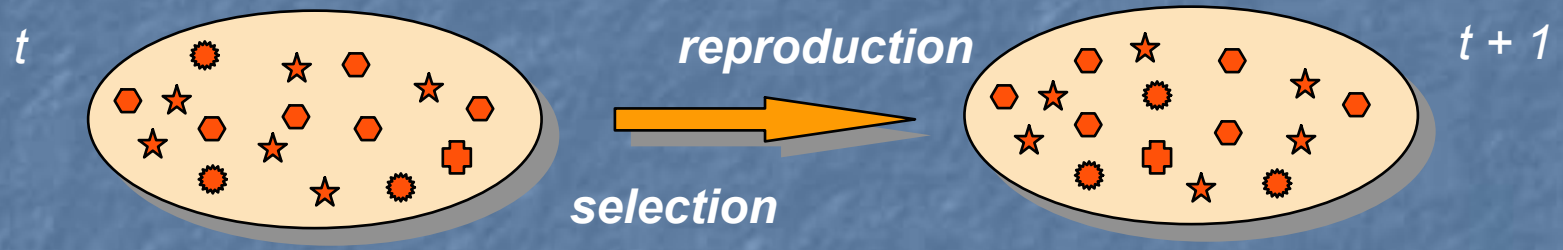
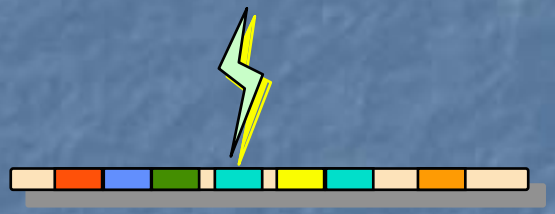
Genetic Algorithm: introduction

The Metaphor

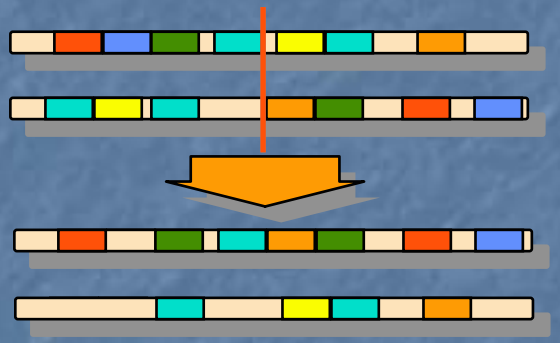
EVOLUTION		PROBLEM SOLVING
Individual	⟷	Candidate Solution
Fitness	⟷	Quality
Environment	⟷	Problem

The Ingredients





mutation

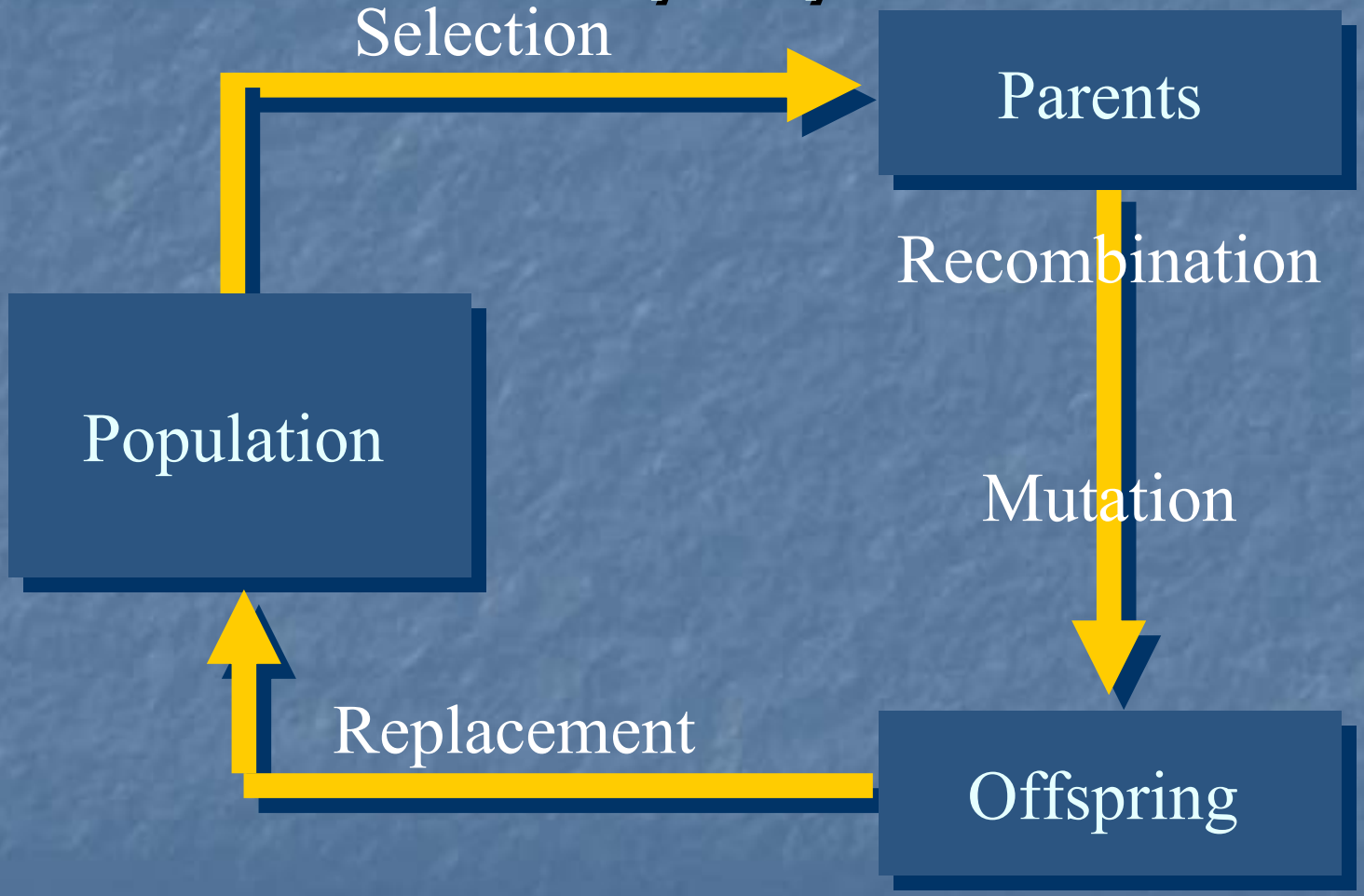


recombination

The Evolution Mechanism

- Increasing diversity by genetic operators
  - mutation
  - recombination
- Decreasing diversity by selection
  - of parents
  - of survivors

The Evolutionary Cycle



Domains of Application

- Numerical, Combinatorial Optimisation
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life

Performance

- Acceptable performance at acceptable costs on a wide range of problems
- Intrinsic parallelism (robustness, fault tolerance)
- Superior to other techniques on complex problems with
  - lots of data, many free parameters
  - complex relationships between parameters
  - many (local) optima

Advantages

- No assumptions on problem space
- Widely applicable
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provide many alternative solutions

Disadvantages

- No guarantee for optimal solution within finite time
- Weak theoretical basis
- May need parameter tuning
- Often computationally expensive, i.e. slow

Summary

EVOLUTIONARY COMPUTATION:

- is based on biological metaphors
- has great practical potentials
- is getting popular in many fields
- yields powerful, diverse applications
- gives high performance against low costs

Prototyping

Completeness vs. Consistency

- *Completeness* in order to absorb the differences among specimen that belong to the same class
- *Consistency* so as to distinguish among similar samples belonging to different classes

Prototyping

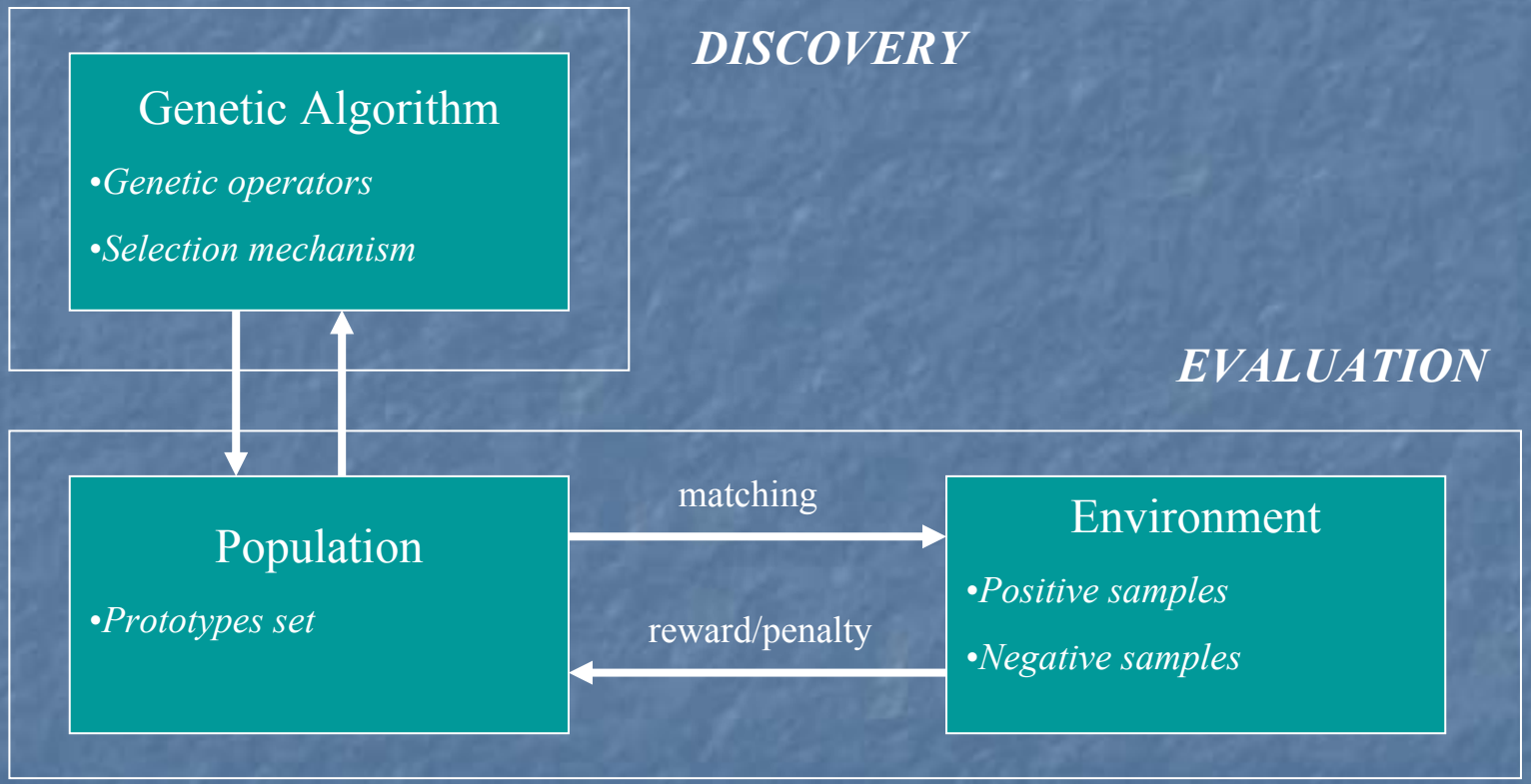
By hand vs. Automatically

- *By hand:* feasible when the samples to classify can be thought as noisy instances of a well-defined, even if very large, set of models.
- *Automatically:* useful when it is impossible to model an unpredictable amount of variability characterizing the specimen for the considered application.

Evolutionary Learning Systems

- learn-by-example process
- an efficient mechanism to explore complex high dimensional feature space
- an explicit mechanism to verify the effectiveness of the prototypes

Genetic Learning



… but

- Simple GAs do not allow the evolution of a set of prototypes because the fitness of each individual is evaluated independently from the fitness of the other ones.
- The task of a GA is to optimize the average value of the fitness in the population.
- Consequently, the optimal solution provided by such a process consists entirely of genetic variations of the best prototype.

… then

- ***How we can allow the formation and the maintenance of sets of prototypes each solving different parts of the problem at hand ?***

NICHING

- *A **niche** is a stable sub-population of competing prototypes sharing the same environmental resources.*

Niching Methods

Crowding

The new offsprings are compared against similar individuals in the population and replace the ones with lower fitness

Sharing

*The individuals in the population are forced to share the resources(**resource sharing**) or the fitness (**fitness sharing**).*

Deterministic Crowding

Do n/2 times

Select two parents p1 and p2;
Cross them, yielding s1 and s2;
Apply genetic operators, yielding c_1 and c_2;
If $[d(p_1,c_1)+ d(p_2,c_2)] \leq [d(p_1,c_2)+ d(p_2,c_1)]$
    if $\phi(c_1) > \phi(p_1)$ replace p_1 with c_1
    if $\phi(c_2) > \phi(p_2)$ replace p_2 with c_2
Else
    if $\phi(c_2) > \phi(p_1)$ replace p_1 with c_2
    if $\phi(c_1) > \phi(p_2)$ replace p_2 with c_1

Resource Sharing

Induces competition for limited resources (e.g. sample of the training set)

for each sample in the training set:

assign a reward ρ to each prototype covering it depending on its fitness and the total number of competing prototypes

for each prototype

fitness = sum($reward_i$)

Fitness Sharing

- ***Fitness sharing*** *derates the fitness of* ***similar*** *individuals within the population.*
- ***Shared Fitness*** $\phi_{sh,t}(\sigma_i) = f(\sigma_i)/m_t(\sigma_i)$
- ***Niche Count*** $m_t(\sigma_i) = \sum_{\sigma_j \in P_t} Sh_t(\sigma_i, \sigma_j)$
- ***Sharing Function*** $$Sh_t(\sigma_i, \sigma_j) = \begin{cases} 1 - \left(\frac{d(\sigma_i, \sigma_j)}{\delta_{sh}} \right)^{\alpha_{sh}} & \text{if } d(\sigma_i, \sigma_j) < \delta_{sh} \\ 0 & \text{otherwise} \end{cases}$$
- *$d(\sigma_i, \sigma_j)$ is the distance function and δ_{sh} is the niche radius.*

Summary

NICHING METHODS:

- motivated by analogy with competition for limited resources among members of natural populations
- allow the formation and maintenance of sets of prototypes for each class
- may need parameter tuning

To probe further

- IEEE Trans. On Evolutionary Computation
- Evolutionary Computation (MIT Press)
- Soft Computing (Springer)
- Applied Soft Computing (Springer)
- Int. Conf. on Evolutionary Computation (ICEC)
- Genetic and Evolutionary Computation Conference (GECCO)
- Parallel Problem Solving by Nature (PPSN)
- www.evonet.org