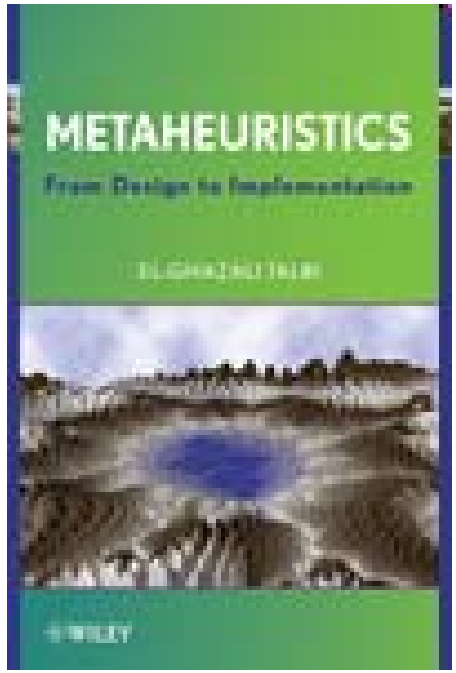
Metaheuristics : from Design to Implementation

Chap 3

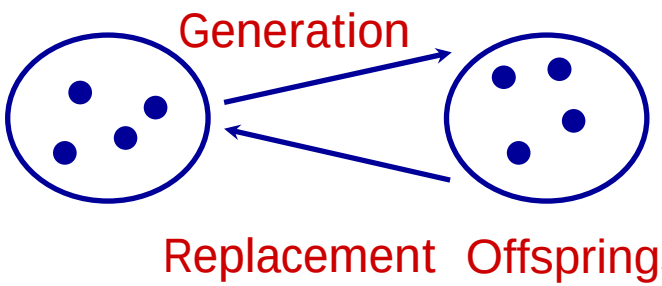
Population based Metaheuristics

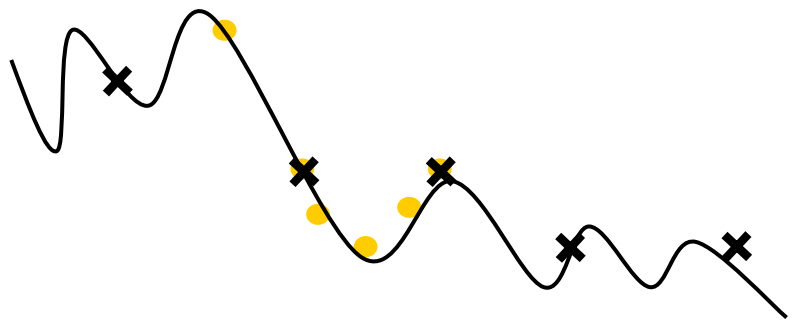


Wiley, 2009 (596pp)
ISBN: 978-0-470-27858-1

Population solution-based metaheuristics

- “Improvement” of a population of solutions
- Recombination of solutions
- Exploration Oriented :





High-level template of P-metaheuristics

Algorithm 3.1 High-level template of P-metaheuristics.

$P = P_0$; /* Generation of the initial population */

$t = 0$;

Repeat

Generate(P'_t) ; /* Generation a new population */

P_{t+1} = Select-Population($P_t \cup P'_t$) ; /* Select new population */

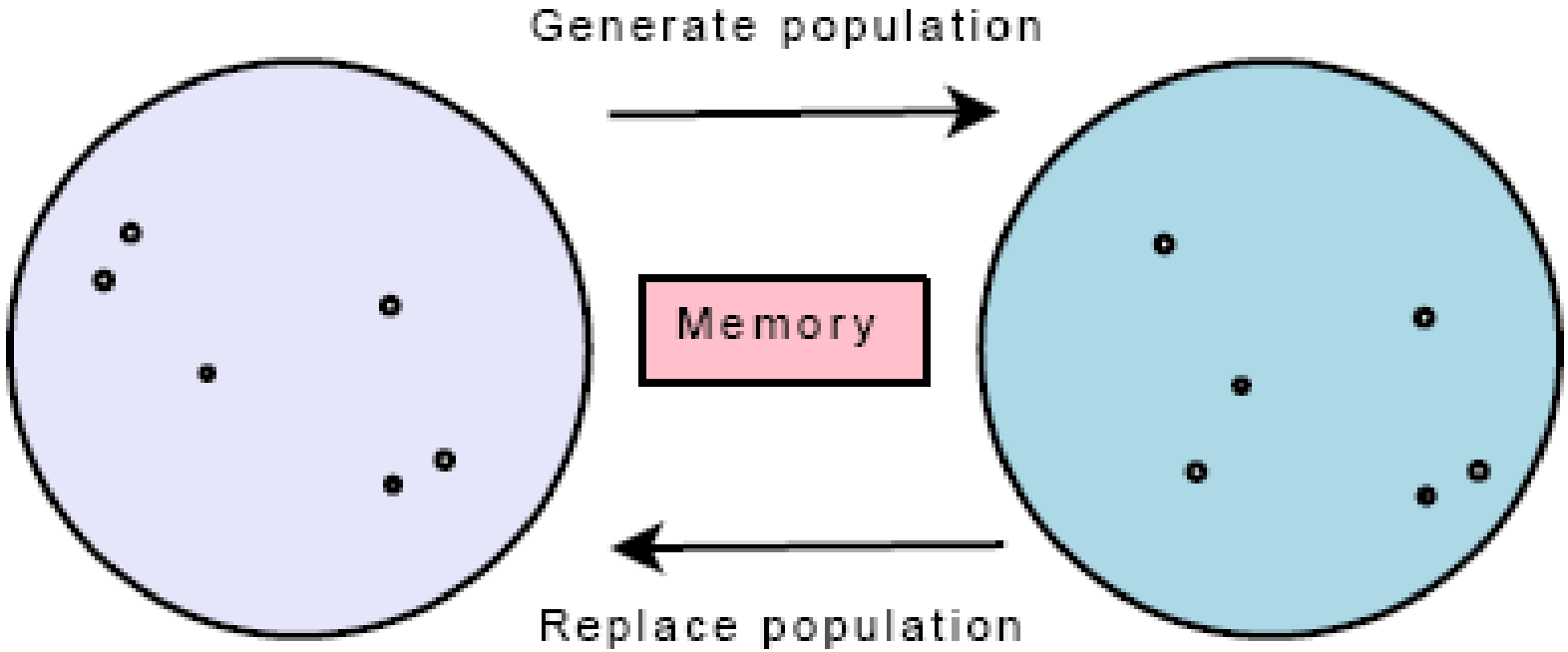
$t = t+1$;

Until Stopping criteria satisfied

Output: Best solution(s) found.

Output: Best solution found.

High-level template of P-metaheuristics

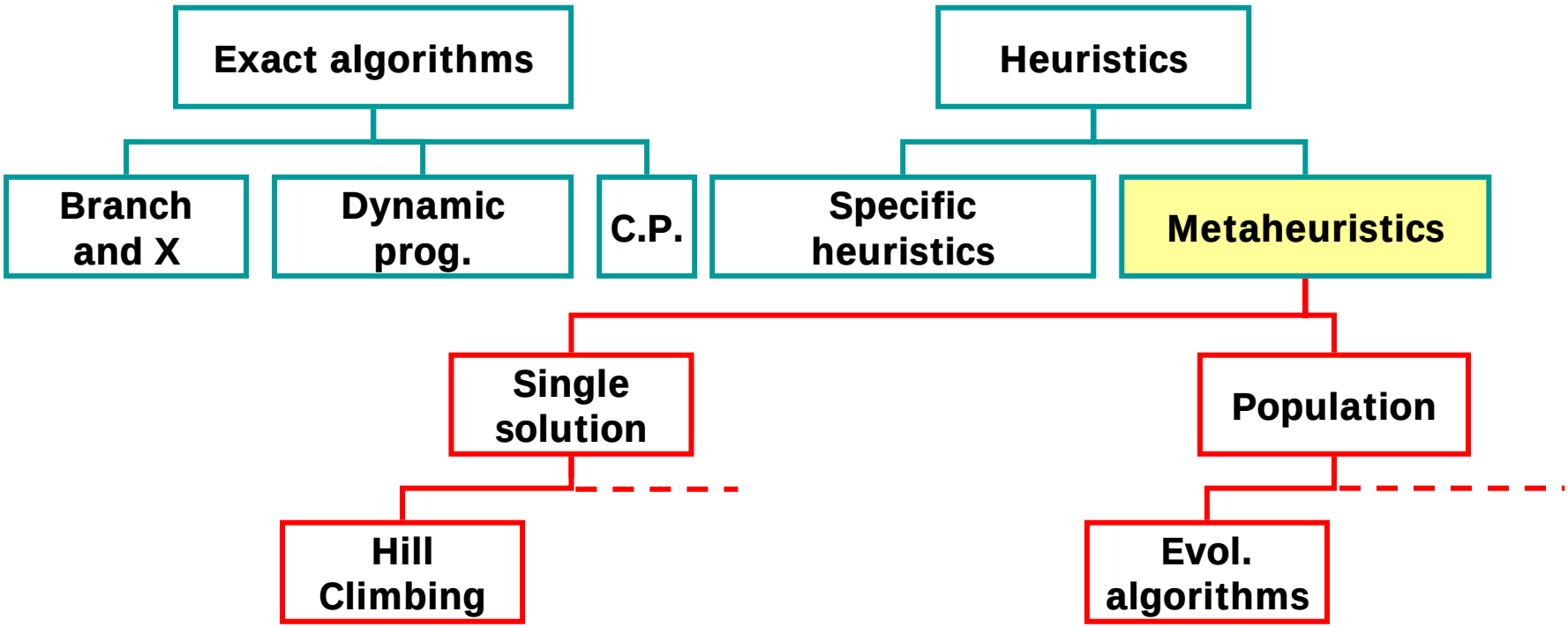


Search memory in P-metaheuristics

Table 3.1 Search memories of some P-metaheuristics.

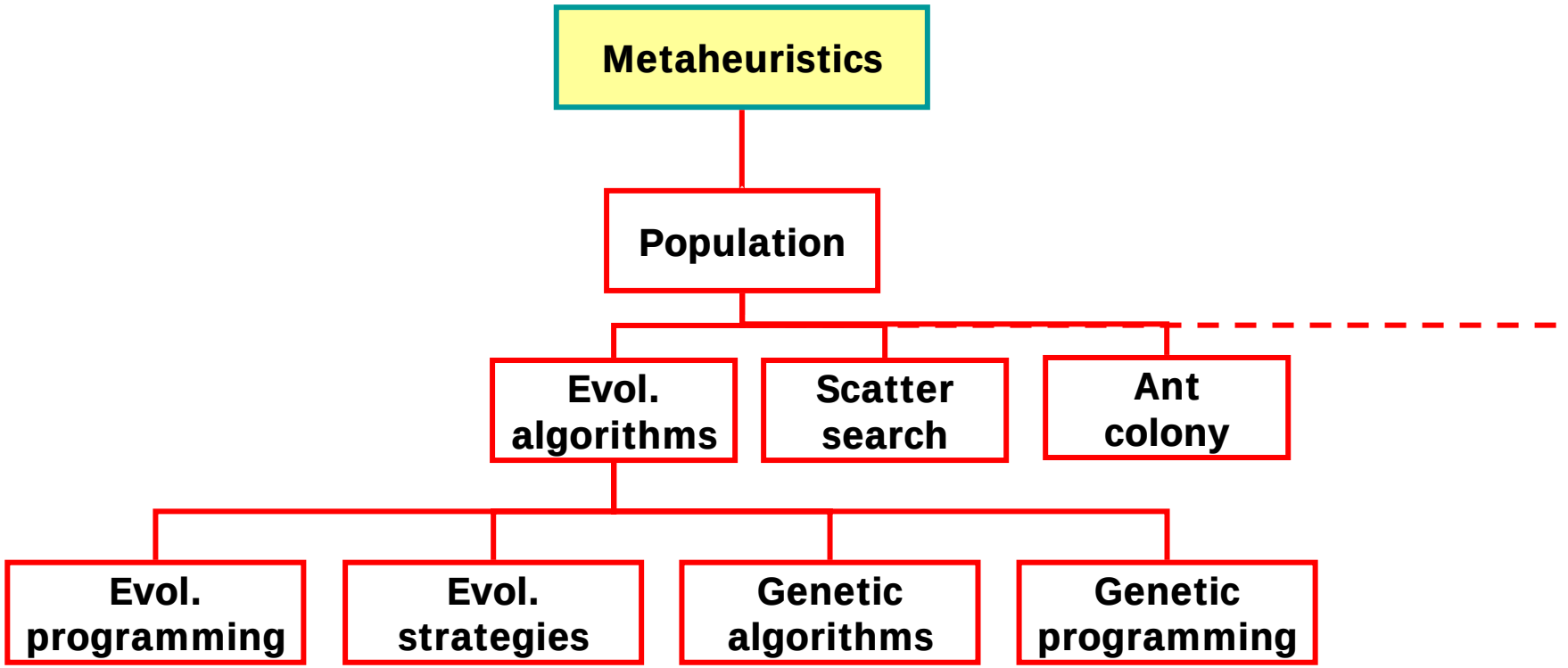
P-metaheuristic	Search memory
Evolutionary algorithms (EA)	Population of individuals
Scatter Search (SS)	Population of solutions, reference set
Ant colonies (AC)	Pheromone matrix
Estimation of Distribution Algorithms (EDA)	Probabilistic learning model
Particle Swarm Optimization (PSO)	Population of particles, best global and local solutions
Bee colonies (BC)	Population of bees
Artificial immune systems(AIS): Clonal selection	Population of antibodies

Taxonomy (metaheuristics)

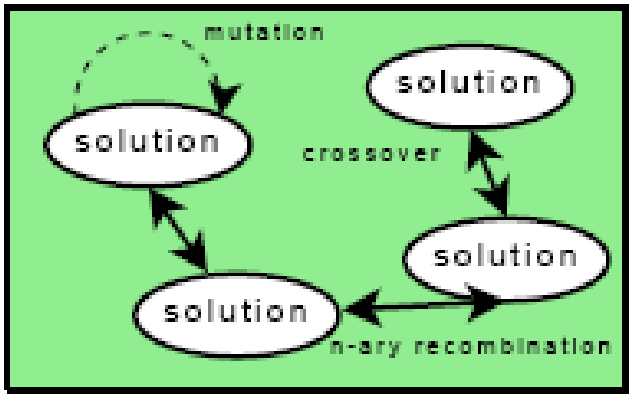


- Single solution metaheuristics are exploitation oriented
- Population-based metaheuristics are exploration oriented

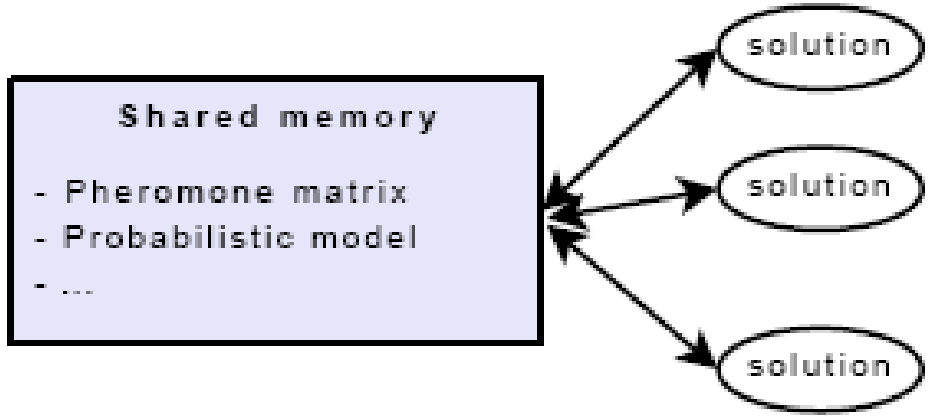
Taxonomy (Population-based Metaheuristics)



Taxonomy (Population-based Metaheuristics)



(a) Evolutionary-based P-metaheuristics: evolutionary algorithms, scatter search, ...



(b) Blackboard-based P-metaheuristics: ant colonies, estimation distribution algorithms, ...

Fig. 3.2 Evolutionary-based versus Blackboard-based strategies in P-metaheuristics.

Outline

- Common concepts for P-metaheuristics
  - Initial population
  - Stopping criteria
- Evolutionary algorithms
  - Common concepts: selection, reproduction, replacement
  - Genetic algorithms
  - Genetic programming
  - Evolution strategies, Evolutionary programming
  - Other evolutionary algorithms
    - Estimation of Distribution Algorithms (EDA)
    - Differential evolution
    - Co-evolutionary algorithms
    - Cultural algorithms
- Scatter search and path relinking
- Swarm intelligence
  - Ant colonies
  - Particle swarm optimization
- Bee colonies
- Artificial immune systems

Initial population

- Random generation
  - Pseudo-random
  - Quasi-random
- Sequential diversification
- Parallel diversification
- Heuristic initialization

Initial population: Pseudo-random / Quasi-random

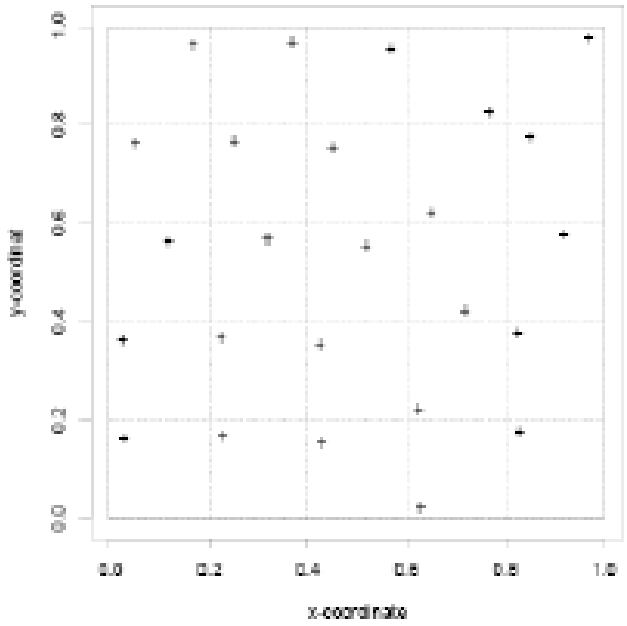


Fig. 3.3 In the Latin hypercube strategy, the search space is decomposed into 25 blocks and a solution is generated pseudo-randomly in each block.

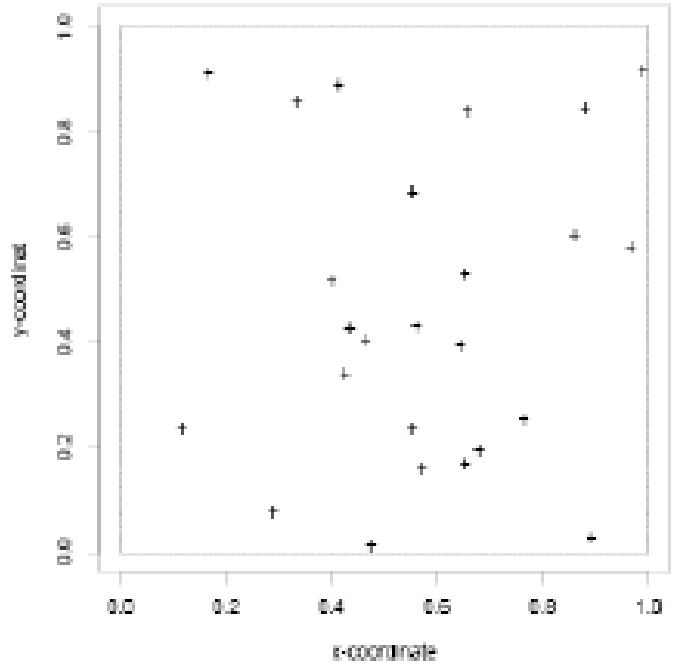


Fig. 3.4 In the pseudo-random generation, 25 solutions are generated independently in the search space.

Initial population: Hybrid approach

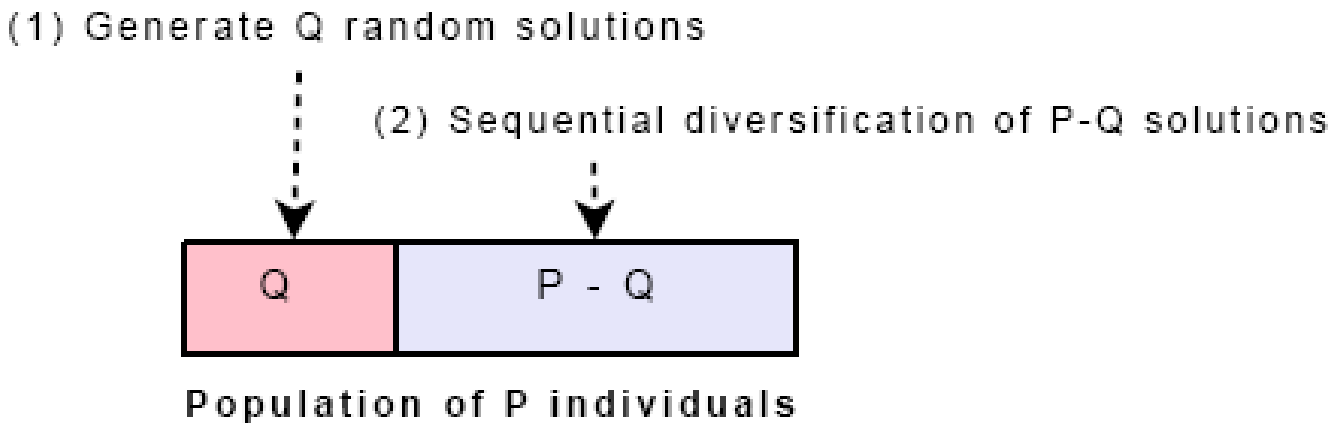


Fig. 3.6 Hybrid initialization of the population.

Stopping criteria

- Static procedure: known a priori
  - Number of iterations
  - CPU time
  - Number of evaluations
- Adaptive procedure
  - Number of iterations without improvements
  - Diversity of the population
  - Optimal or satisfactory solution is reached

Initial population

Table 3.2 Analysis of the different initialization strategies. The evaluation is better with more plus sign (+). Sequential and parallel diversification strategies provide in general the best diversity followed by the quasi-random strategy. The heuristic initialization provides in general better solutions in terms of quality but with the expense of a higher computational cost and a reduced diversity. This will depend on the fitness landscape of the tackled optimization problem. For some landscapes (e.g. flat rugged), the diversity may remain important.

Strategy	Diversity	Computational cost	Quality of initial solutions
Pseudo-random	++	+++	+
Quasi-random	+++	+++	+
Sequential diversification	++++	++	+
Parallel diversification	++++	+++	+
Heuristic	+	+	+++

Evolutionary Computation

History

- L. Fogel 1962 (San Diego, CA): *Evolutionary Programming*
- J. Holland 1962 (Ann Arbor, MI): *Genetic Algorithms*
- I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany): *Evolution Strategies*
- J. Koza 1989 (Palo Alto, CA): *Genetic Programming*

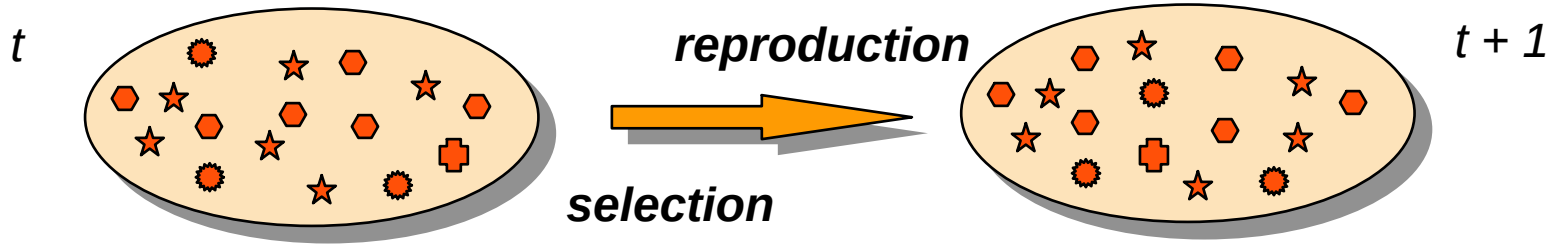
The metaphor

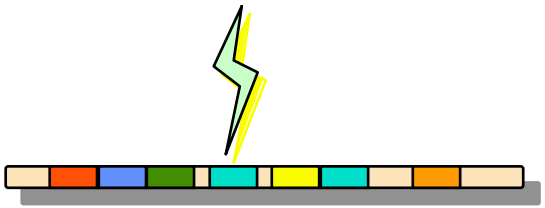
Table 3.3 Evolution process versus solving an optimization problem.

Metaphor	Optimization
Evolution	Problem solving
Individual	Solution
Fitness	Objective function
Environment	Optimization problem
Locus	Element of the solution
Allele	Value of the element (locus)

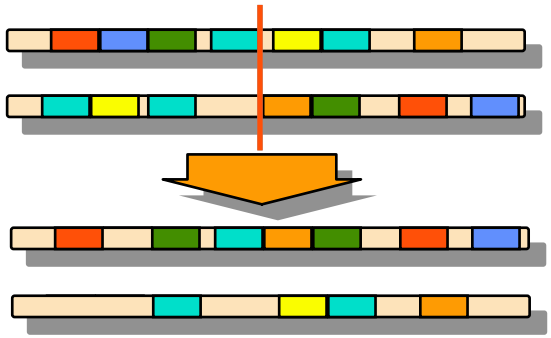
- Based on the evolution of a population of individuals
- Evolution features
  - Variation operators (crossover, mutation) to increase diversity,
  - Selection of parents, replacement by offspring to decrease diversity

The ingredients





mutation



recombination

Evolutionary cycle

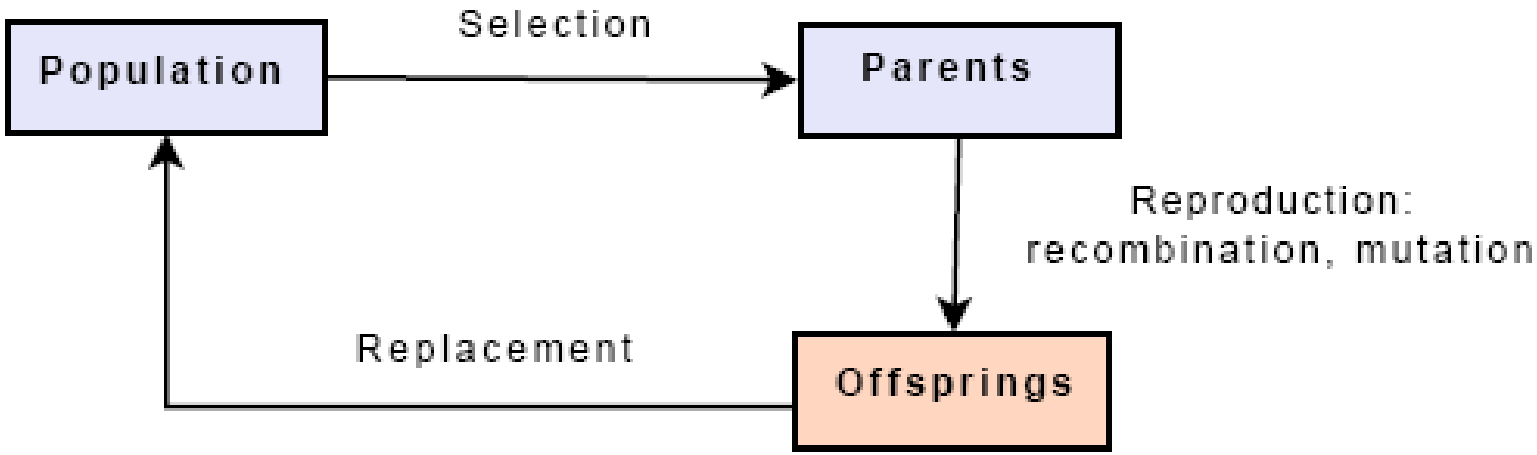


Fig. 3.7 A generation in evolutionary algorithms.

Evolutionary Algorithm procedure

Algorithm 3.2 Template of an evolutionary algorithm (EA).

Generate($P(0)$) ; /* Initial population */

$t = 0$;

While not Termination_Criterion($P(t)$) **Do**

Evaluate($P(t)$) ;

$P'(t)$ = Selection($P(t)$) ;

$P'(t)$ = Reproduction($P'(t)$); Evaluate($P'(t)$) ;

$P(t+1)$ = Replace($P(t)$, $P'(t)$) ;

$t = t + 1$;

End While

Output Best individual or best population found.

Genotype / Phenotype

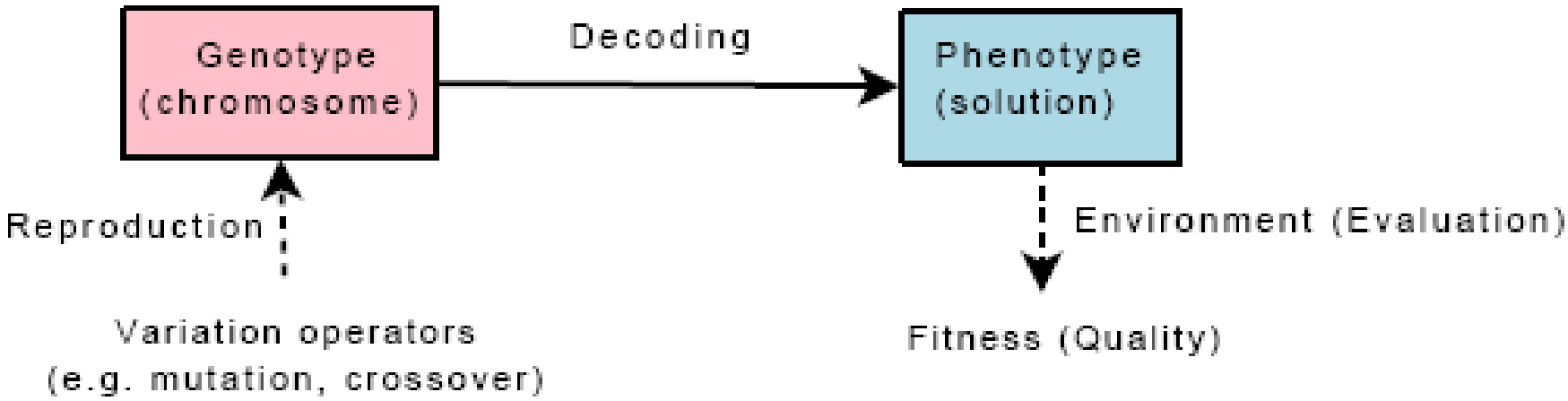


Fig. 3.8 Genotype versus phenotype in evolutionary algorithms.

Domains of application

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

Performances

- 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
  - Adaptive, dynamic problems

Advantages

- No presumptions w.r.t. 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
- Robust regards any change of the environment (data, objectives, etc)
- Co-evolution, parallelism and distribution ...

Disadvantages

- No guarantee for optimal solution within finite time (in general)
- May need parameter tuning
- Often computationally expensive, i.e. slow

Genetic Algorithms

- Developed: USA in the 1970's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

Genetic Algorithms (SGA)

Representation	Binary strings
Recombination	N-point or uniform
Mutation	Bitwise bit-flipping with fixed probability
Parent selection	Fitness-Proportionate
Survivor selection	All children replace parents
Speciality	Emphasis on crossover

Evolution Strategies

- Developed: Germany in the 1970's
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to:
  - numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
  - relatively much theory
- Special:
  - self-adaptation of (mutation) parameters standard

Evolution Strategies

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ,λ) or $(\mu+\lambda)$
Specialty	Self-adaptation of mutation step sizes

Evolutionary Programming

- Developed: USA in the 1960's
- Early names: D. Fogel
- Typically applied to:
  - traditional EP: machine learning tasks by finite state machines
  - contemporary EP: (numerical) optimization
- Attributed features:
  - very open framework: any representation and mutation op's OK
  - crossbred with ES (contemporary EP)
  - consequently: hard to say what "standard" EP is
- Special:
  - no recombination
  - self-adaptation of parameters standard (contemporary EP)

Evolutionary Programming

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic
Survivor selection	Probabilistic ($\mu+\mu$)
Specialty	Self-adaptation of mutation step sizes (in meta-EP)

Genetic Programming

- Developed: USA in the 1990's
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification...)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees, graphs
  - mutation possible but not necessary (disputed!)

Genetic Programming

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Common search components for evolutionary algorithms

- Selection strategies: which parents are selected for reproduction
- Reproduction strategies: which variation operators we use
- Replacement strategies: how the current population is updated according to the generated offsprings

The selection strategy

- The main rule: “The better is an individual, the higher is its chance of being parent”.
- Such a selection pressure will drive the population forward
- Worst individuals shouldn’t be discarded but have some chance to be selected. This may lead to useful genetic material

The most common selection strategies

- Proportional fitness assignment: absolute fitnesses are associated to individuals
- Rank based fitness assignment: relative fitnesses are associated to individuals (e.g. rank)

The most common selection strategies

- Roulette wheel selection
- Stochastic Universal Sampling (SUS)
- Tournament selection
- Rank based selection

Roulette / SUS

Individuals:	1	2	3	4	5	6	7
Fitness:	1	1	1	1,5	1,5	3	3

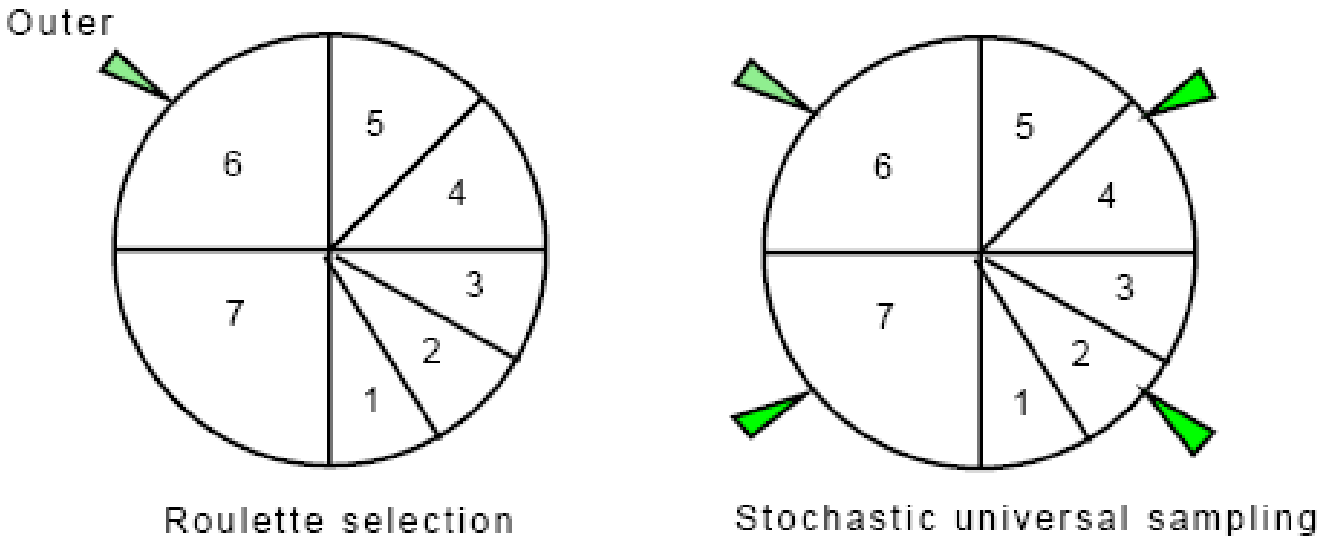


Fig. 3.11 Roulette selection strategies. In the standard roulette selection, each spin selects a single individual. In stochastic universal sampling (SUS), a spin will select as individuals as outers (e.g. 4 individuals in the example).

Tournament selection

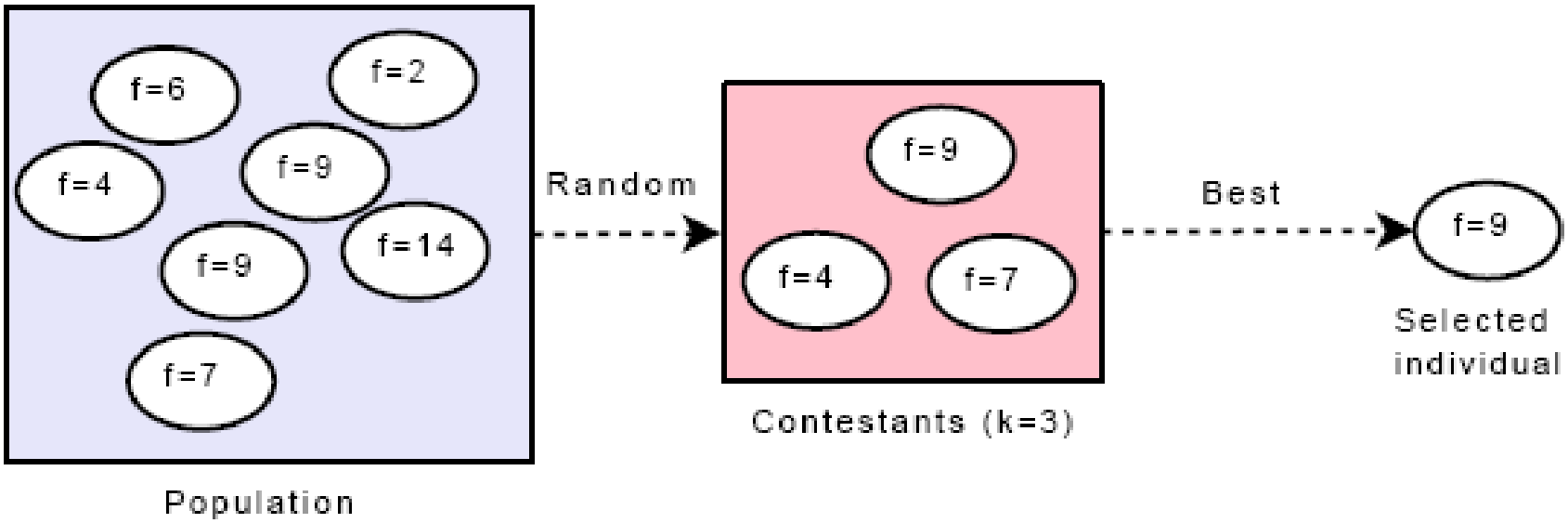


Fig. 3.12 Tournament selection strategy. For instance, a tournament of size 3 is performed. Three solutions are picked randomly from the population. The best solution from the picked individuals is then selected.

The rank based selection

- Individuals are sorted on their fitness value from best to worse. The place in this sorted list is called rank
- Instead of using the fitness value of an individual, the rank is used by a function to select individuals from this sorted list. The function is biased towards individuals with a high rank (i.e. good fitness)

Rank based selection

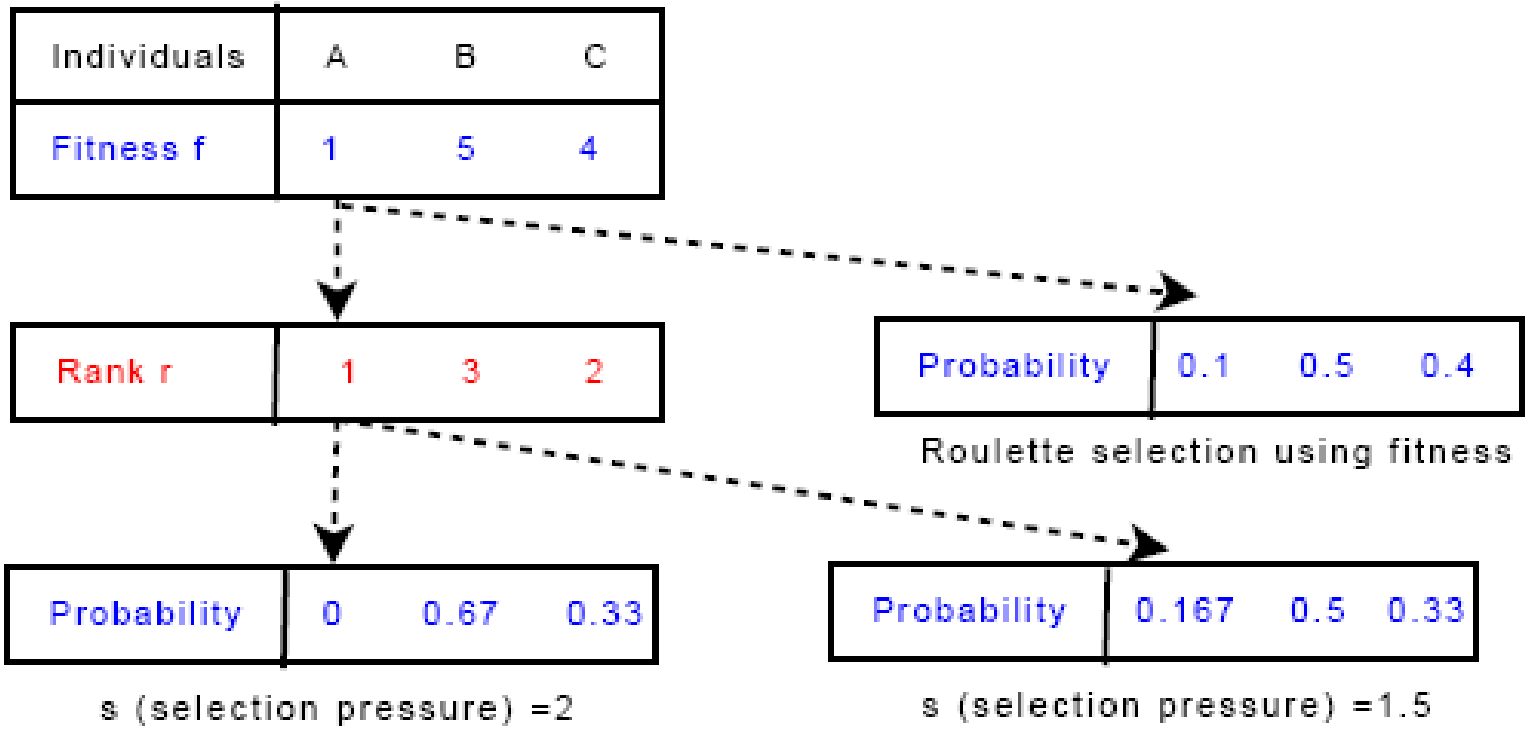


Fig. 3.13 Rank-based selection strategy using a linear ranking.

Replacement strategies

- **Generational replacement:** the replacement will concern the whole population of size μ. The offspring population will replace systematically the parent population. This strategy is applied in the canonical GA as proposed by J. Holland.
- **Steady state replacement:** at each generation of an EA, only one offspring is generated. For instance, it replaces the worst individual of the parent population.

The replacement strategy

- Selection pressure is also affected in the replacement step (survivors of both population and offspring)
- Stochastic methods/deterministic strategies
- Elitism (i.e. should fitness ever improve ?)
  $\rightarrow$ Reintroduce in the new generation the best solution found during the search

Variation operators

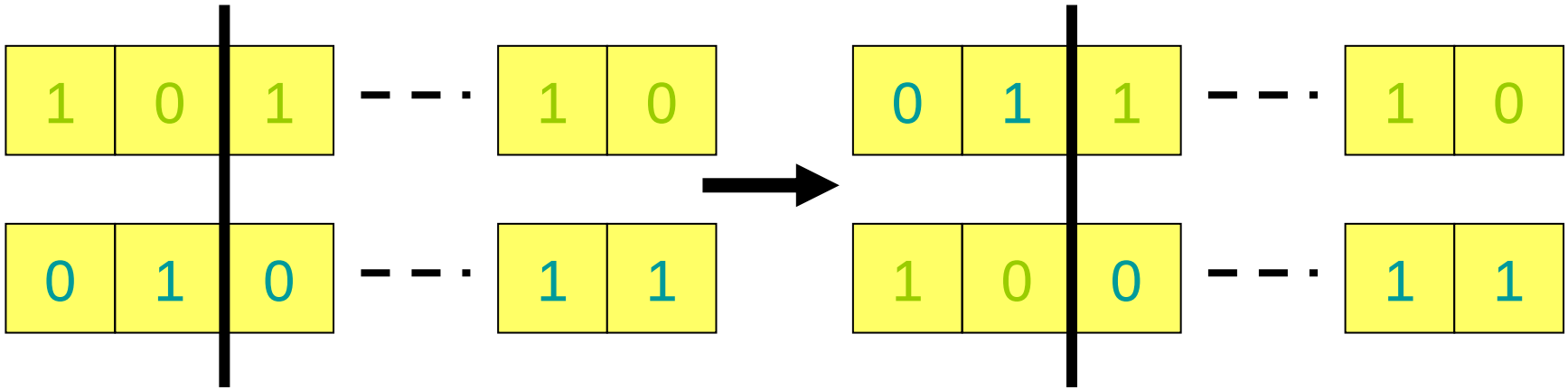
- Crossover operators
  - Heritability: should inherit material from the two parents
  - Valid: provides valid solutions
- Mutation operators
  - Ergodicity: every solution of the search space to be reached
  - Validity: provides valid solutions
  - Locality: minimal change (perturbation) → neighborhood concept in S-metaheuristics

Recombination operators

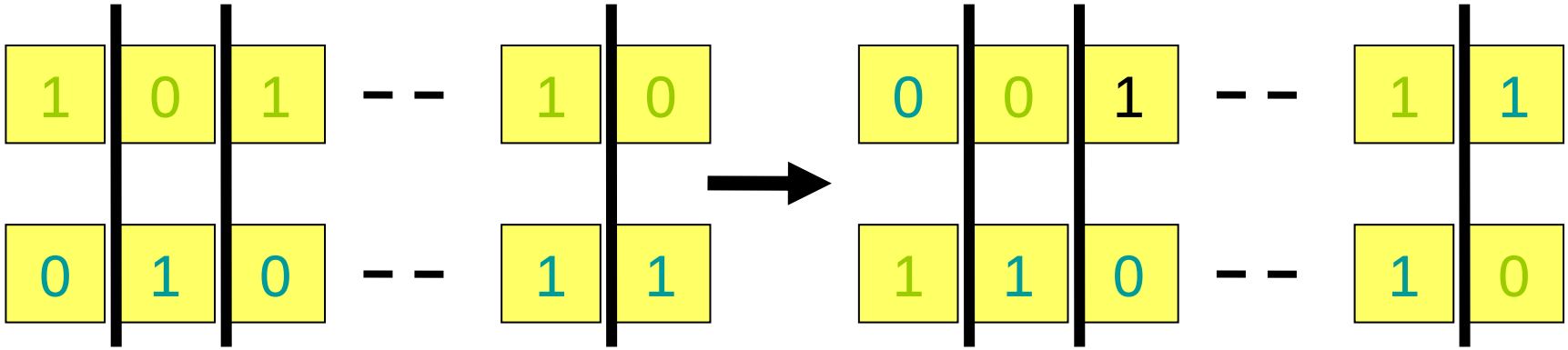
- We might have one or more recombination operators for our representation
- Some important points are
  - The child should inherit something from each parent. If this is not the case then the operator is a copy operator
  - The recombination operator should be designed in conjunction with the representation.
  - Recombination should produce valid chromosomes

Example. Recombination for discrete representation

- N-points crossover (e.g. 1 point)

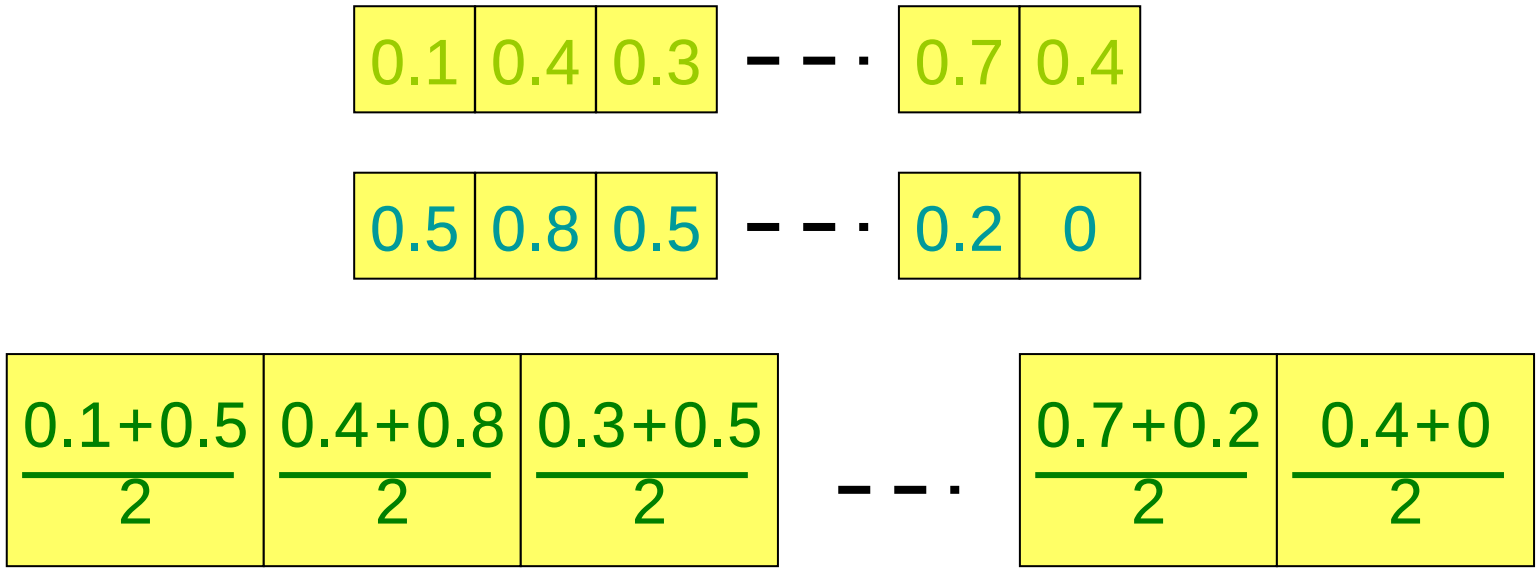


- Uniform crossover



Example. Recombination for real valued representation

- Intermediate binary recombination (arithmetic crossover). Given two parents one child is created as follows



Example. Recombination for real valued representation

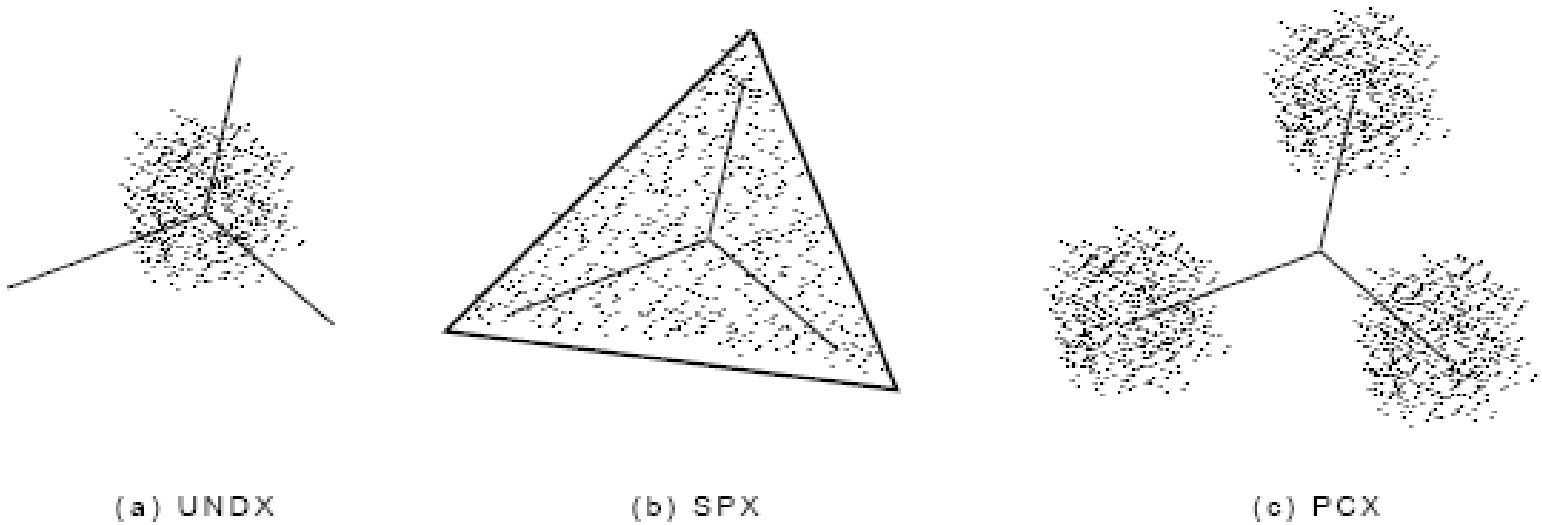
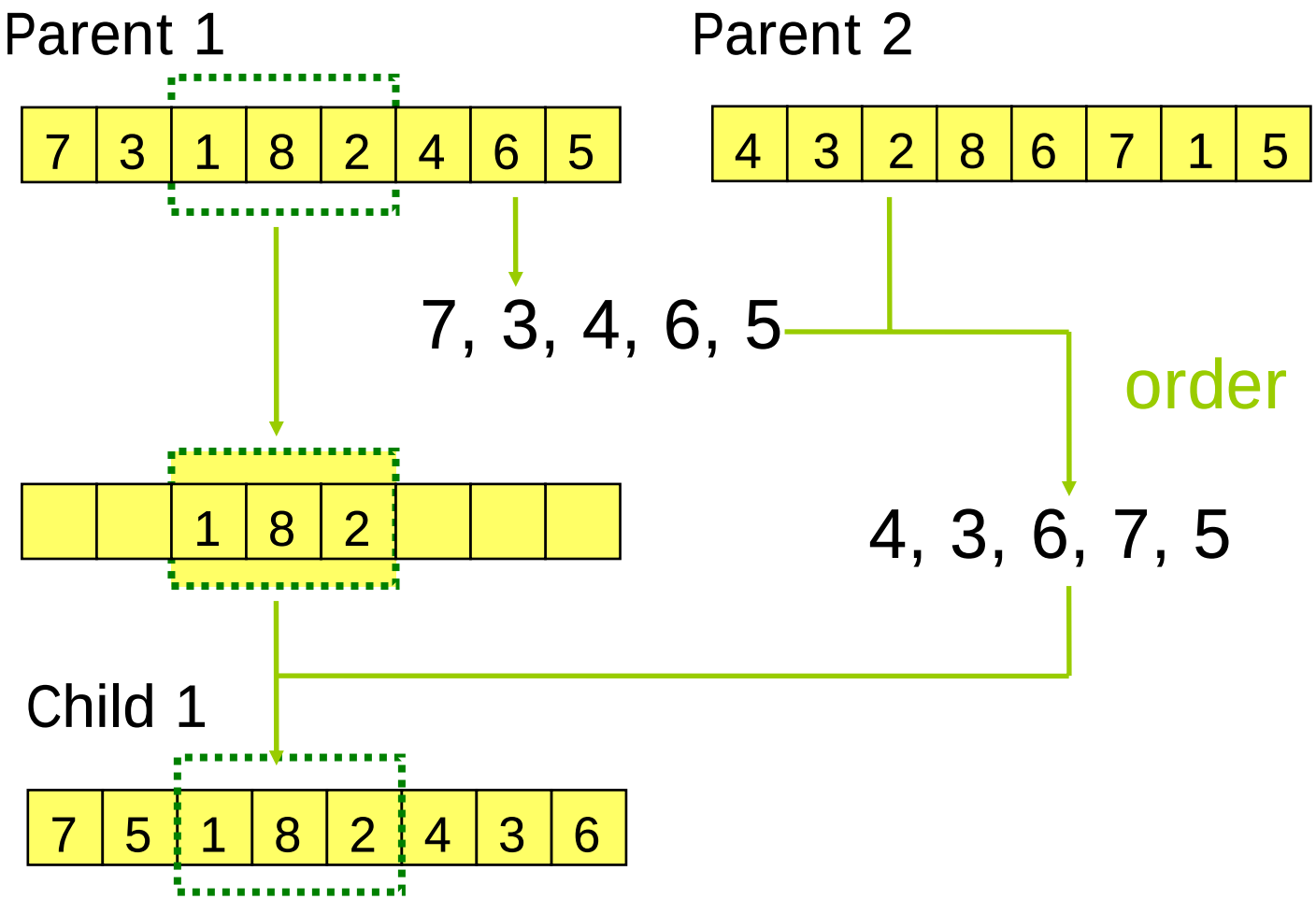


Fig. 3.18 Offsprings distribution using the crossover operators UNDX, SPX and PCX.

Recombination for order based representation

- Choose an arbitrary part from the first parent and copy this to the first child
- Copy the remaining genes that are not in the copied part to the first child
  - starting right from the cut point of the copied part
  - using the order of genes from the second parent
  - wrapping around at the end of the chromosome
- Repeat this process with the parent roles reversed

Example. Recombination for order based representation (Order 1)



Example. Recombination for order based representation (PMX)

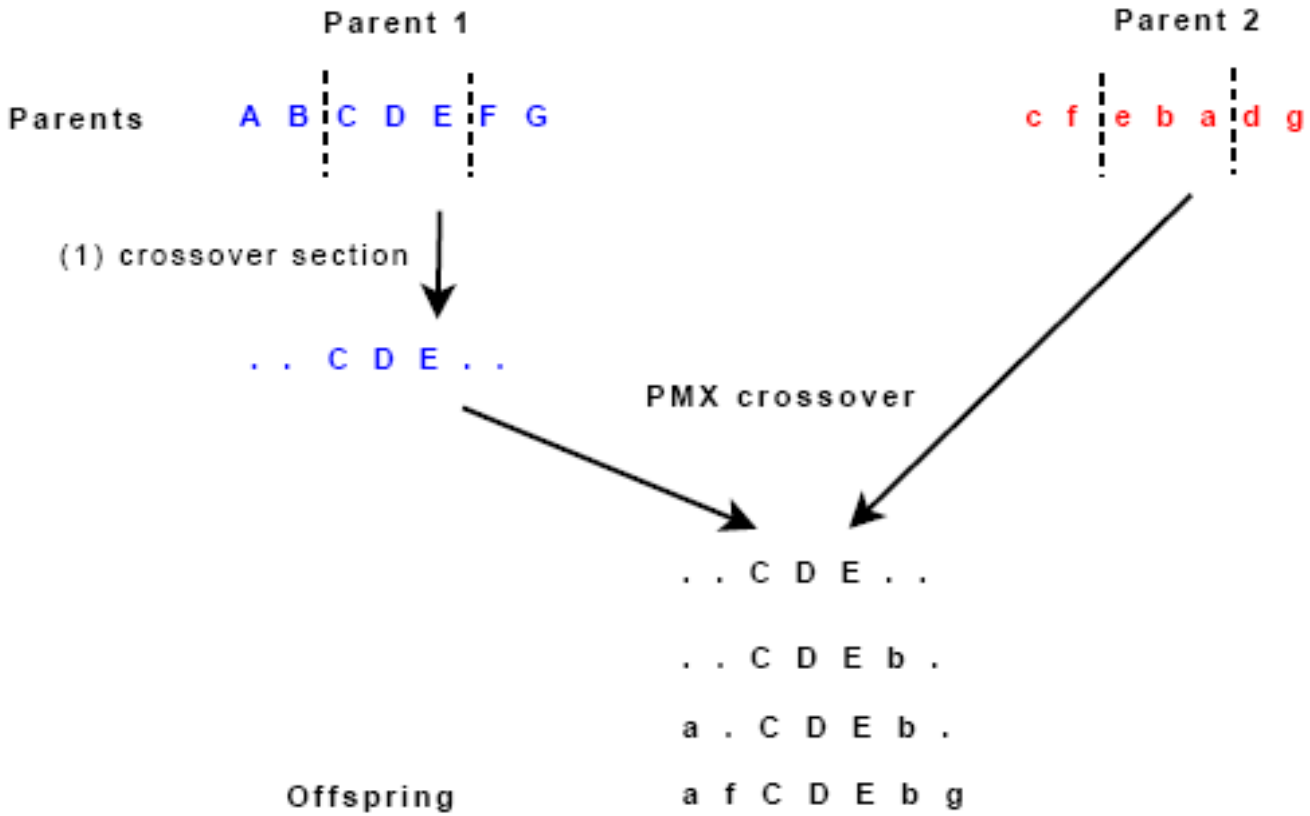
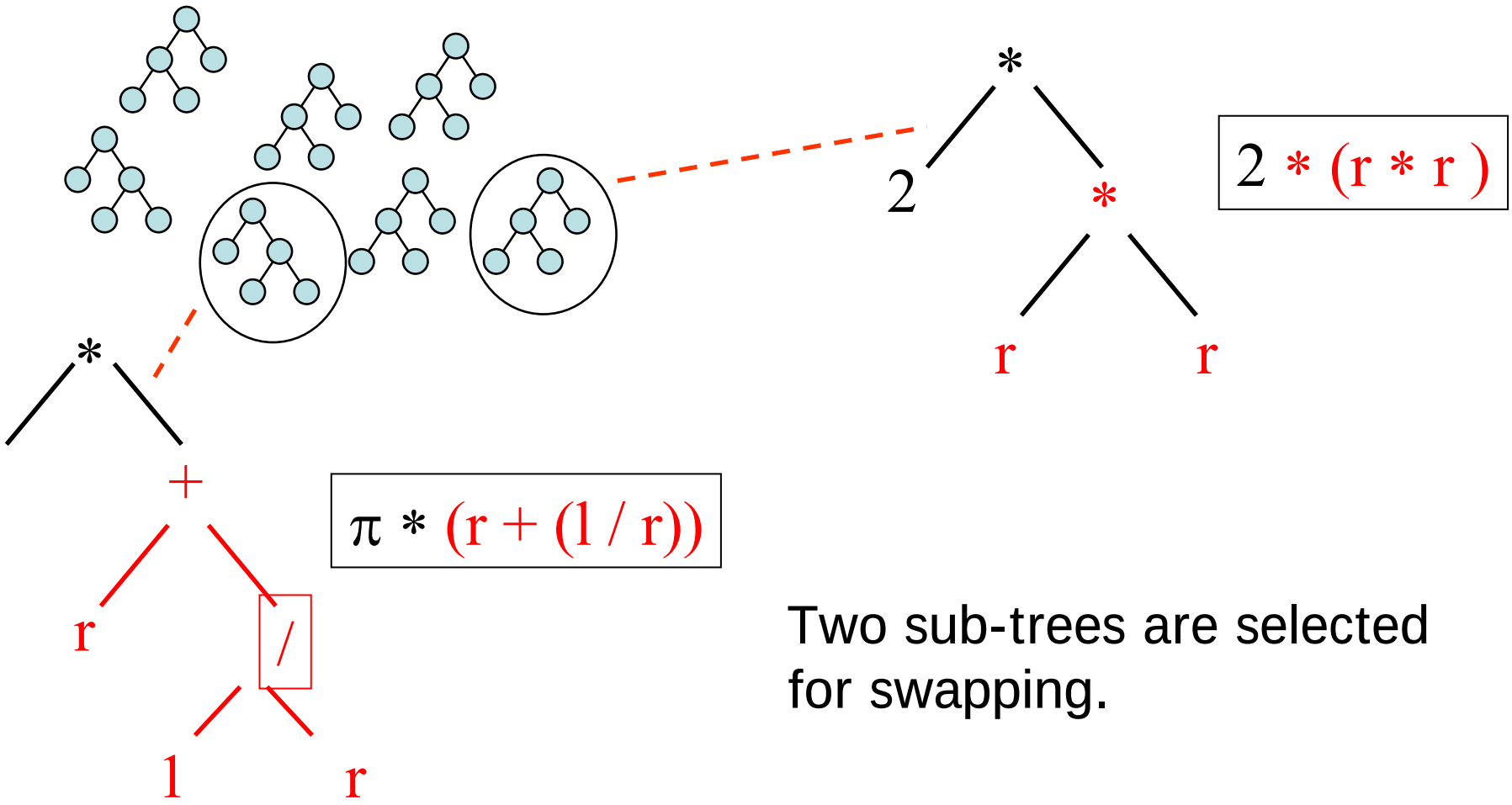


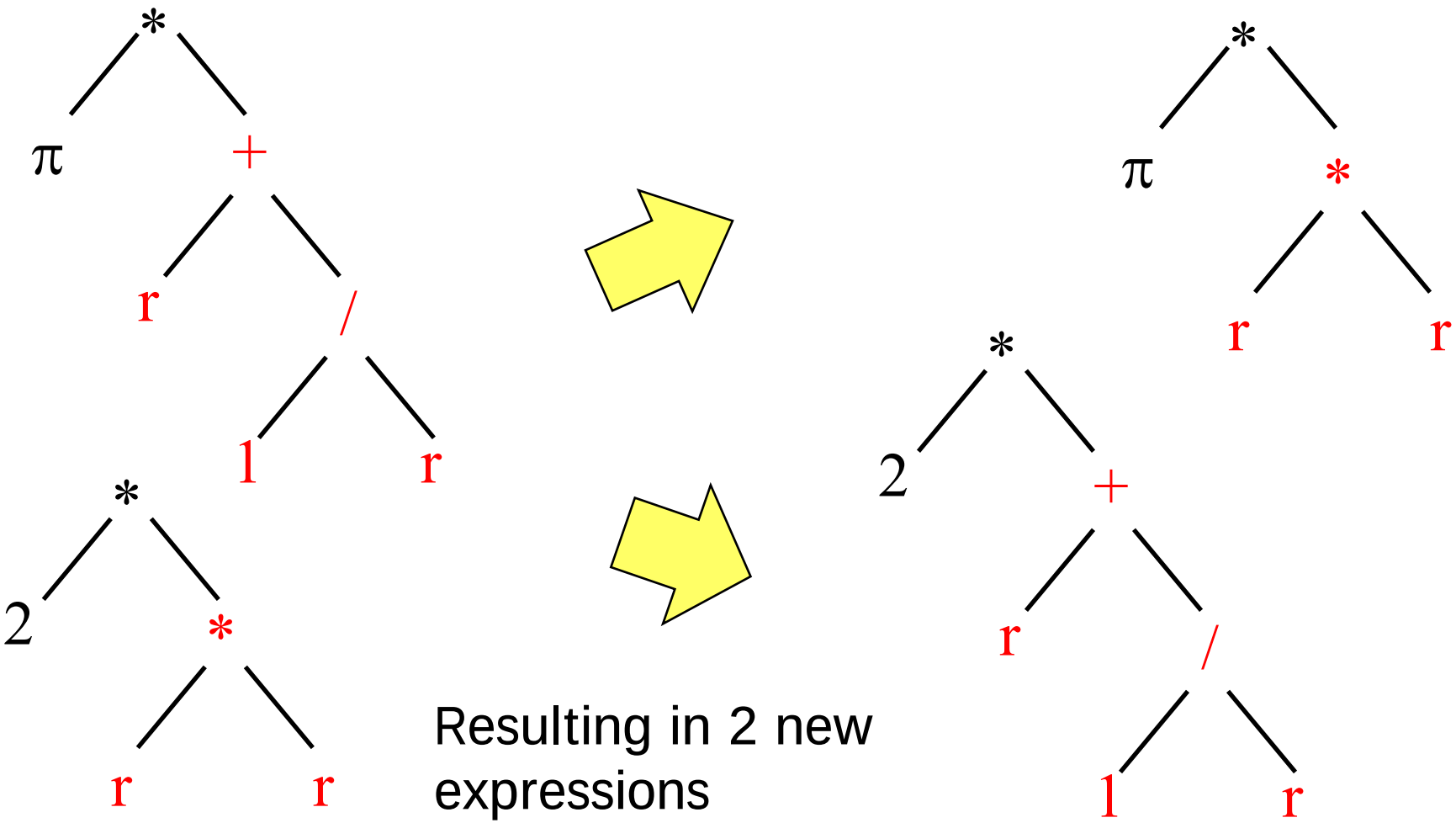
Fig. 3.20 The partially mapped crossover (PMX) for permutations.

Recombination for tree based representation



Two sub-trees are selected for swapping.

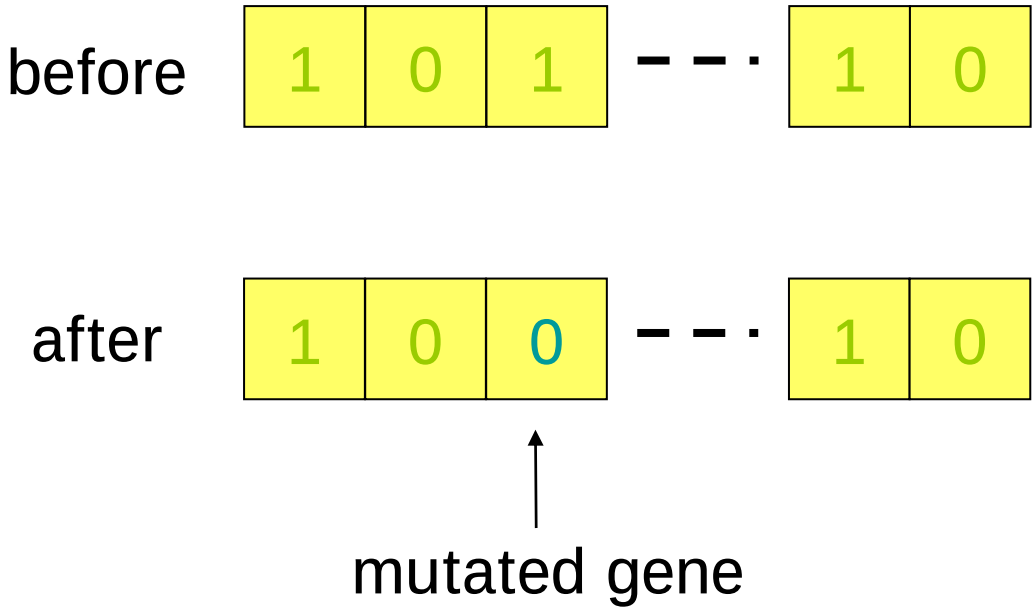
Recombination for tree based representation



Mutation operators

- We might have one or more mutation operators for our representation
- Some important points are
  - At least one mutation operator should allow every part of the search space to be reached
  - The size of mutation is important and should be controllable
  - Mutation should produce valid chromosomes

Example. Mutation for discrete representation

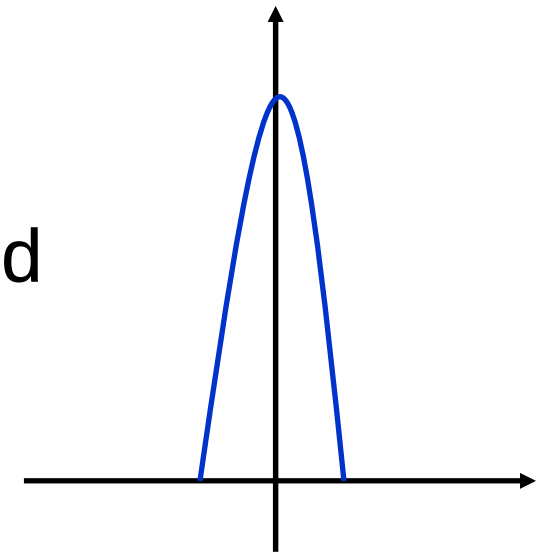


- Mutation usually happens with probability p_m for each gene
- It could affect only one gene too

Example. Mutation for real valued representation

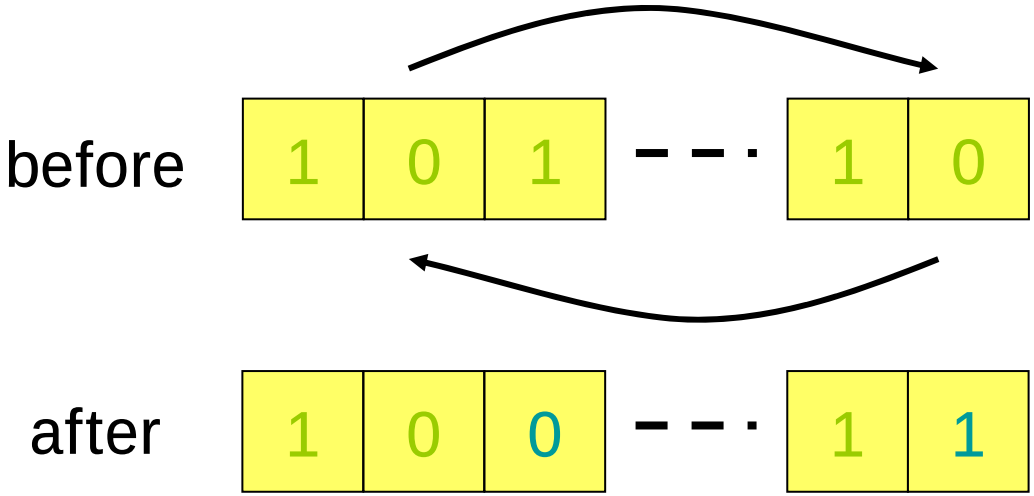
- Perturb values by adding some random noise
- Often, a Gaussian/normal distribution $N(0,\sigma)$ is used, where
  - 0 is the mean value
  - σ is the standard deviation and

$$x'i = xi + N(0,\sigma i)$$



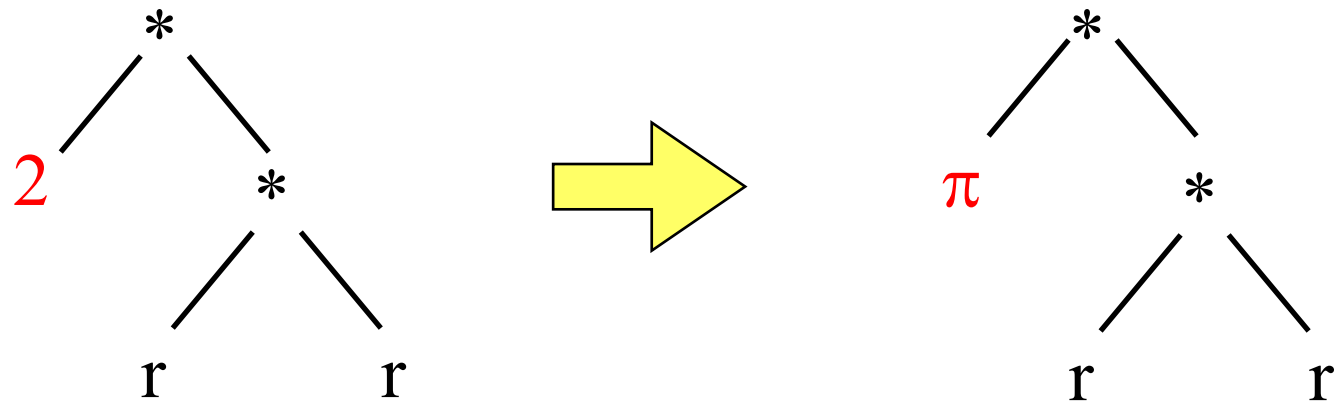
Example. Mutation for order based representation

- Randomly select two different genes and swap them



Example. Mutation for tree-based representation

- Single point mutation selects one node and replaces it with a similar one



Swarm Intelligence

Swarm Intelligence

- Collective system capable of accomplishing difficult tasks in dynamic and varied environments without any external guidance or control and with no central coordination

- Achieving a collective performance which could not normally be achieved by an individual acting alone

- Constituting a natural model particularly suited to distributed problem solving

Inherent features

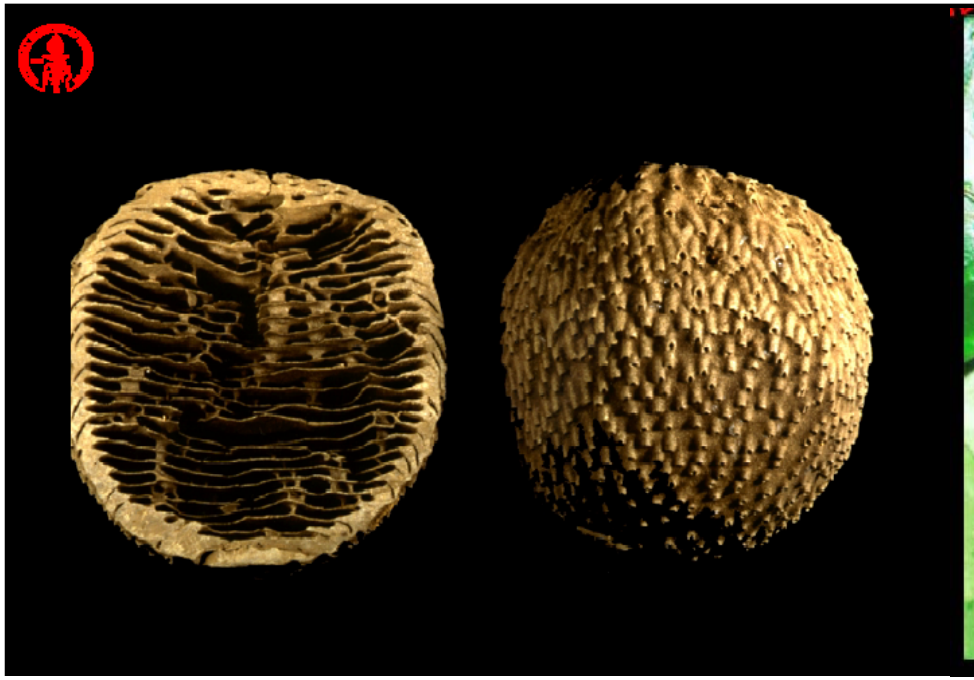
- Inherent parallelism
- Stochastic nature
- Adaptivity
- Use of positive feedback (reinforcement learning)
- Autocatalytic in nature

Swarm intelligence

Bird flocking

Fish

Swarm intelligence



Bees

Ants

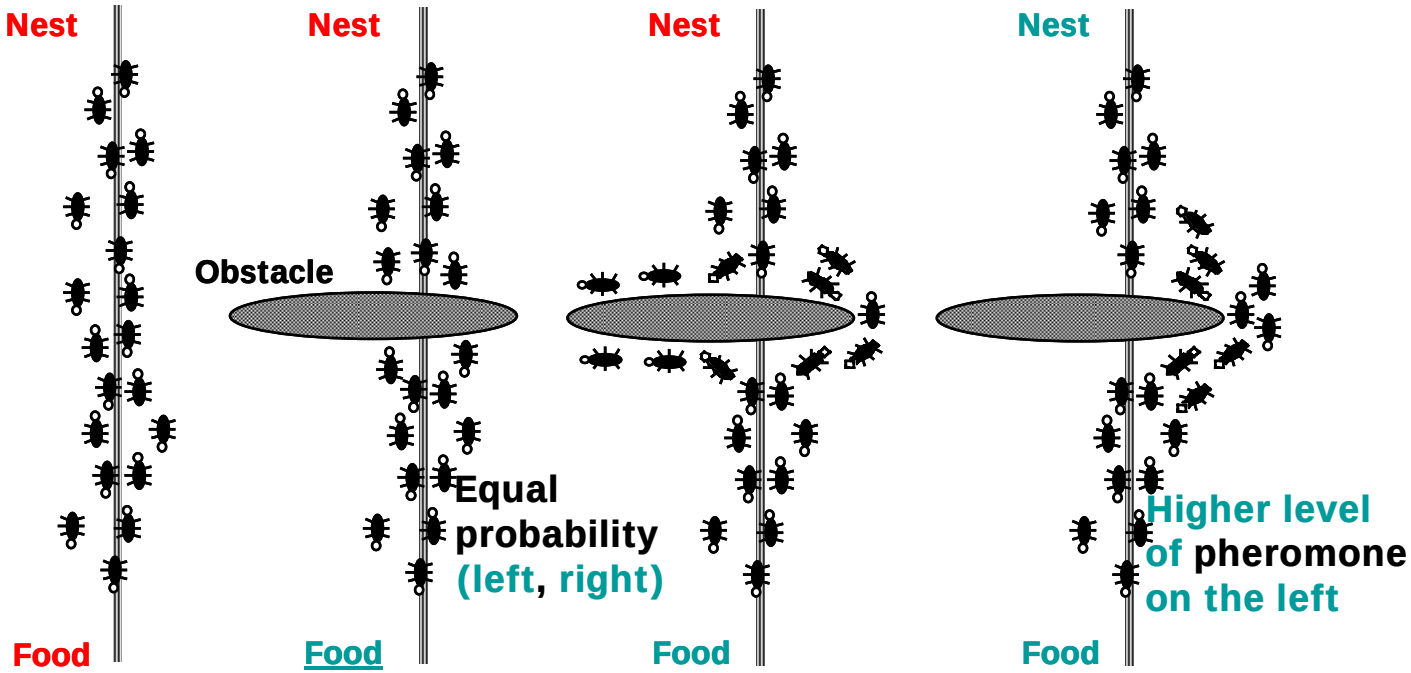
Ant colony Optimization (ACO)

Ant colonies

- Artificial ants: Dorigo (1992)
- Imitate the cooperative behavior of ant colonies to solve optimization problems
- Use very simple communication mechanism : pheromone
  - Olfactive and volatile substance
  - Evolution : evaporation, reinforcement

M. Dorigo, & G. Di Caro. “The Ant Colony Optimization Meta-heuristic”. *New Ideas in Optimization*, 1999

A nature-inspired process



- During the trip, a pheromone is left on the ground.
- The quantity left depends on the amount of food found.
- The path is chosen accordingly to the quantity of pheromones.
- The pheromone has a decreasing action over time.

Template of Ant colony optimization

Algorithm 3.12 Template of the ant colony algorithm (ACO).

Initialize the pheromone trails ;

Repeat

  For each ant **Do**

    Solution construction using the pheromone trail ;

    Update the pheromone trails:

      Evaporation ;

      Reinforcement ;

Until Stopping criteria

Output: Best solution found or a set of solutions.

General ACO

- A stochastic construction procedure
- Probabilistically build a solution
- Iteratively adding solution components to partial solutions
    - Heuristic information
    - Pheromone trail
- Reinforcement Learning reminiscence
- Modify the problem representation at each iteration

Pheromone based Construction

- At the beginning of the search process, a constant amount of pheromone is assigned to all arcs. When located at a node *i* an ant *k* uses the pheromone trail to compute the probability of choosing *j* as the next node:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}} & if \quad j \in N_i^k \\ 0 & if \quad j \notin N_i^k \end{cases}$$

- where N_i^k is the neighborhood of ant *k* when in node *i.*

Reinforcement of the pheromone

- When the arc (i,j) is traversed , the pheromone value changes as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^{k}$$

- By using this rule, the probability increases that forthcoming ants will use this arc.

Evaporation

- After each ant *k* has moved to the next node, the pheromones evaporate by the following equation to all the arcs:

$$\tau_{ij} \leftarrow (1-p)\tau_{ij}, \ \forall (i,j) \in A$$

- where $p \in (0,1]$ is a parameter. An iteration is a completer cycle involving ants' movement, pheromone evaporation, and pheromone deposit.

ACO Design

1. Represent the problem in the form of sets of components and transitions, or by a set of weighted graphs, on which ants can build solutions
2. Define the meaning of the pheromone trails
3. Define the heuristic preference for the ant while constructing a solution
4. If possible implement a efficient local search algorithm for the problem to be solved.
5. Choose a specific ACO algorithm and apply to problem being solved
6. Tune the parameter of the ACO algorithm.

How to implement in a program

- Ants: Simple computer agents
- Move ant: Pick next component in the const. solution
- Pheromone: $\Delta\tau_{i,j}^{k}$
- Memory: M_K or $Tabu_K$
- Next move: Use probability to move ant

A simple TSP example

A

B

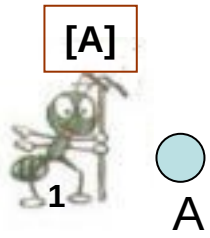
C

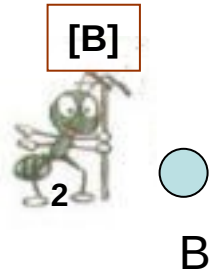
D

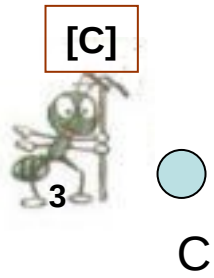
E

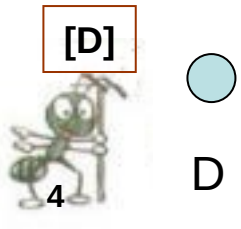
$d_{AB}=100; d_{BC}=60\ldots; d_{DE}=150$

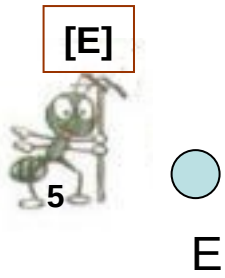
Iteration 1



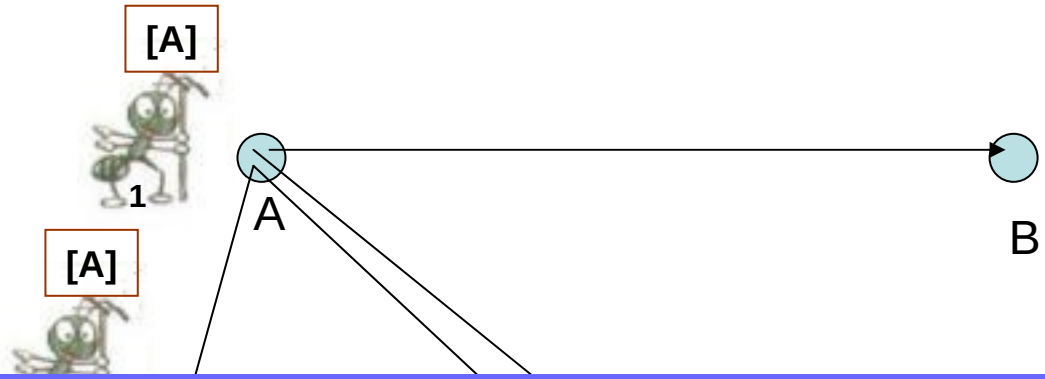








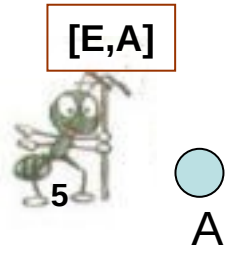
How to build next sub-solution?

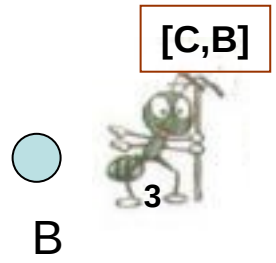


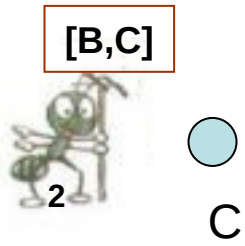
$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha}\,[\eta_{ik}]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

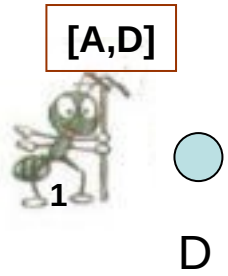
E

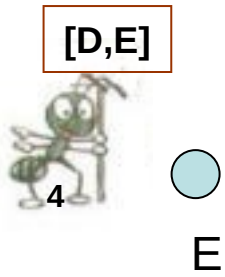
Iteration 2



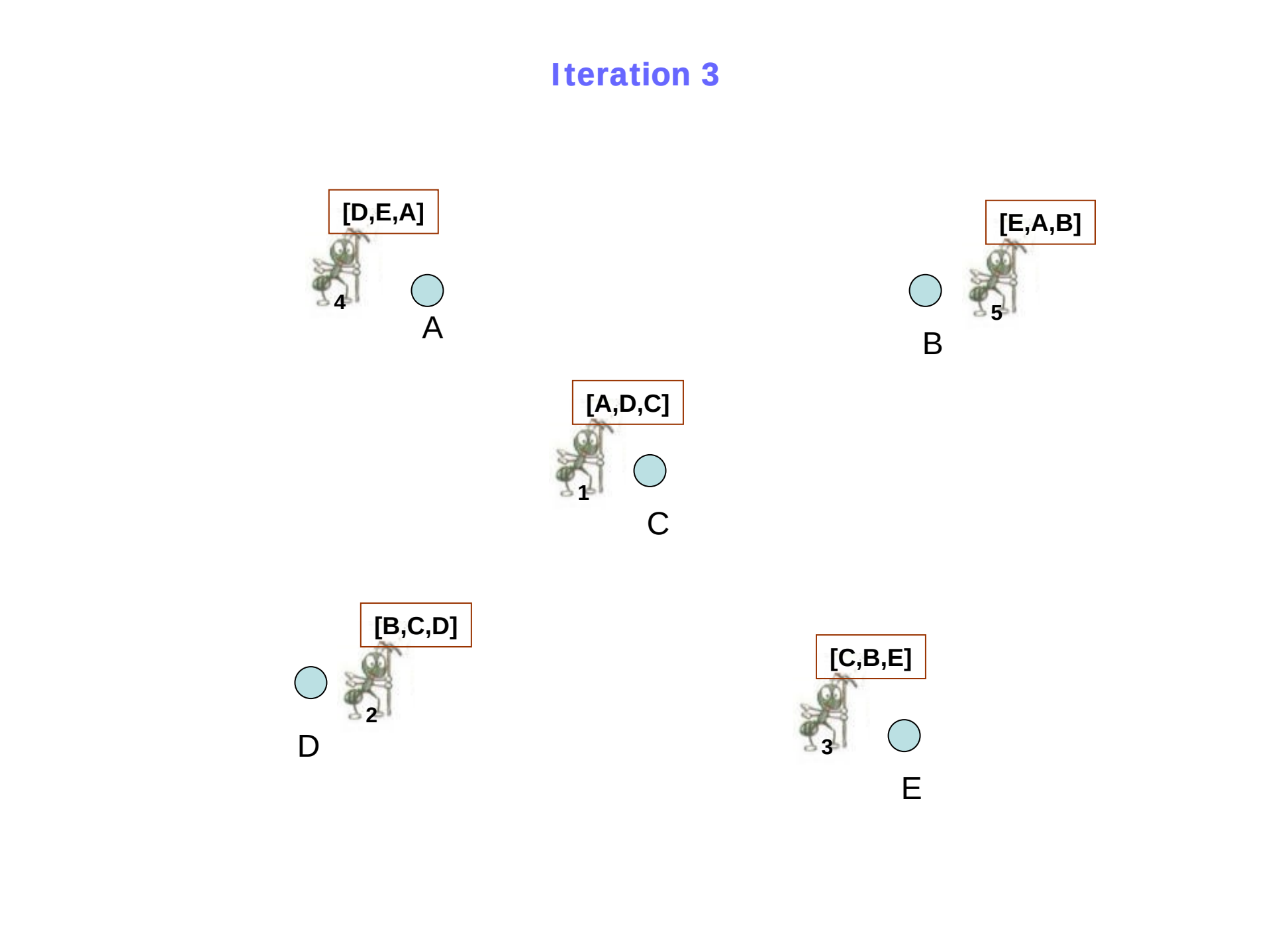




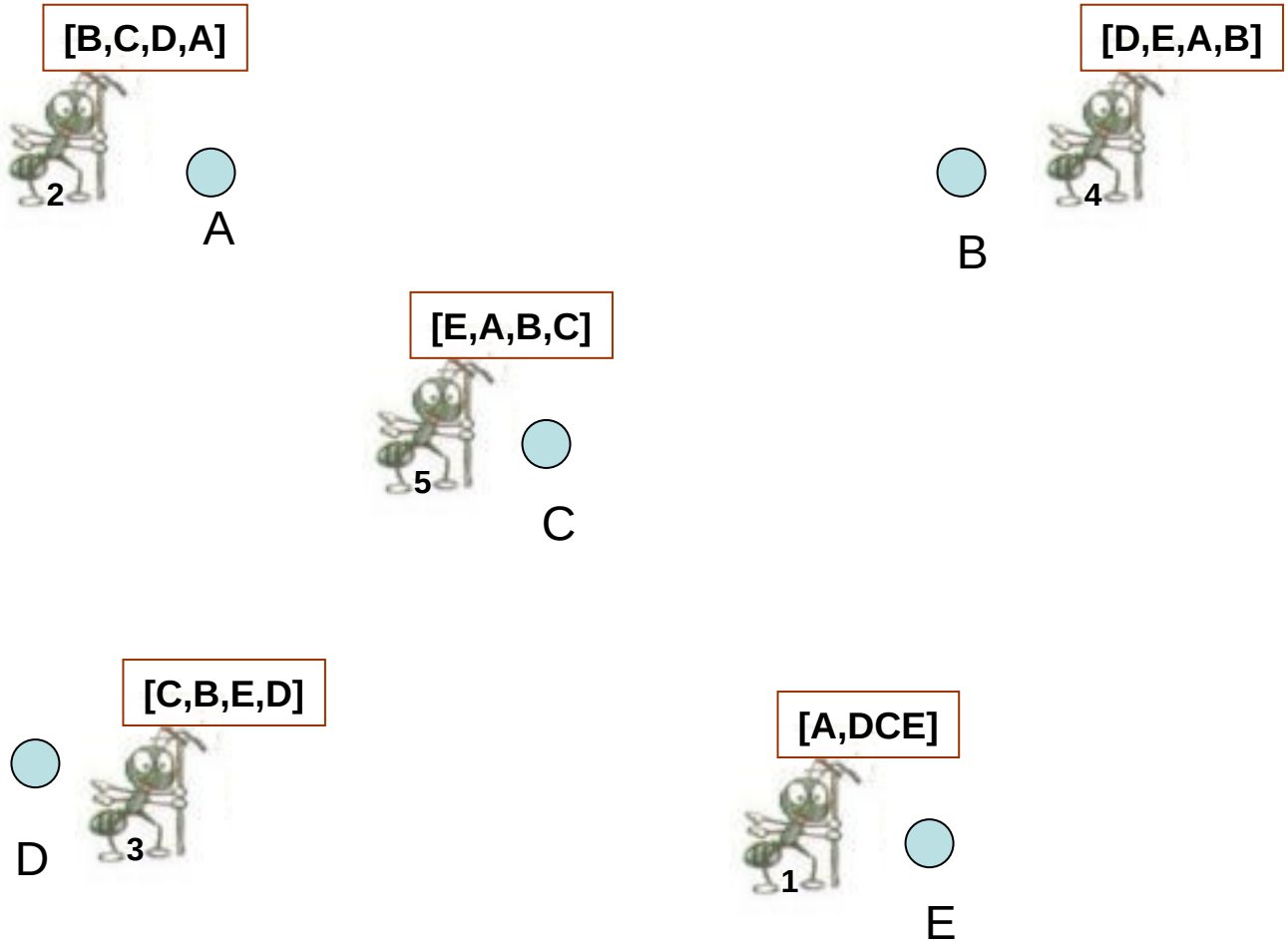




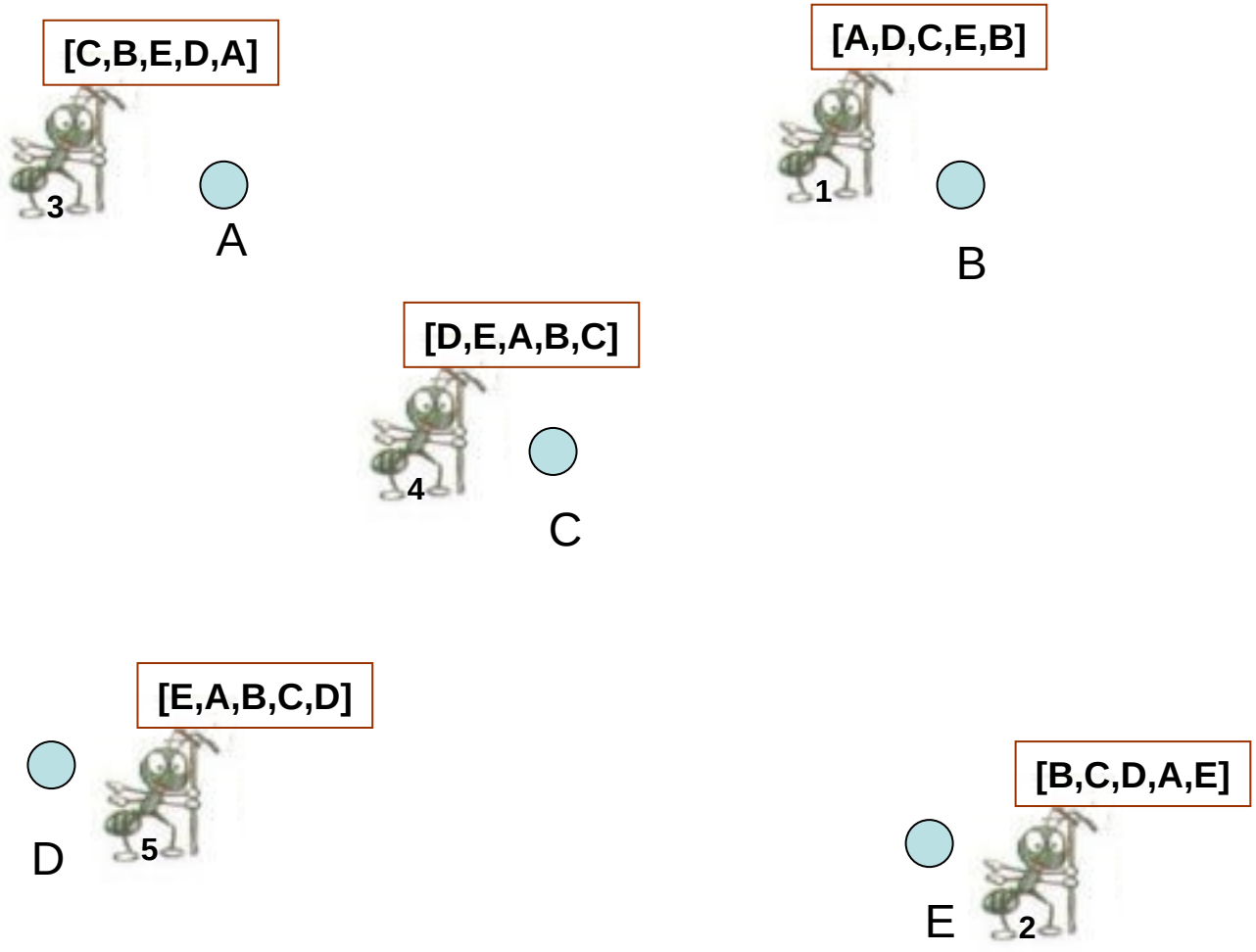
Iteration 3



Iteration 4



Iteration 5



Path and Pheromone Evaluation

[A,D,C,E,B]

$L_1 = 300$

$$\Delta\tau_{i,j}^{k} = \begin{cases} \dfrac{Q}{L_k} & if\ (i,j) \in \text{tour} \\ 0 & \text{otherwise} \end{cases}$$

[B,C,D,A,E]

$L_2 = 450$

[C,B,E,D,A]

$$\Delta\tau_{A,B}^{total} = \Delta\tau_{A,B}^{1} + \Delta\tau_{A,B}^{2} + \Delta\tau_{A,B}^{3} + \Delta\tau_{A,B}^{4} + \Delta\tau_{A,B}^{5}$$

[D,E,A,B,C]

$L_4 = 280$

[E,A,B,C,D]

$L_5 = 420$

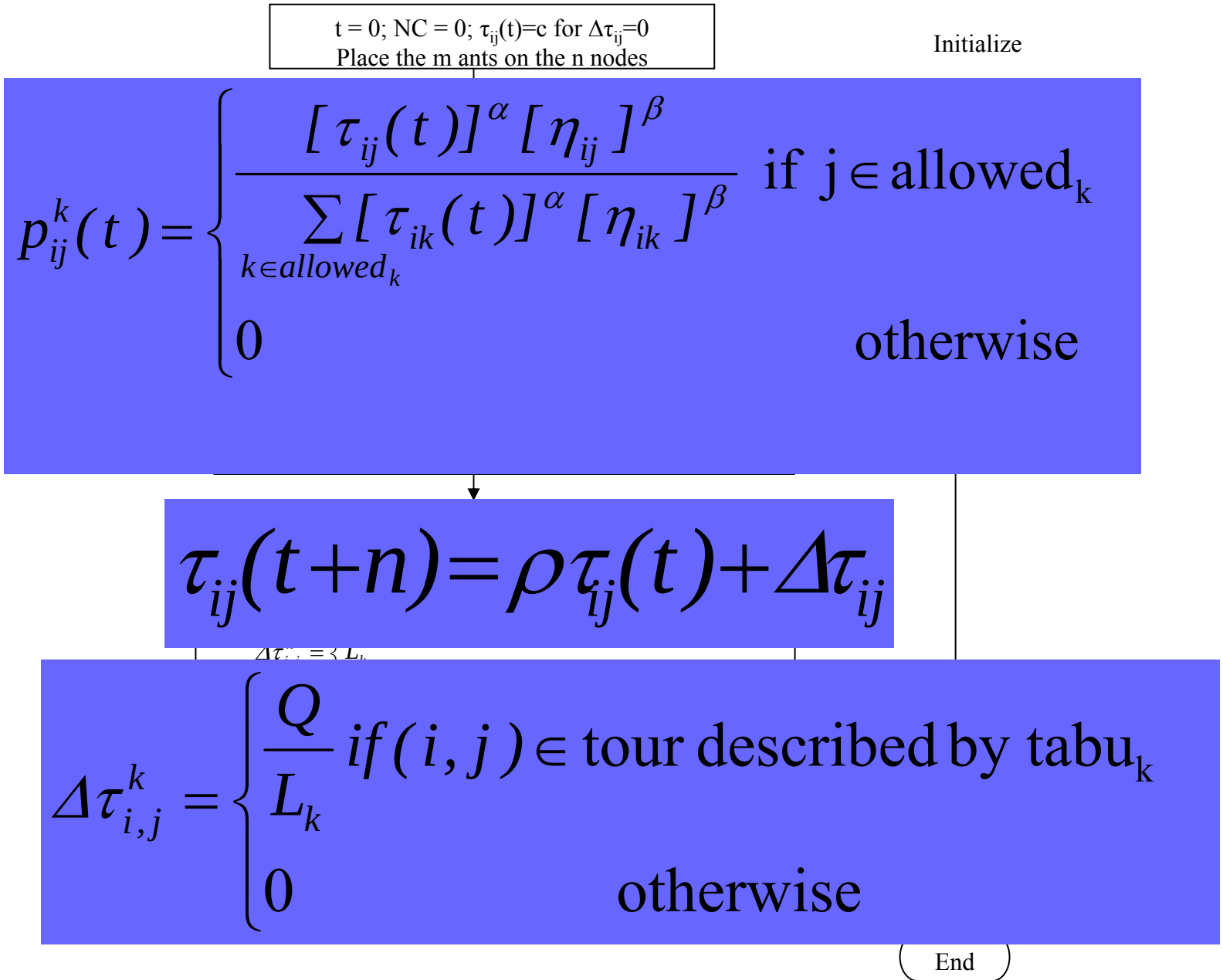
End of First Run

Save Best Tour (Sequence and length)

All ants die

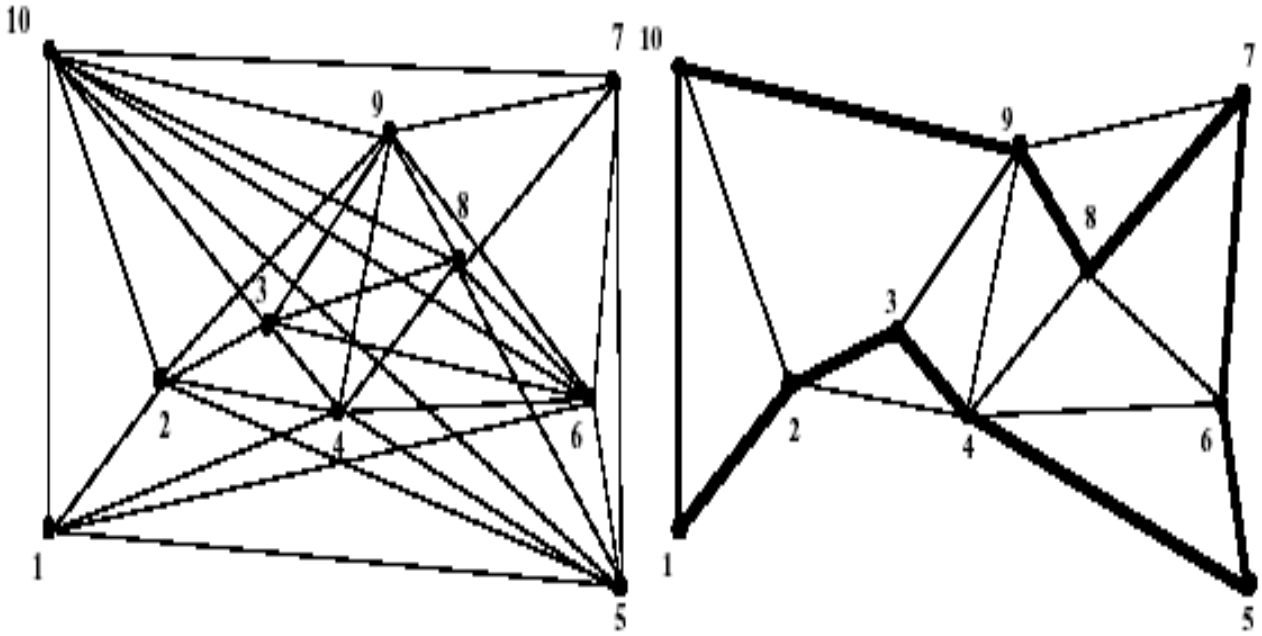
New ants are born

Ant System (Ant Cycle) Dorigo [1] 1991



Stopping Criteria

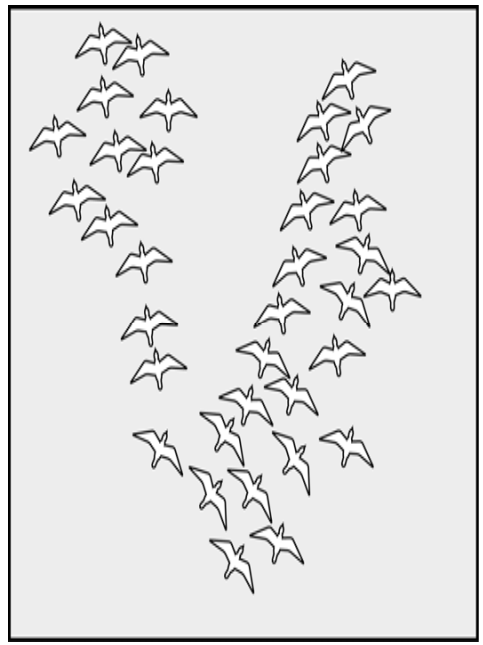
- Stagnation
- Max Iterations



Particle Swarm Optimization (PSO)

Particle Swarm

- Population based stochastic metaheuristic
- Dr. Eberhart and Dr. Kennedy (1995)
- Inspired by social behavior of bird flocking or fish schooling
- Similarities with genetic algorithm



Kennedy, J. and Eberhart, R. C., "The particle swarm: social adaptation in information processing systems.," in Corne, D., Dorigo, M., and Glover, F. (eds.) New Ideas in Optimization London, UK: McGraw-Hill, 1999

Particle Swarm

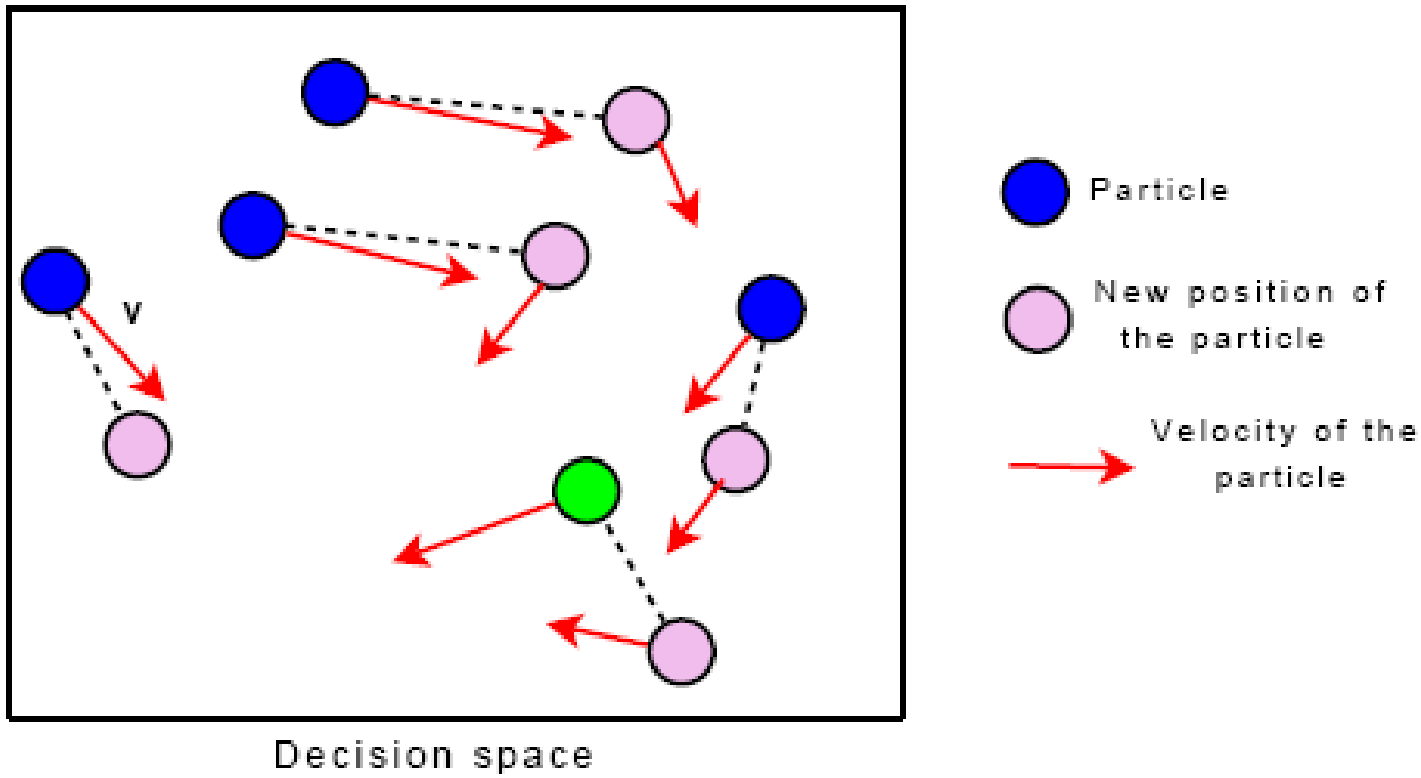
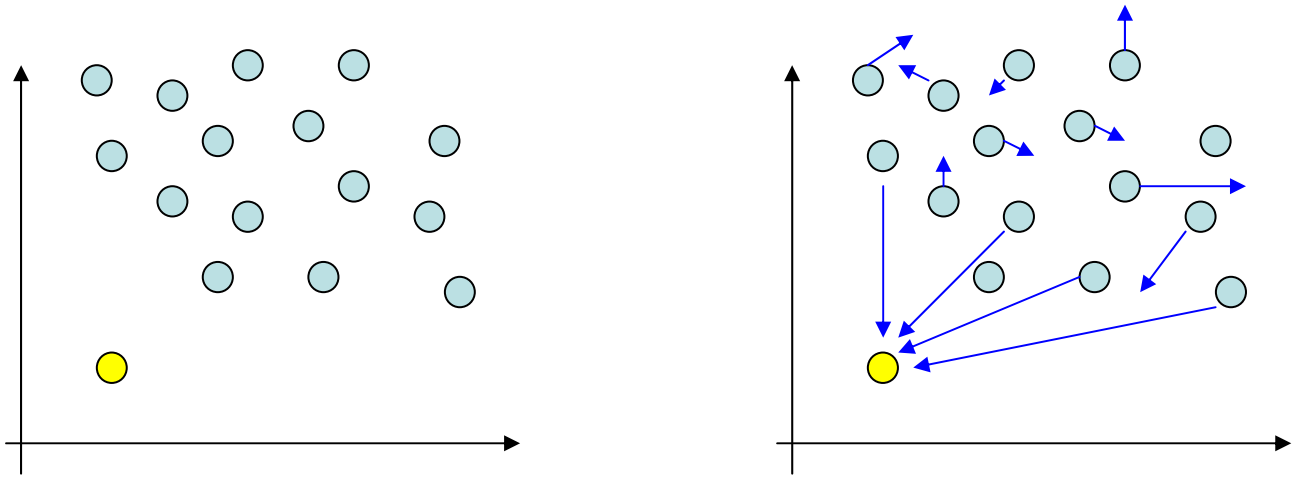


Fig. 3.33 Particle swarm with their associated positions and velocities. At each iteration, a particle moves from one position to another one in the decision space. PSO uses no gradient information during the search.

A nature-inspired process



- Particles fly through the problem space
- Flight = add a velocity to the current position
- Social adaptation of knowledge
- Particles follow the current optimum particles («follow the bird which is nearest to the food »)

Template of the PSO algorithm

Algorithm 3.14 Template of the particle swarm optimization (PSO) algorithm.

Random initialization of the whole swarm ;

Repeat

Evaluate $f(x_i)$;

For all particles i

Update velocities:

$$v_i(t) = v_i(t-1) + \rho_1 \times (p_i - x_i(t-1)) + \rho_2 \times (p_g - x_i(t-1)) \ ;$$

Move to the new position: $x_i(t) = x_i(t-1) + v_i(t)$;

If $f(x_i) < f(pbest_i)$ **Then** $pbest_i = x_i$;

If $f(x_i) < f(gbest)$ **Then** $gbest = x_i$;

Update(x_i, v_i) ;

EndFor

Until Stopping criteria

Swarm construction

- Initialize positions P:

  P_i = random

- Initialize the first best position P_{best} of each particle :

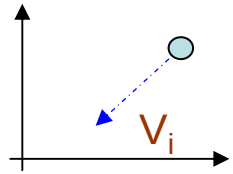
  $P_{i\ best} = P_i$ (standard strategy)

- Initialize the global best Pgbest particle:

  P_{gbest} = best (P_i) (standard strategy)

Make the particles flying

- Evaluate the velocities :

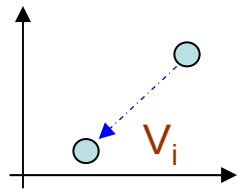


$$V_i = V_i + \underbrace{c_1 \times (P_{i\,best} - P_i)}_{\text{local direction}} + \underbrace{c_2 \times (P_{gbest} - P_i)}_{\text{global direction}}$$

→ *c1 and c2 = learning factors*

- Perform the flight

$$P_i = P_i + V_i$$



Make the particles flying

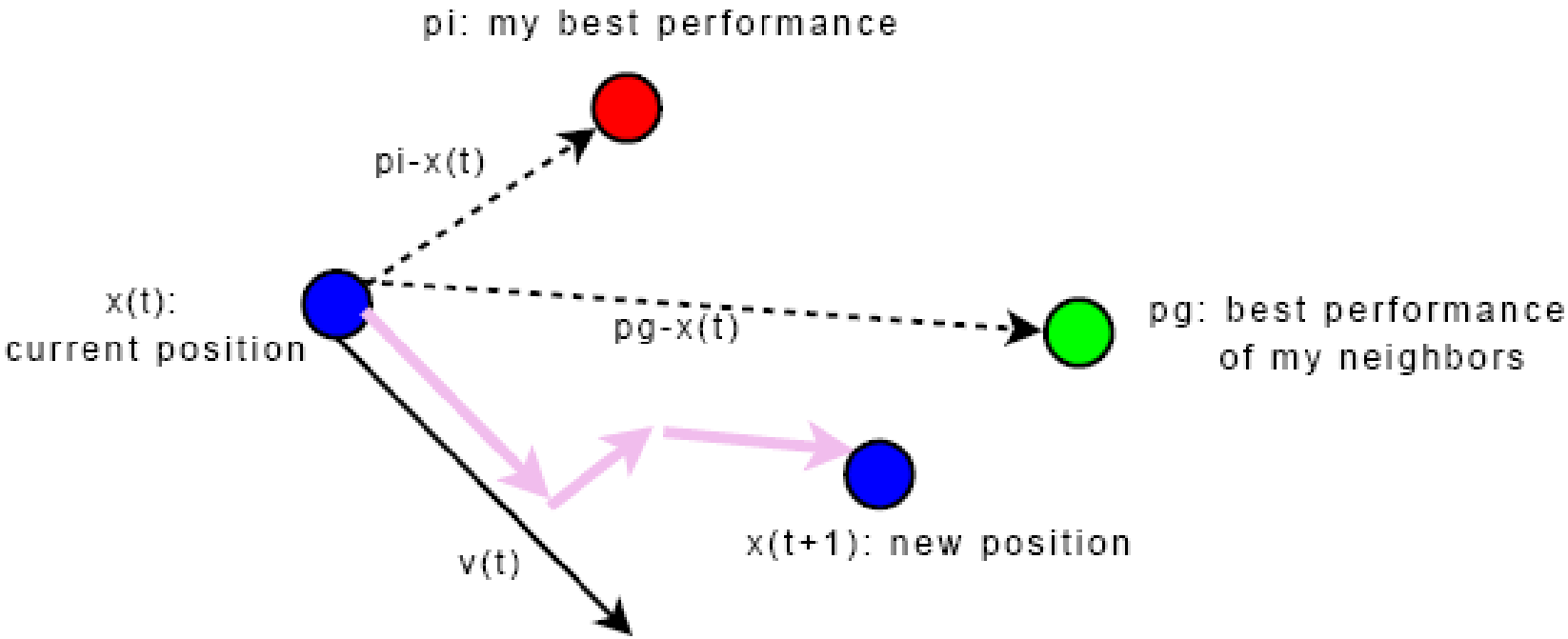
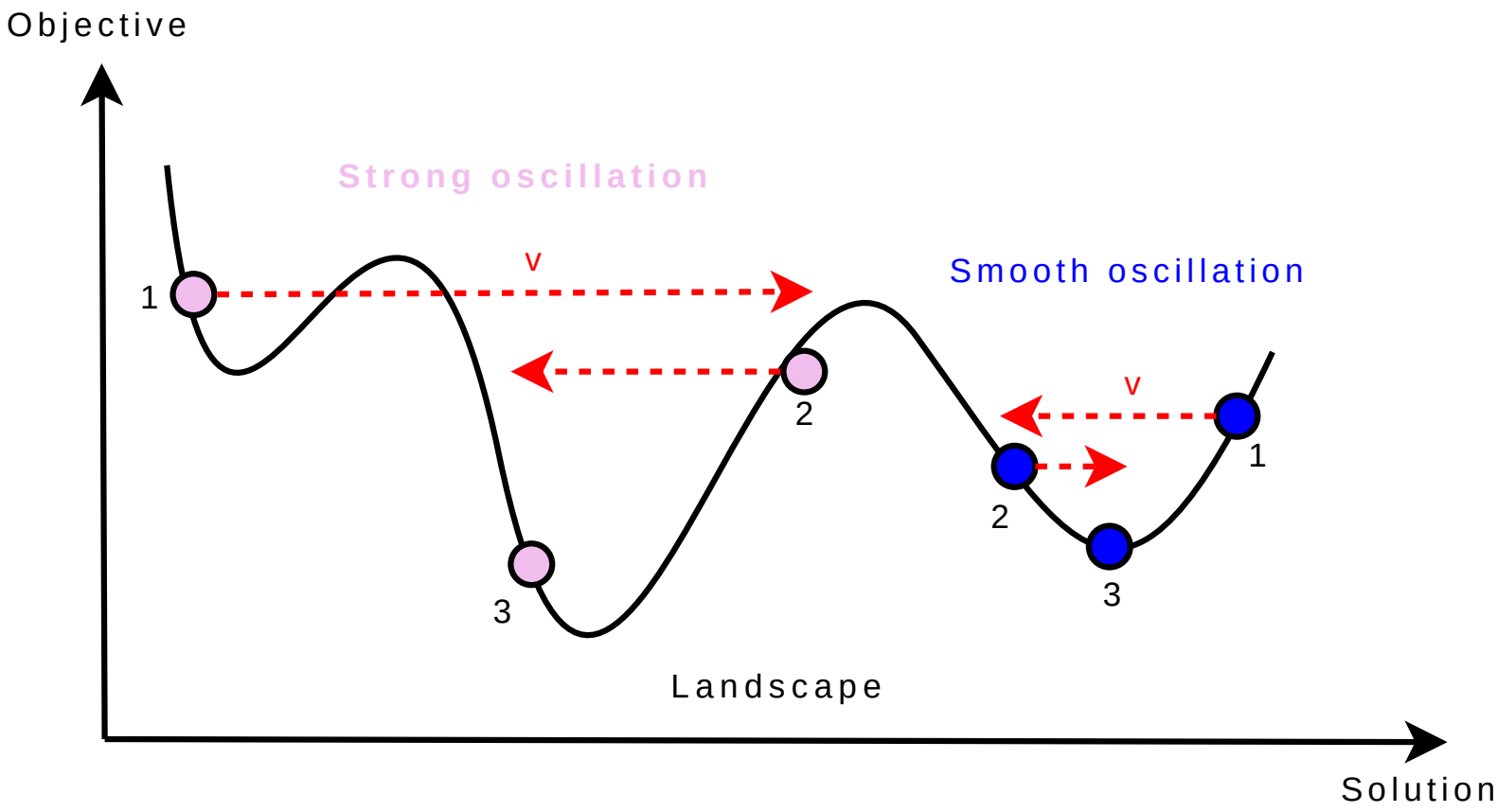


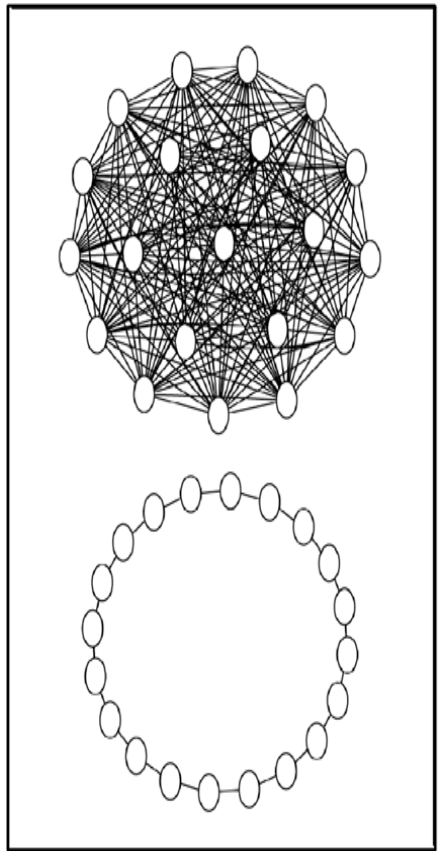
Fig. 3.35 Movement of a particle and the velocity update.

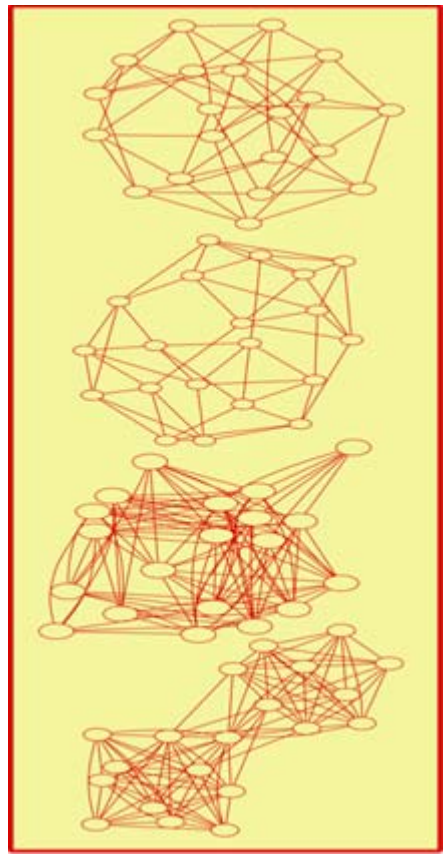
Particle oscillation



Topology (neighborhoods)

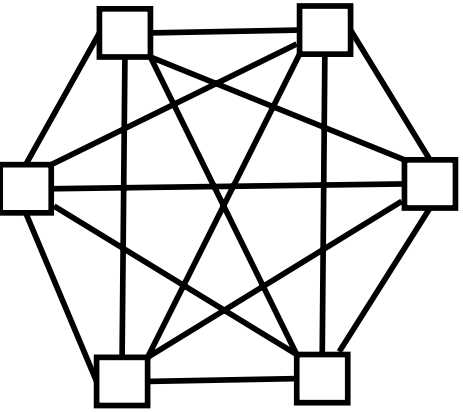
- Determines how the solution spread through the population
- Local, global, neighbourhood best ?
- Affects the rate of convergence
- Advanced parallel search



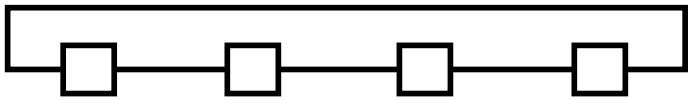


Mean degree, Clustering, Heterogeneity

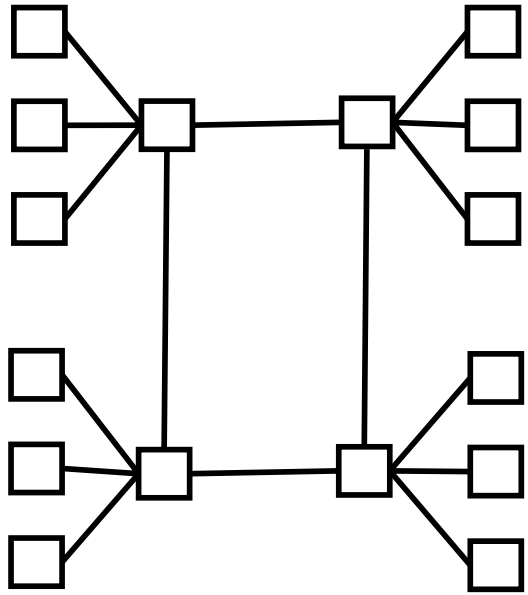
Topology (neighborhoods)



(a) Complete graph



(b) Local structure: a ring



(c) "Small world graph"

Update the particle's best

- Update the best fitness value of each particle :
  - ***If P_i better than $P_{i\ best}$***
    $$P_{i\ best} = P_i$$
- Update the global best :
  - ***If P_i better than P_{gbest}***
    $$P_{gbest} = P_i$$

Discrete PSO

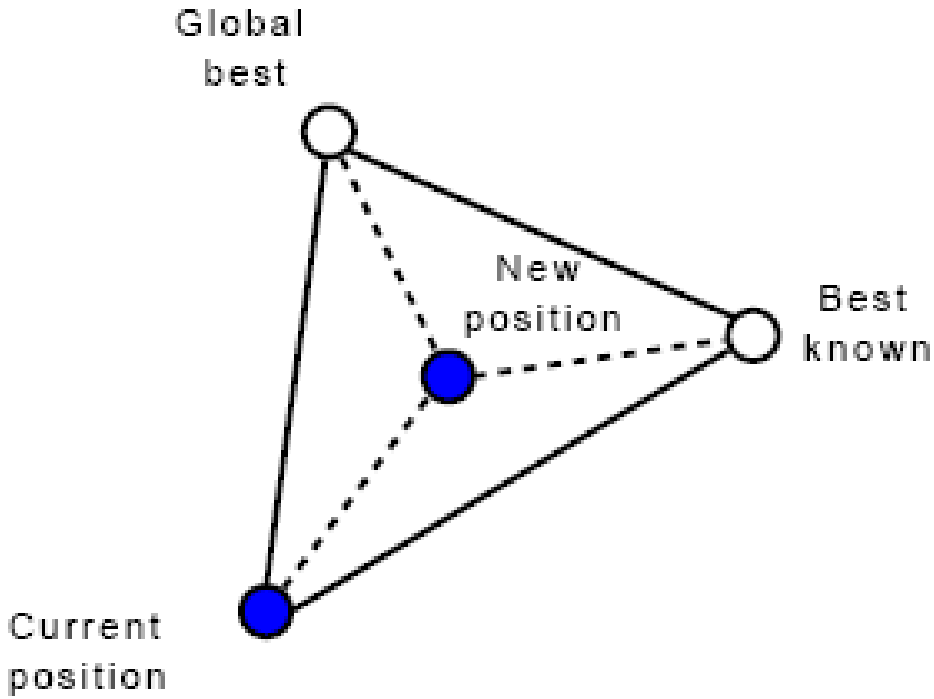


Fig. 3.36 Geometric crossover in the GPSO algorithm.

Oscillation

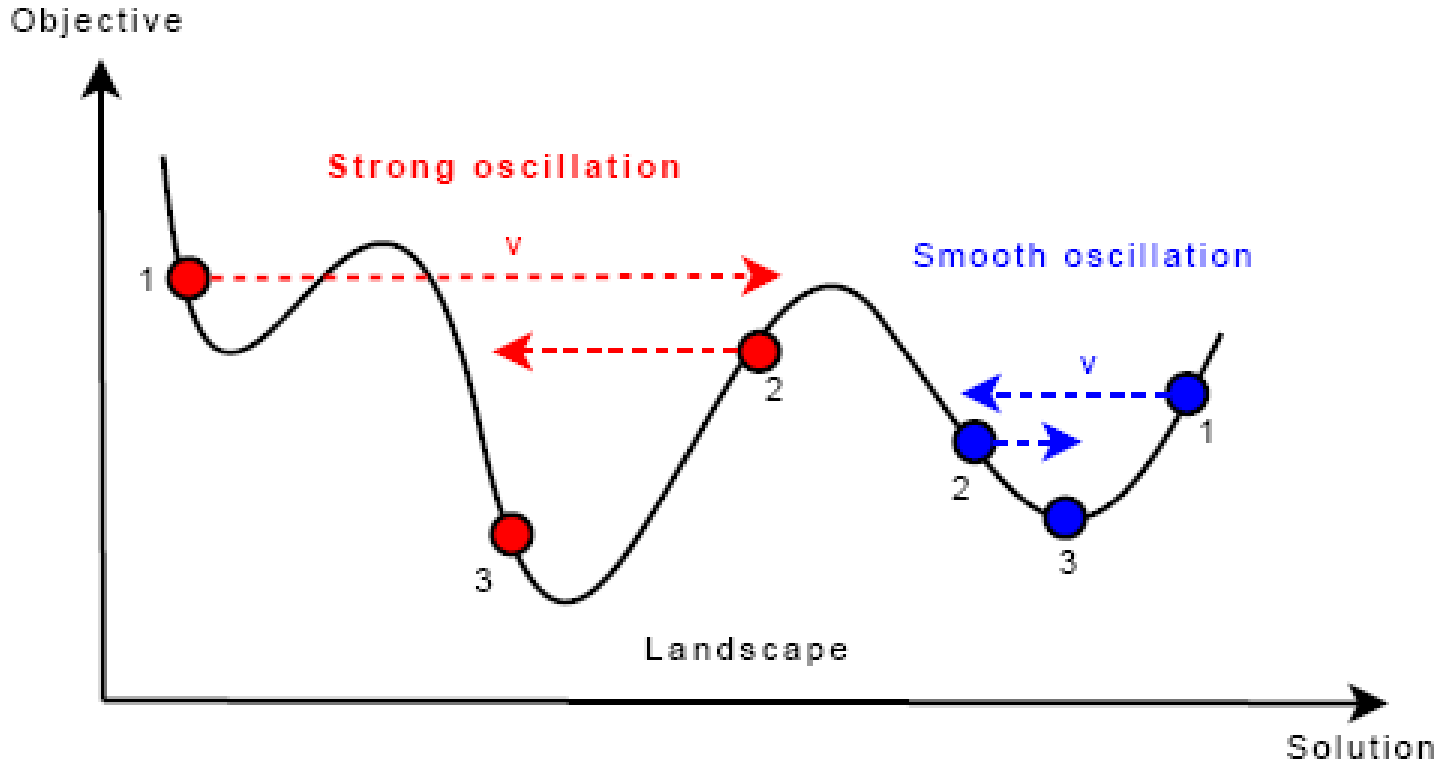


Fig. 3.37 Strong oscillations versus smooth ones in a particle swarm.

Bee colony in nature: organisation

- Queen
- Drones
- Workers
- Broods

Bee colony

- Nest site selection
- Food foraging
- Marriage process

Bee colony: Nest selection

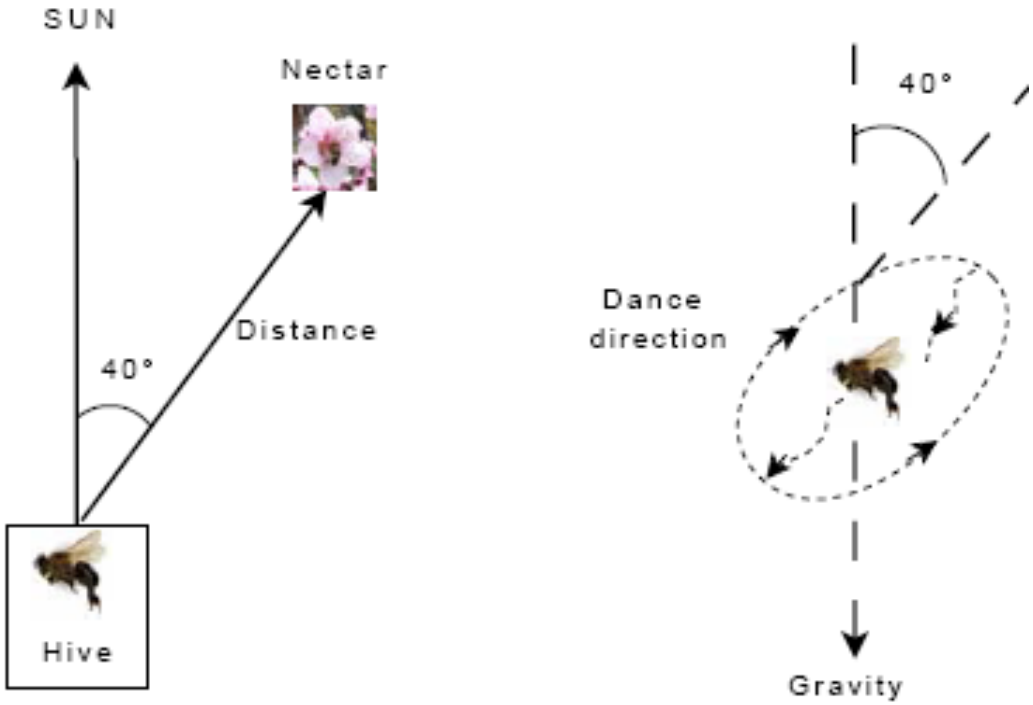


Fig. 3.38 The waggle dance: the direction is indicated by the angle from the sun; the distance is defined by the duration of the waggle part of the dance. A waggle run oriented 40 degrees to the right indicates a food source 40 degrees to the right of the direction of the sun outside the hive. The bee runs through a figure-eight pattern on a vertical comb. It passes through the central, and performs the waggle run with vibrating her body laterally. The waggle run duration is related to the food source distance with a rate of increase of about 75 milliseconds per 100 meters [692].

Bee colony: Food foraging

- Unemployed foragers
- Employed foragers

Table 3.8 Analogy between natural and artificial bee colonies.

Natural bee colony	Artificial bee colony
Food source	Solution
Quality of nectar	Objective function
Onlookers	Exploitation of search
Scout	Exploration of search

- Exploration of food sources
- Exploitation of food sources

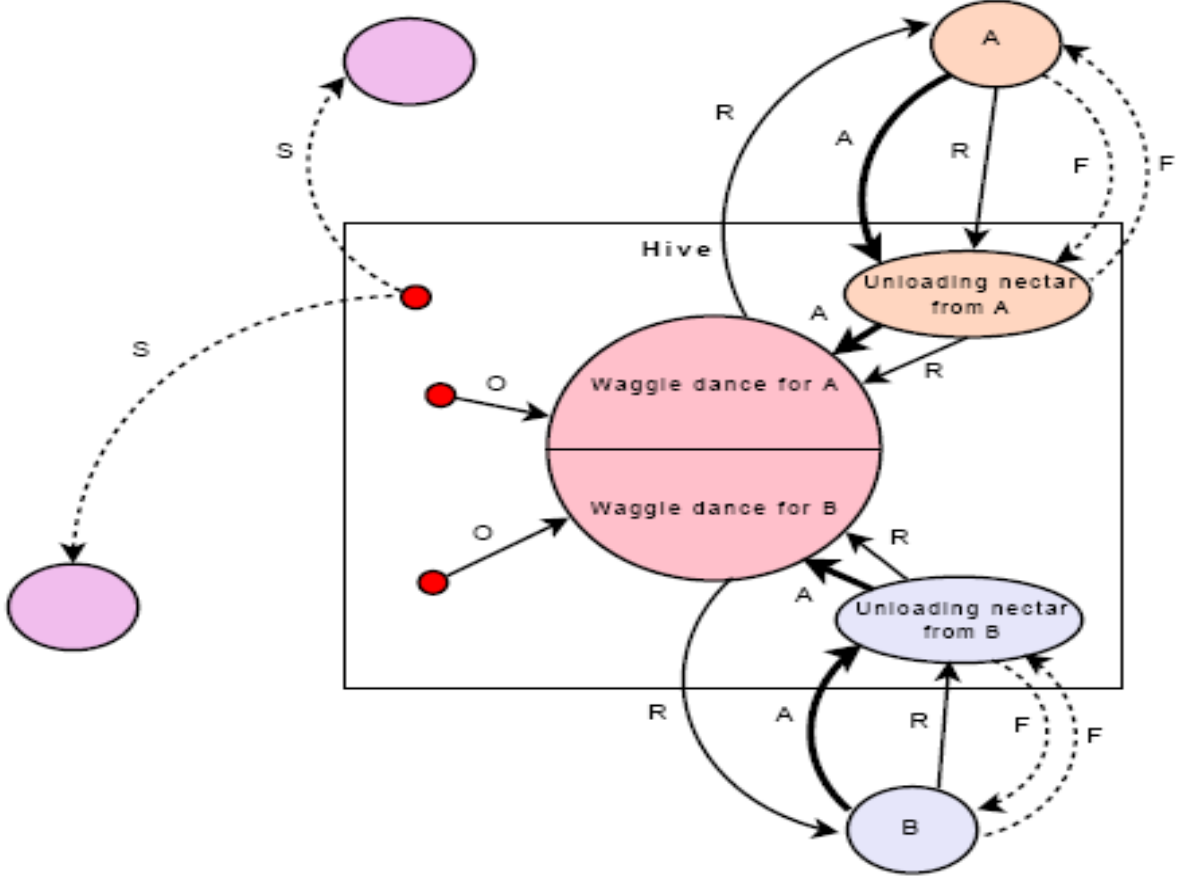


Fig. 3.39 Bee colony behavior for food source (nectar) discovering. We assume two discovered food sources A and B [446]. An unemployed bee has no knowledge about food source. It can be a scout bee S, starting to explore the neighborhood of the nest, or a onlooker bee O, watching for the waggle dances. After the localization of a food source, the bee becomes an employed bee. After unloading the nectar, the employed bee has three alternatives: abandoning the food source (A), recruit other bees from the nest (R), foraging without recruiting other bees (F).

Template of the bee algorithm (BA)

Algorithm 3.15 Template of the bee algorithm (BA).

Random initialization of the whole colony of bees ;

Evaluate the fitness of the population of bees ;

Repeat /* Forming new population */

Select sites for neighborhood search ; Determine the patch size ;

Recruit bees for selected sites and evaluated their fitness ;

Select the representative bee from each patch ;

Assign remaining bees to search randomly and evaluate their fitness ;

Until Stopping criteria

Algorithm 3.16 Template of the Marriage in honeyBees Optimization (MBO) algorithm.

Random initialization of the queen's ;

Improve the queen with workers (S-metaheuristic) ;

For predefined maximum number of mating-flights **Do**

Initialize energy and speed ;

While queen's energy > 0 **Do**

The queen moves between states and probabilistically chooses drones ;

If a drone is selected **Then**

Add its sperm to the queen's spermatheca ;

Update the queen's internal energy and speed ;

Endwhile

Generate broods by haploid-crossover and mutation ;

Use the workers (S-metaheuristic) to improve the broods ;

Update workers' fitness ;

If The best brood is fitter than the queen **Then**

Replace the queen's chromosome with the best brood's chromosome ;

Kill all broods ;

Endfor

Artificial Immune Systems

- Representation: representation of AIS components (e.g. antigens, antibodies, cells, molecules, …)
- Affinity: evaluation of the interaction of the system's components (i.e. each other and with the environment)
- Adaptation: procedure that govern the dynamics of the whole system

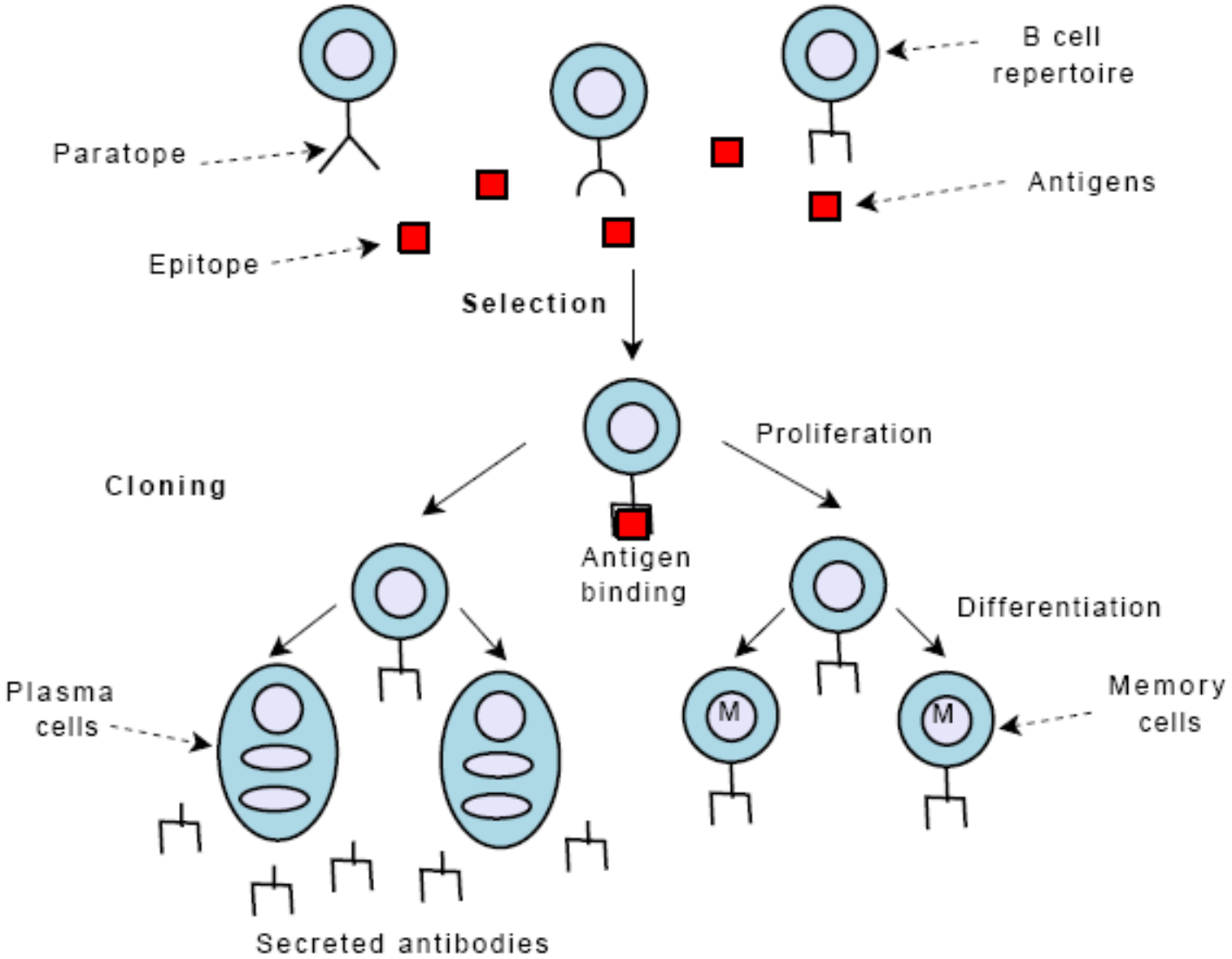
Artificial Immune Systems

- Population-based: do not take into account the immune network
  - Clonal selection
  - Negative selection
- Network-based: inspired by the netxork theory
  - Immune networks
  - Danger theory

AIS in nature

- Two type of immunity
  - Innate immune system: the body is born with the ability to recognize a microbe and immediately destroy it. It protects our body from non-specifics pathogens
  - Adaptive immune system: acquired immune system complete the innate one and removes the pathogens that persist to it
    - Lymphocytes: two type of cells: B cells and T cells responsible for recognizing and destroying any antigen

AIS: Clonal selection



AIS: Clonal selection

Table 3.9 Analogy between the natural immune system and the optimization problem according to the clonal selection theory.

Natural immune system	Optimization problem
Antibody	Solution
Affinity	Objective function
Antigen	Optimization problem
Cloning	Reproduction of solutions
Somatic mutation (hypermutation)	Multiple mutation of a solution
Affinity maturation	Mutation and selection of best solutions
Receptor editing	Diversification

AIS: Clonal selection

Algorithm 3.17 Template of the CLONALG (CLOnal selection ALGorithm).

Input: Initial population P_0.

$P = P_0$; /* Generation of the initial population of random antibodies */

Repeat

Evaluate all existing antibodies and compute their affinities ;

Select $N\%$ of antibodies with highest affinities ;

Clone the selected antibodies ;

Maturate the cloned antibodies ;

Evaluate all cloned antibodies ;

Add $R\%$ of best cloned antibodies to the pool of antibodies ;

Remove worst members of the antibodies pool ;

Add new random antibodies into the population ;

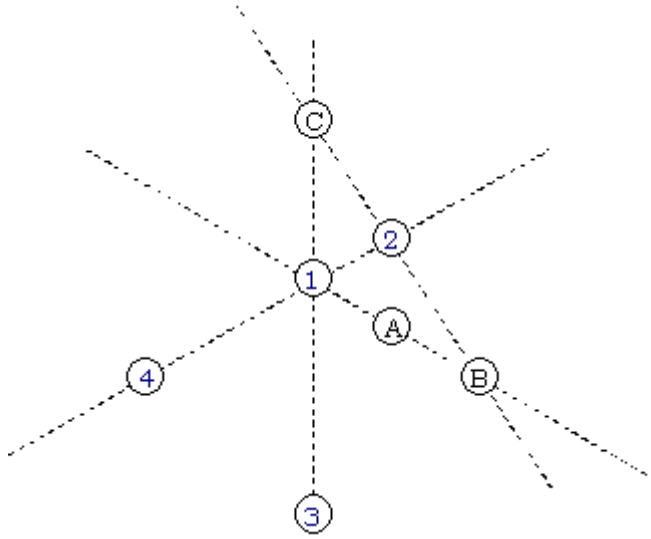
Until Stopping criteria satisfied

Output: Best population found.

Scatter search and path relinking

Scatter Search and Path Relinking

- Scatter Search and its generalized form Path Relinking provide unifying principles for joining (or recombining) solutions based on generalized path constructions in Euclidean or neighborhood spaces



F. Glover. “Scatter Search and Path relinking”.
New ideas in optimization, Advanced topics in computer science series, 1999.

F. Glover, M. Laguna and R. Marti. “Fundamentals of scatter search and path relinking”. Control and Cybernetic, 2000.

Main operators

- Diversification Generation Method : Generate a collection of diverse trial solutions
- Improvement Method : Transform a trial solution into one or more enhanced trial solutions
- Reference Set Update Method : Build and maintain a reference set - "best" solutions found (quality, diversity)
- Subset Generation Method : Operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions
- Solution Combination Method : Transform a given subset of solutions produced by the Subset Generation Method

Scatter search components

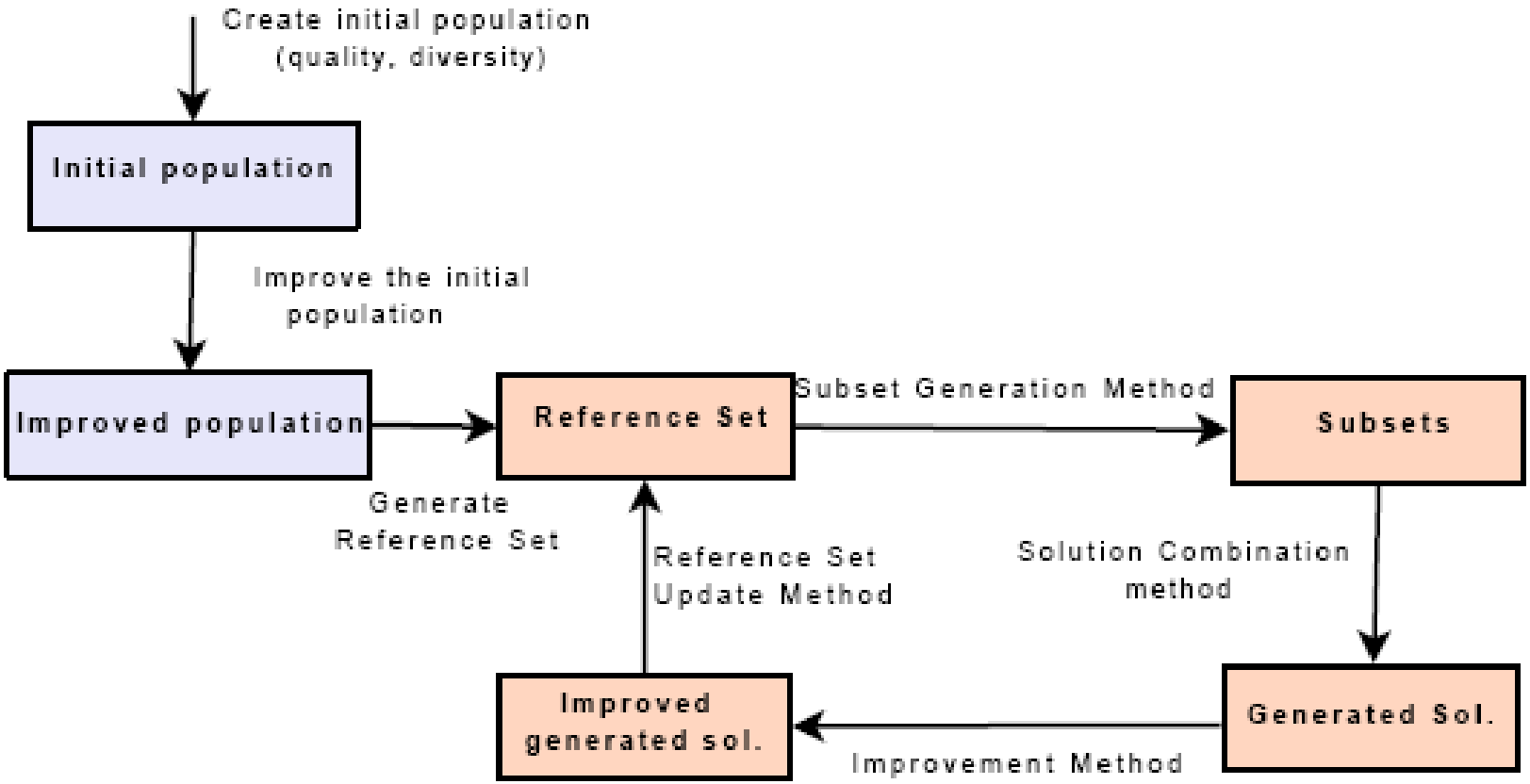


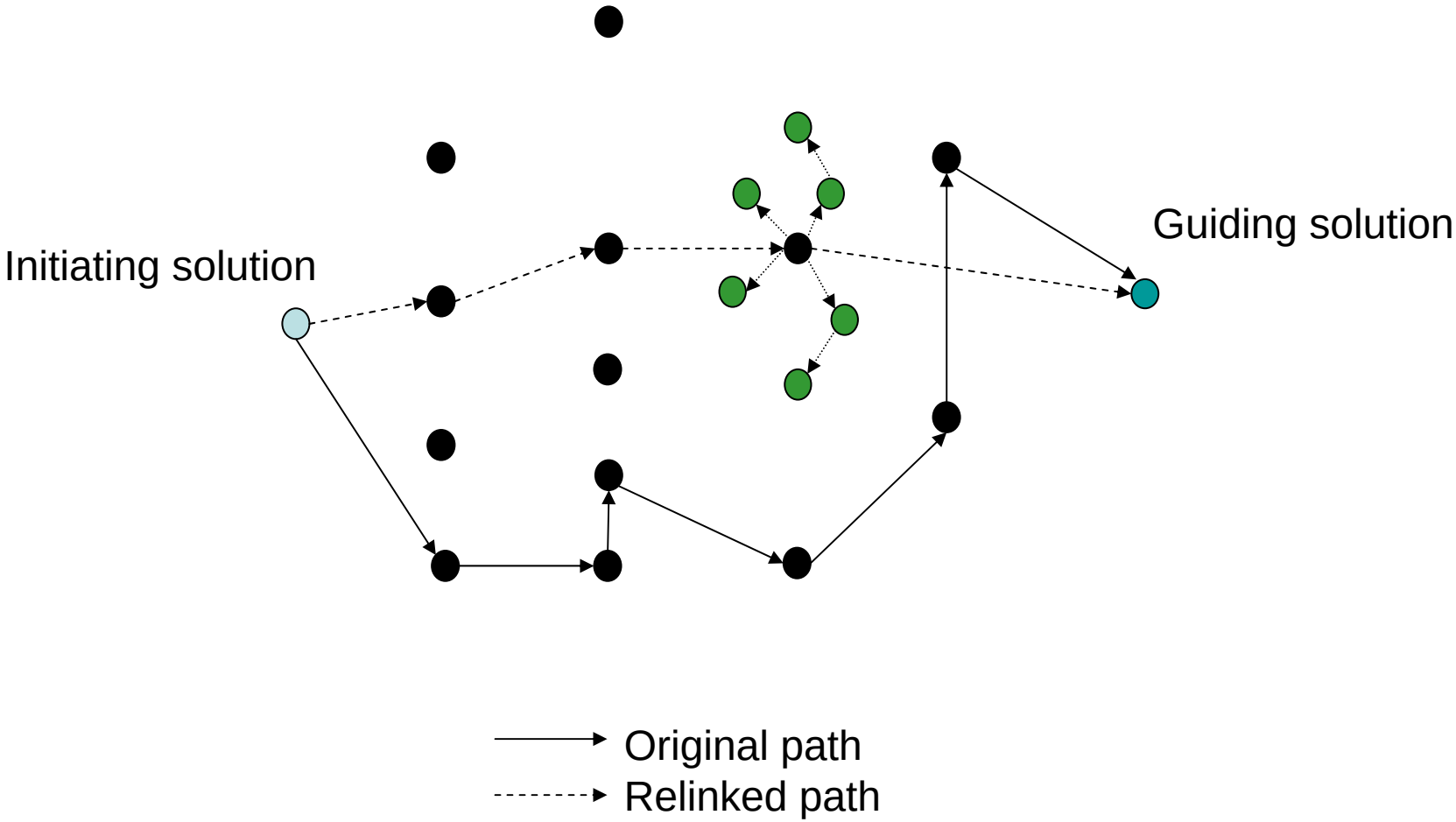
Fig. 3.29 Search components of scatter search algorithms.

Template of Scatter Search

Algorithm 3.10 Template of the scatter search algorithm (SS).

```
/* Initial phase */
Initialize the population Pop using a diversification generation method ;
Apply the improvement method to the population ;
Reference set Update Method ;
/* Scatter search iteration */
Repeat
   Subset generation method ;
   Repeat
      Solution Combination Method ;
      Improvement Method ;
   Until Stopping criteria 1
Reference Set Update Method ;
Until Stopping criteria
Output: Best found solution or set of solutions.
```

Path-Relinking Solutions



Template of Path relinking

Algorithm 3.11 Template of the basic path relinking (PR) algorithm.

Input: Starting solution s and target solution t.

$x = s$;

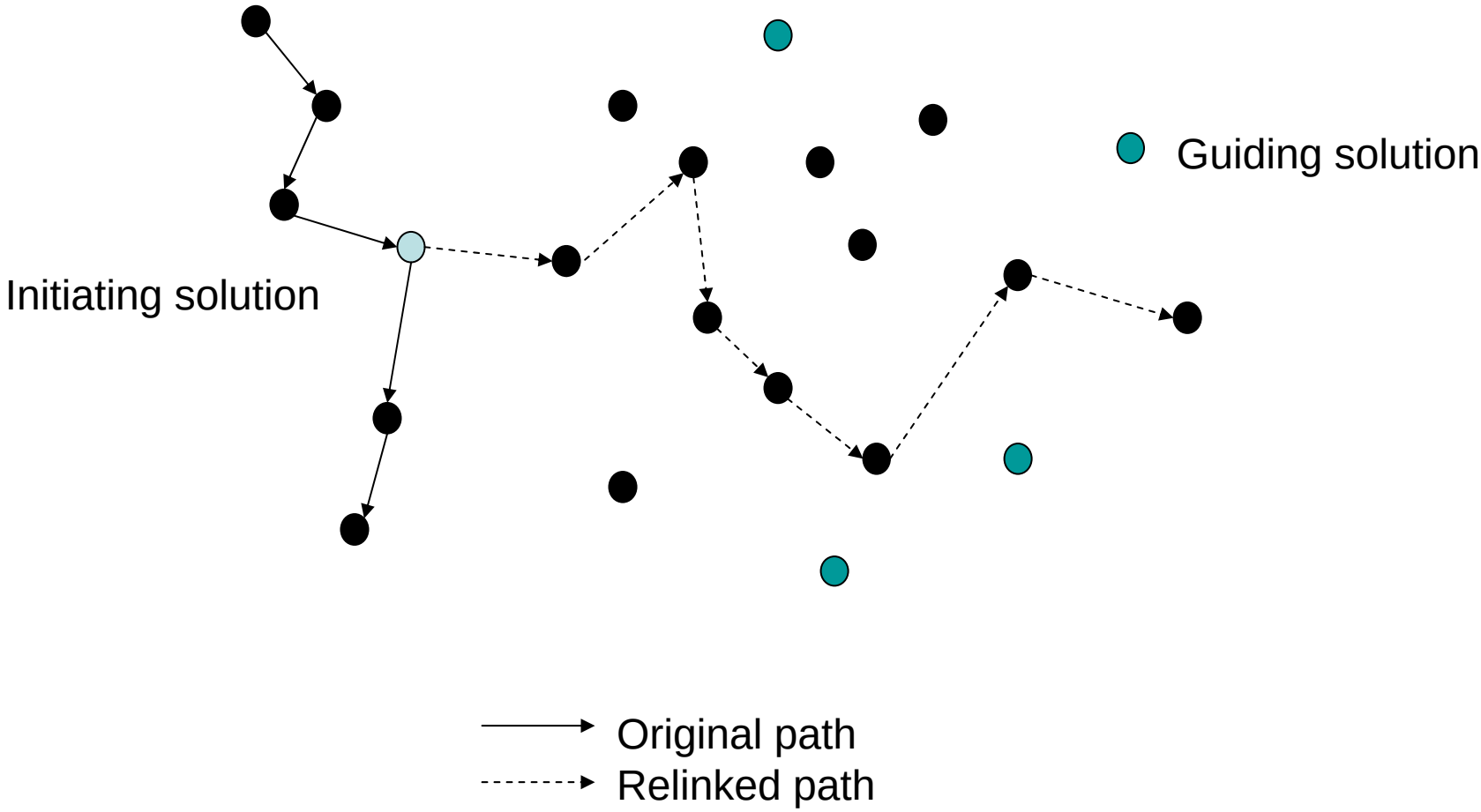
While $dist(x, t) \neq 0$ **Do**

Find the best move m which decreases $dist(x \oplus m, t)$;

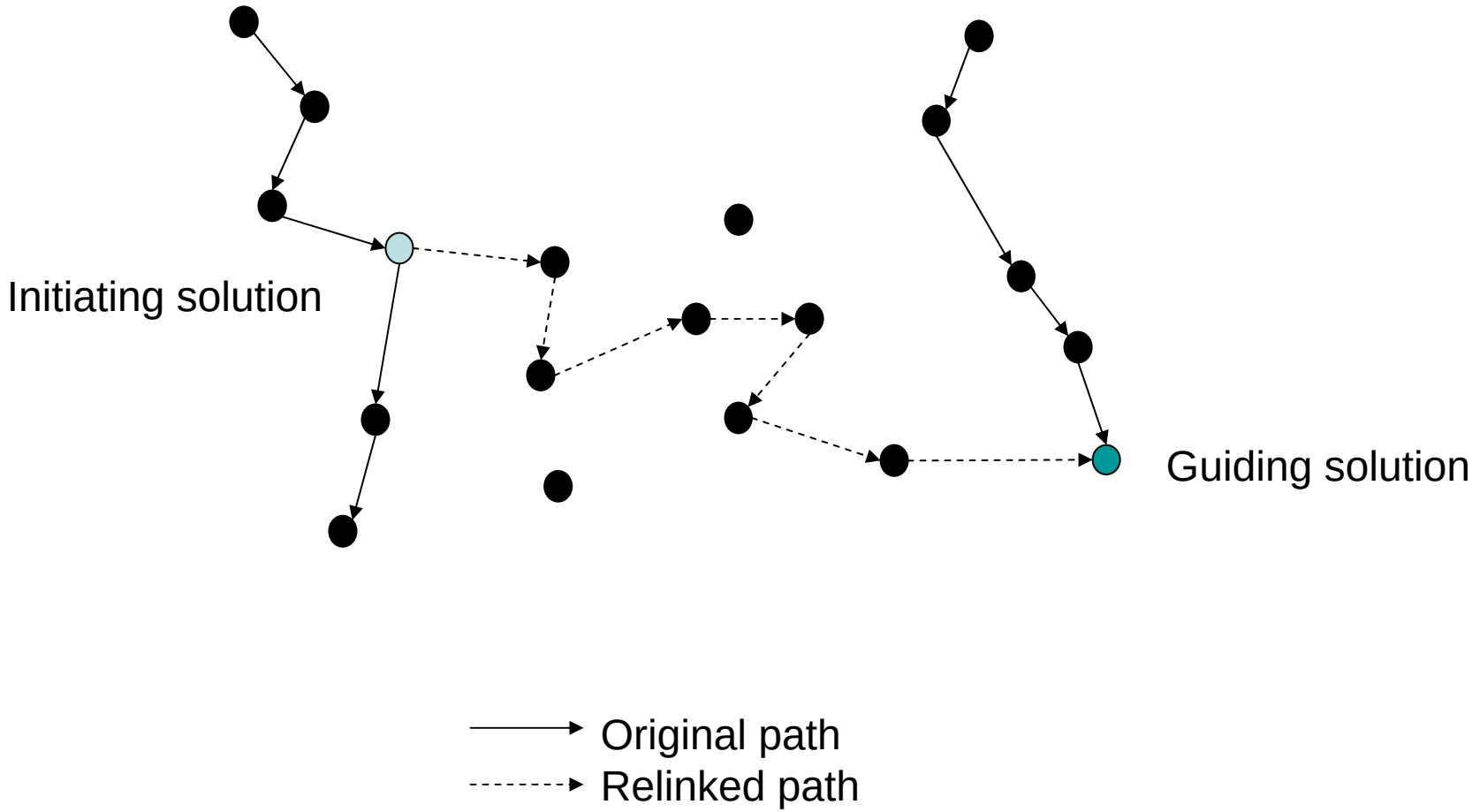
$x = x \oplus m$; /* Apply the move m to the solution x */ ;

Output: Best solution found in trajectory between s and t.

Multiple Guiding Solutions



Linking Solutions



Path relinking strategies

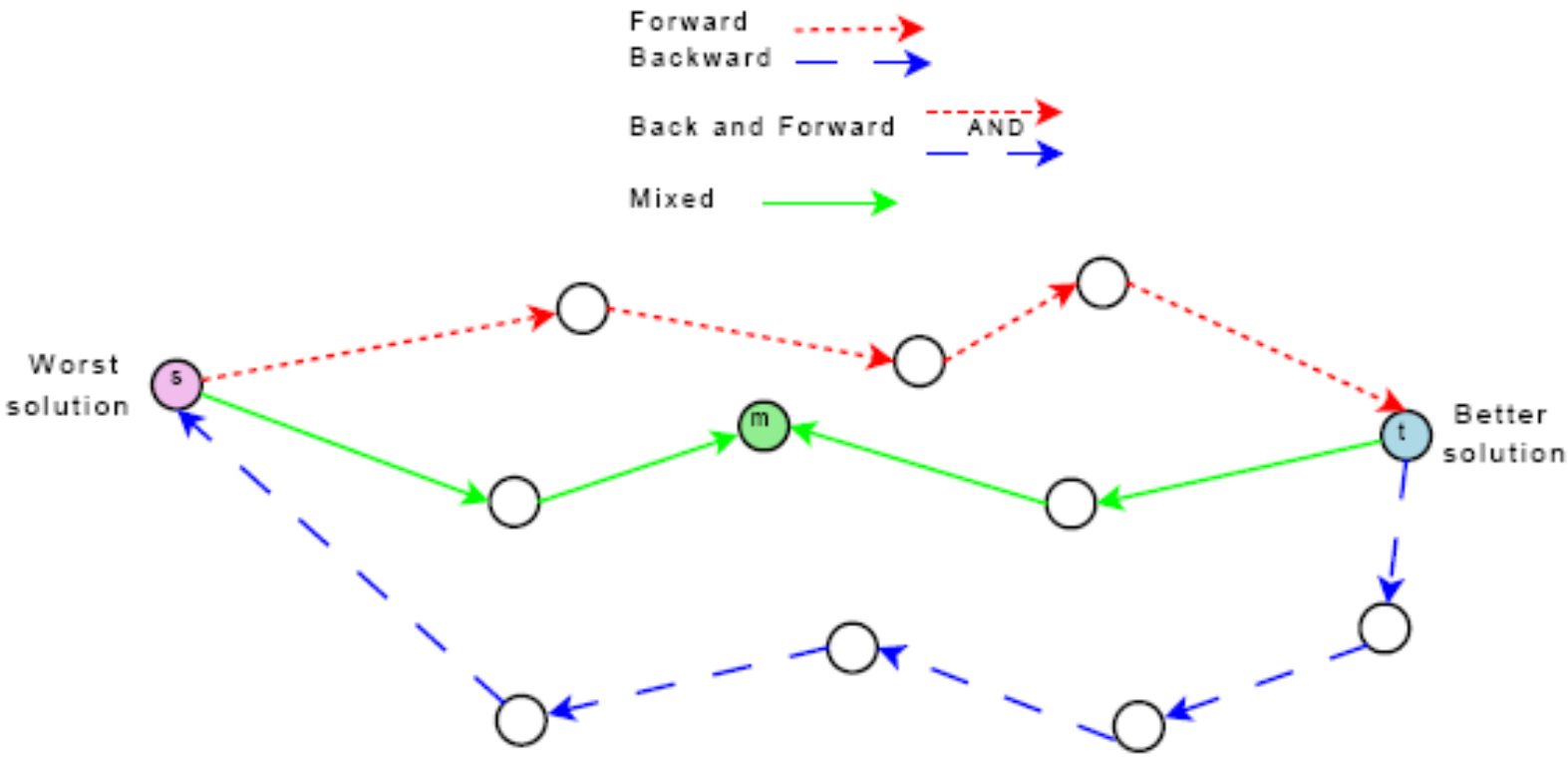
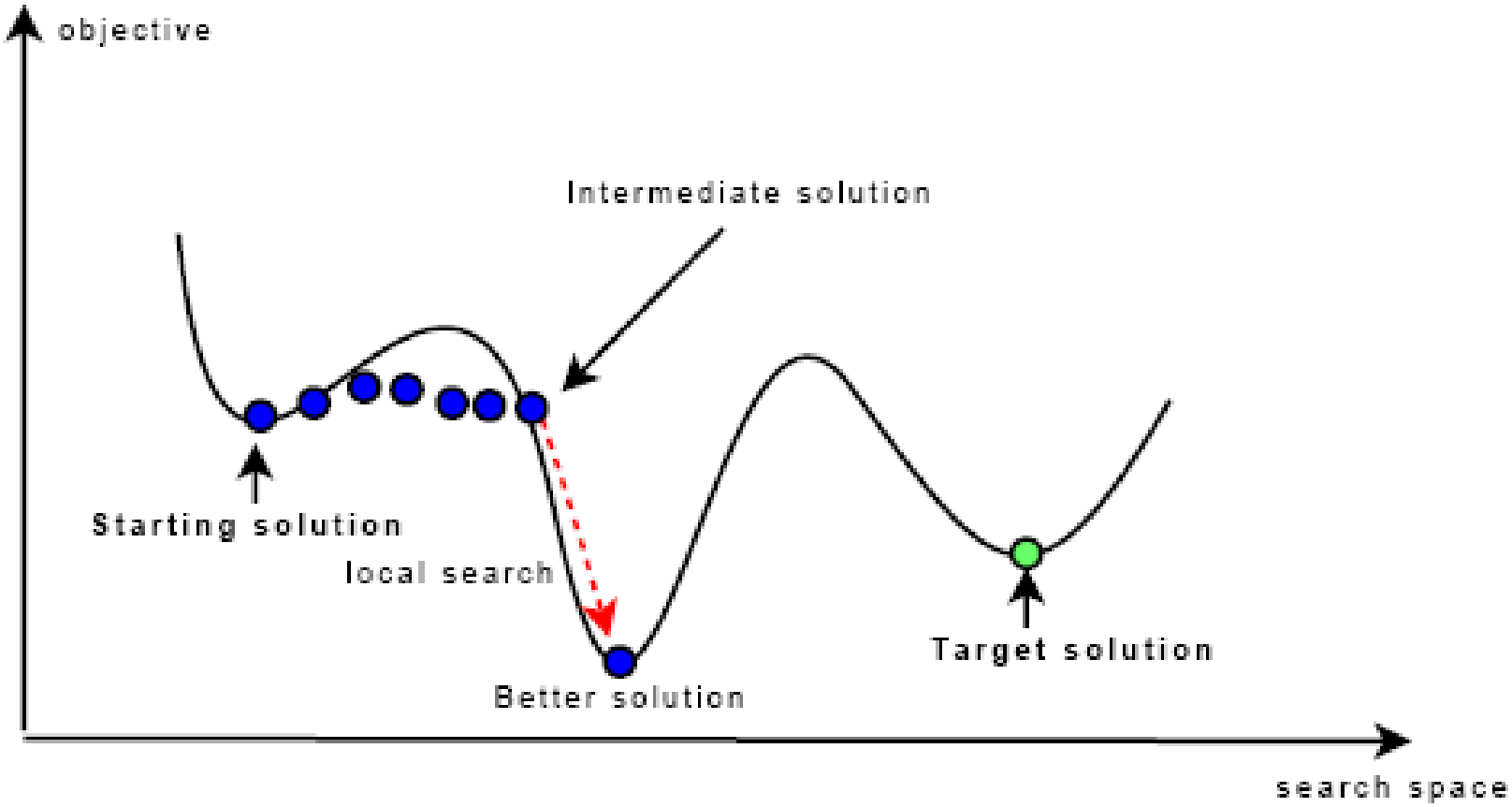


Fig. 3.31 Different path relinking strategies in terms of the starting and guiding solutions.

Intermediate Solutions



Estimation of Distribution Algorithm (EDA)

Estimation of Distribution Algorithm

- Based on the use of (unsupervised) density estimators/generative statistical models
- Idea is to convert the optimization problem into a search over probability distributions
- The probabilistic model is in some sense an explicit model of (currently) promising regions of the search space

Estimation of Distribution Algorithm

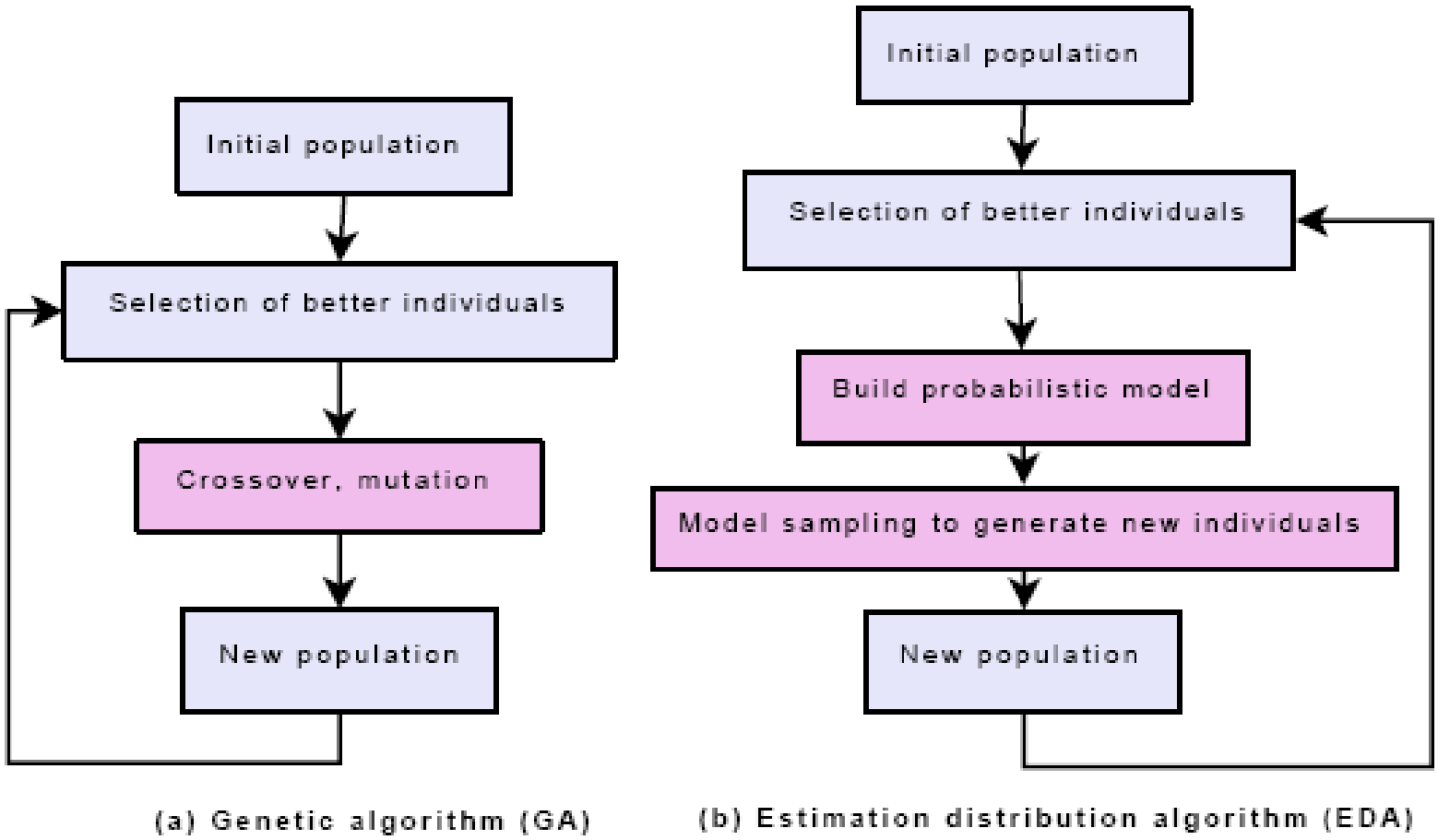


Fig. 3.23 Estimation of distribution algorithms versus genetic algorithms.

Template of the EDA algorithm

Algorithm 3.4 Template of the EDA algorithm.

$t = 1$;

Generate randomly a population of n individuals ;

Initialize a probability model $Q(x)$;

While Termination criteria are not met **Do**

  Create a population of n individuals by sampling from $Q(x)$;

  Evaluate the objective function for each individual ;

  Select m individuals according to a selection method ;

  Update the prob. model $Q(x)$ using selected population and $f()$ values ;

  $t = t + 1$;

End While

Output: Best found solution or set of solutions.

Level of variable interaction in the definition of the probabilistic model

- Univariate : no interaction between variables
- Bivariate: interaction between two variables
- Multivariate: more than two variables

EDA simplest probability model:
Population-based incremental Learning (PBIL)

Algorithm 3.5 Template of the PBIL algorithm (Population-based incremental learning).

Initial distribution $D = (0.5, ..., 0.5)$;

Repeat

Generation of the population ;

If $r < D_i$ (r uniform in $[0,1]$) **Then** $X_i = 1$

Else $X_i = 0$;

Evaluate and sort the population ;

Update the distribution $D = (1 - \alpha)D + \alpha X_{best}$;

Until Stopping criteria

EDA simplest probability model

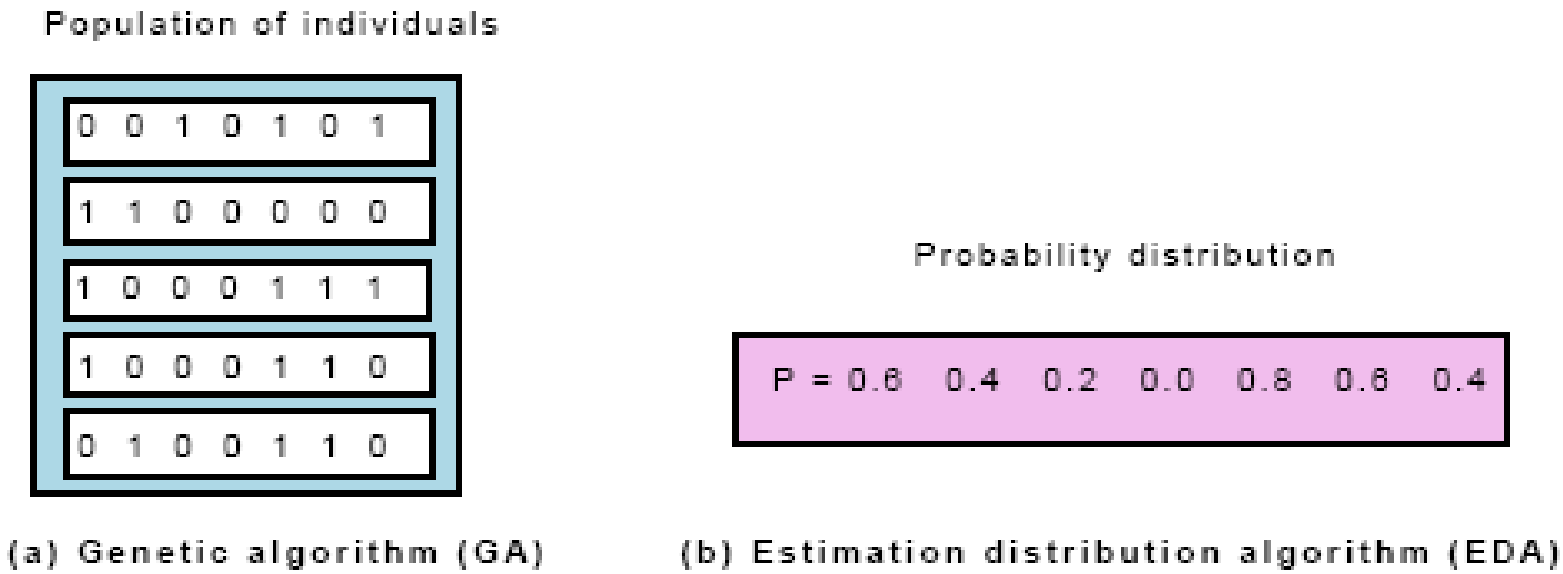


Fig. 3.24 EDA and GA in solving a bit vector optimization problem.

Probability distribution in PBIL

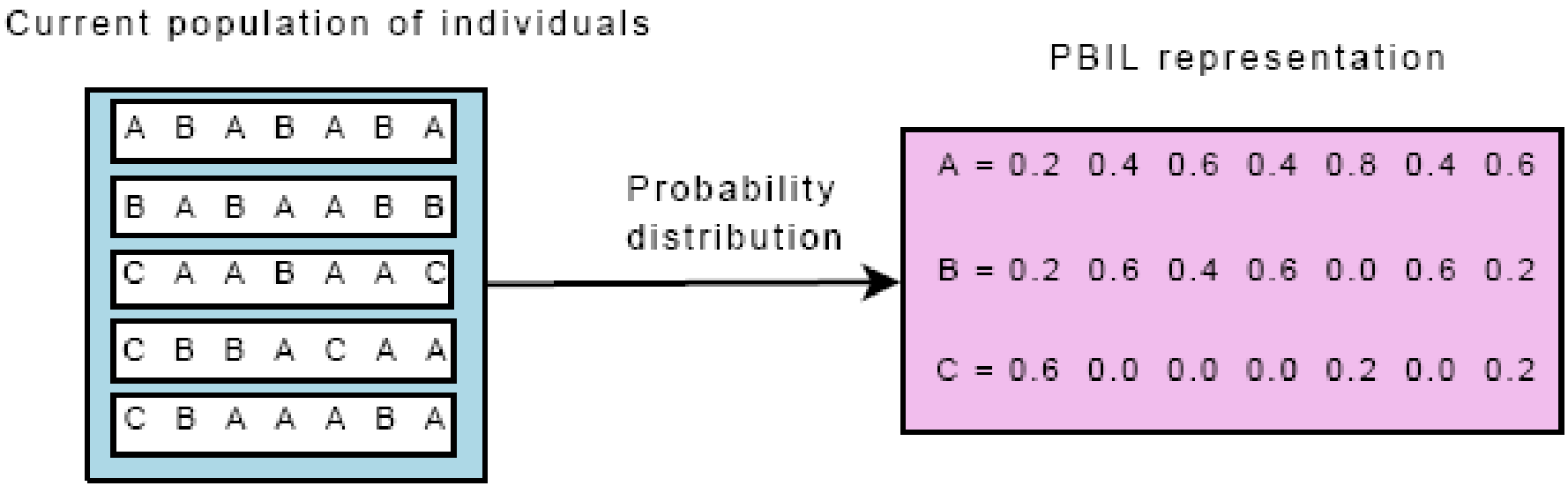


Fig. 3.25 Probability distribution in the PBIL (Population-based Incremental Learning) algorithm.

Other probability models

- Mutual Information Maximization for Input Clustering (MIMIC) regards pairwise dependances
- Gaussian networks for continuous optimization
- Bayesian networks for discrete optimization: Bayesian Optimization Algorithm (BOA) for multivariate dependances

Template of the BOA algorithm

Algorithm 3.6 Template of the BOA algorithm (Bayesian Optimization Algorithm).

Initialize the population $P(0)$ randomly ; Evaluate $(P(0))$;

Repeat

Select a set of promising solutions $S(t)$ from $P(t)$;

Construct the network B using a given metric and constraints ;

Generate new solutions $O(t)$ according to the joint distribution encoded by B ;

Create population $P(t+1)$ by replacing some solutions from $P(t)$ with $O(t)$;

Evaluate $P(t))$; $t = t + 1$;

Until Stopping criteria

EDA : Applications

- Field of EDA is quite young. Much effort is focused on methodology rather than performance

- First applications
  - Knapsack problem
  - Job Shop Scheduling

Co-evolutionary algorithms: competitive

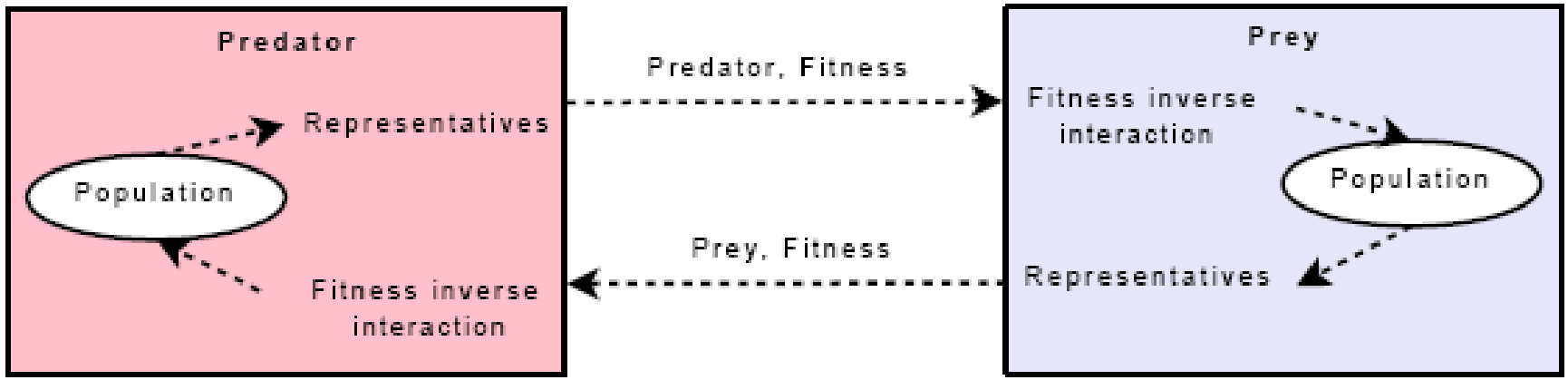


Fig. 3.26 Competitive coevolutionary algorithms based on the predator-prey model.

Co-evolutionary algorithms: cooperative

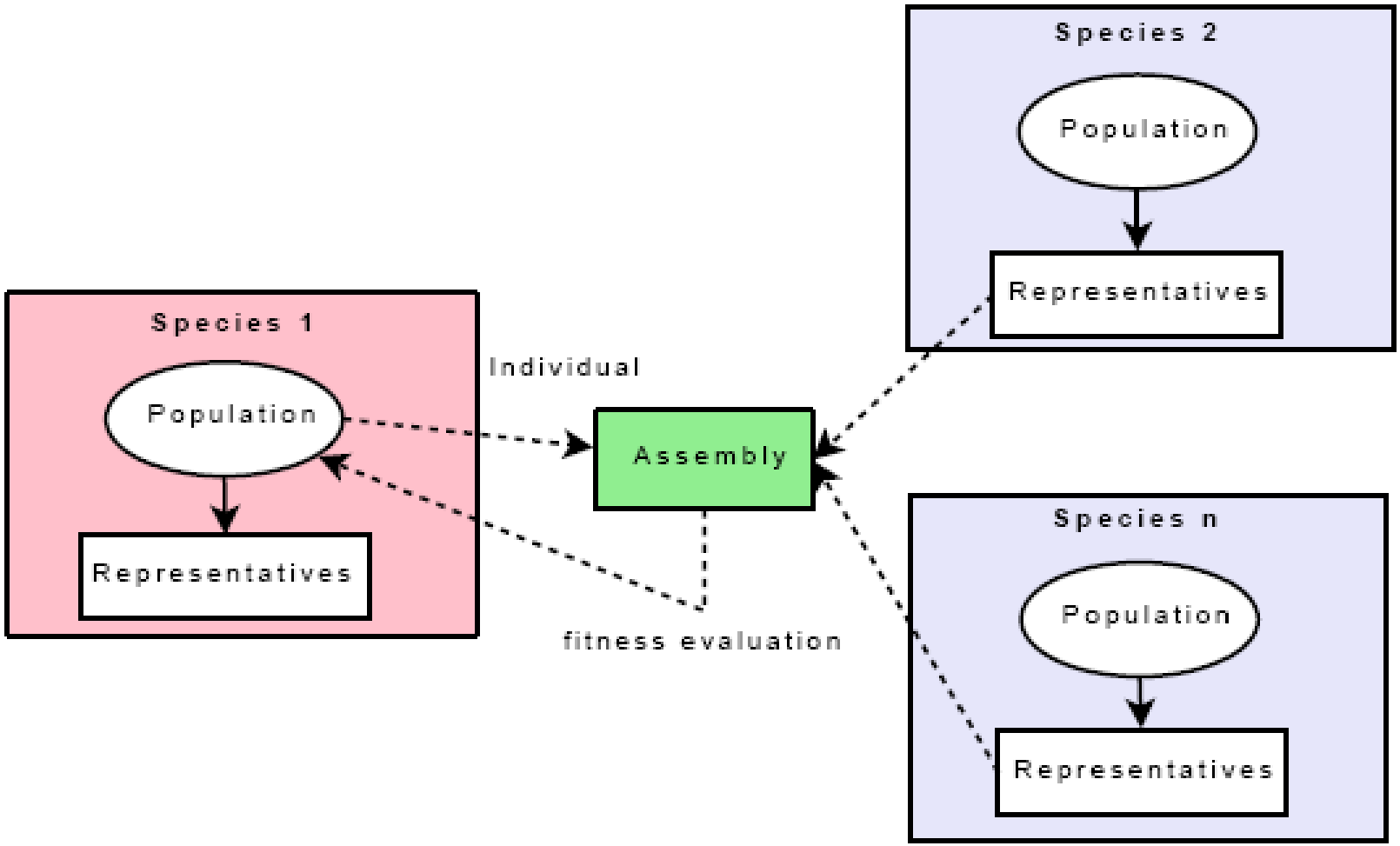


Fig. 3.27 A cooperative coevolutionary algorithm.

Cultural algorithms

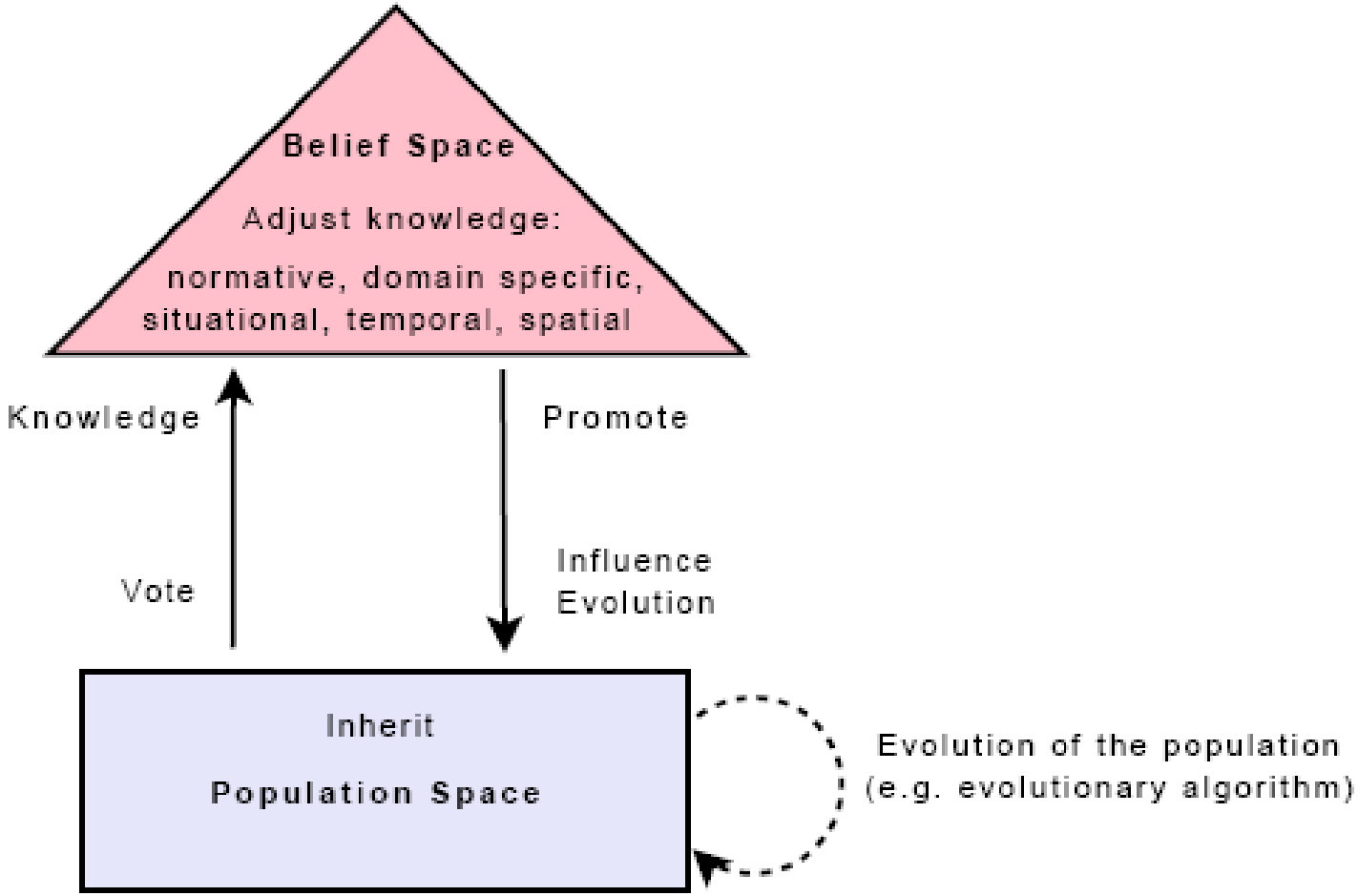


Fig. 3.28 Search components of cultural algorithms.

Cultural algorithms

Algorithm 3.9 Template of the cultural algorithm (CA).

Initialize the population $Pop(0)$;

Initialize the Belief $BLF(0)$;

$t = 0$;

Repeat

Evaluate population $Pop(t)$;

Adjust(BLF(t), Accept(POP(t))) ;

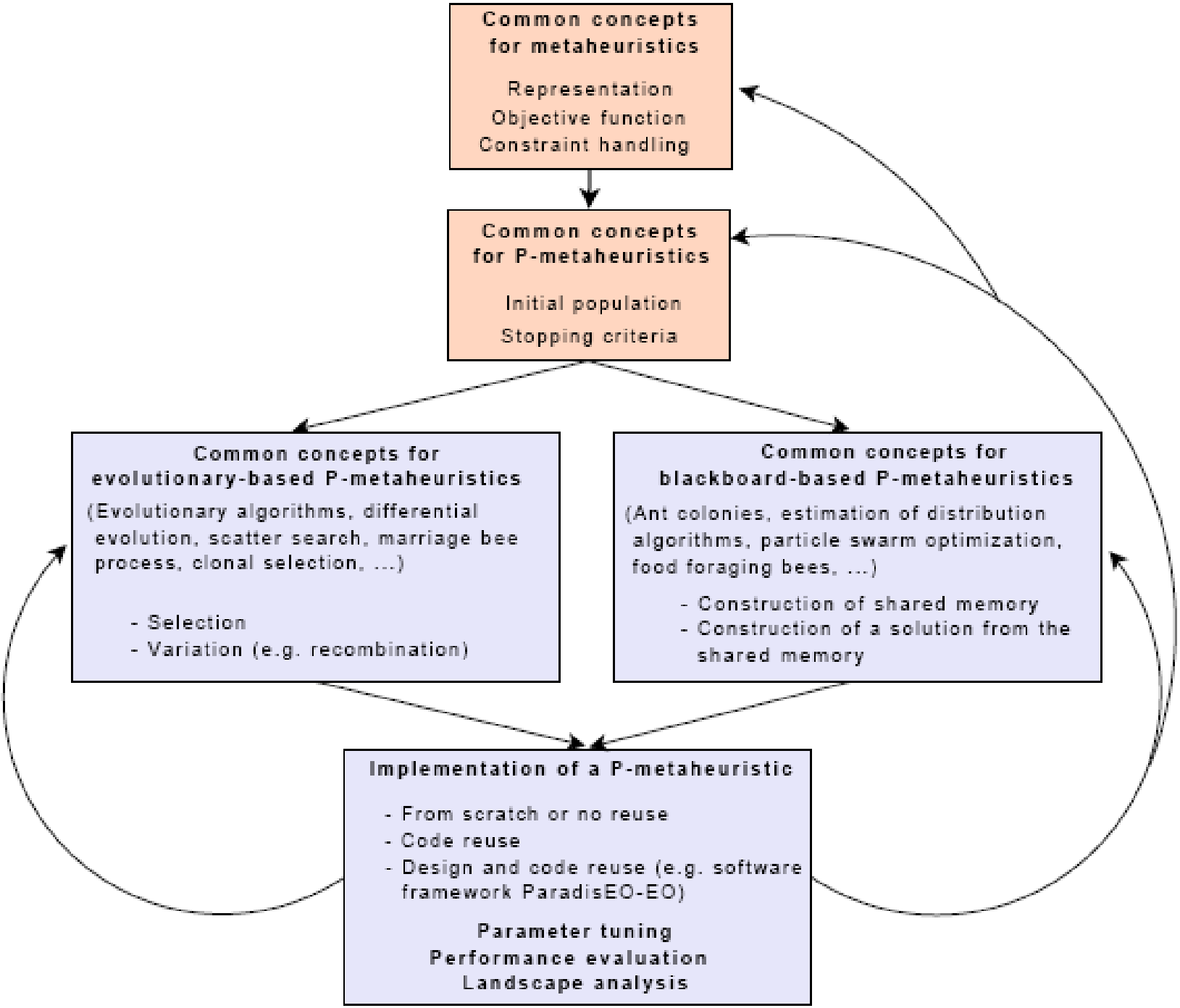
Evolve(Pop(t+1),Influence(BLF(t))) ;

$t = t + 1$;

Until Stopping criteria

Output: Best found solution or set of solutions.

Guideline to design and implement a P-metaheuristic


Common concepts for metaheuristics
Representation
Objective function
Constraint handling
Common concepts for P-metaheuristics
Initial population
Stopping criteria
Common concepts for evolutionary-based P-metaheuristics
(Evolutionary algorithms, differential evolution, scatter search, marriage bee process, clonal selection, ...)
- Selection
- Variation (e.g. recombination)
Common concepts for blackboard-based P-metaheuristics
(Ant colonies, estimation of distribution algorithms, particle swarm optimization, food foraging bees, ...)
- Construction of shared memory
- Construction of a solution from the shared memory
Implementation of a P-metaheuristic
- From scratch or no reuse
- Code reuse
- Design and code reuse (e.g. software framework ParadisEO-EO)
Parameter tuning
Performance evaluation
Landscape analysis