Evolutionary Algorithms

Meta heuristics

and related optimization techniques II/II

Prof. Dr. Rudolf Kruse **Pascal Held**

{kruse,pheld}@iws.cs.uni-magdeburg.de

Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Institute of Knowledge and Language Engineering

Outline

Swarm- and Population-Based Optimization

swarm intelligence

- part of AI's developing intelligent multi-agent systems
- inspired by the behaviour of certain species, in particular
  - social insects (e.g. ants, termites, bees etc.) and
  - animals living in swarms (e.g. fish, birds etc.)

these species are capable of solving complex tasks by cooperation.

main idea

- generally quite simple individuals with limited skills
- self-coordinated without central control unit
- individuals exchanging information (cooperation)

techniques are classified by their way of information exchange

Techniques

Genetic/Evolutionary Algorithms

- biological pattern: evolution of life
- exchange of information by recombination of genotypes
- every individual serves as a candidate solution

Population Based Incremental Learning

- biological pattern: evolution of life
- exchange of information by prevalence in population
- every individual serves as a candidate solution

Techniques

Particle Swarm Optimization

- biological pattern: foraging of fish or bird swarms for food
- exchange of information by aggregation of single solutions
- every individual serves as a candidate solution

Ant Colony Optimization

- biological pattern: ants searching a route for food
- exchange of information by manipulating their environments (stigmergy, extended phenotype to Darwin)
- individuals generate a candidate solution

Population based incremental learning (PBIL)

- genetic algorithm without population
- instead: only store population statistics ⇒ by $\mathcal{G} = \{0, 1\}^L$ for all L bits the frequency of „1“
- specific individuals (e.g. for evaluation) are generated randomly according to the statistical frequency
- recombination: uniform crossover ⇒ implicitly when generating an individual
- selection: choosing the best individuals B for updating the population statistics $Pr_k^{(t)} \leftarrow B_k \cdot \alpha + Pr_k^{(t-1)}(1 - \alpha)$
- mutation: bit-flipping ⇒ slightly random changes within the population statistics

Algorithm 1 PBIL

Input: evaluation function F
Output: best individual A_{best}

$t \leftarrow 0$
$A_{\text{best}} \leftarrow$ create random individual from $\mathcal{G} = \{0, 1\}^L$
$Pr^{(t)} \leftarrow (0.5, \ldots, 0.5) \in [0, 1]^L$
while termination condition not satisfied {
    $P \leftarrow \emptyset$
    for $i \leftarrow 1, \ldots, \lambda$ {
        $A \leftarrow$ generate individual from $\{0, 1\}^L$ according to $Pr^{(t)}$
        $P \leftarrow P \cup \{A\}$
    }
    evaluate P according to F
    $B \leftarrow$ select best individuals P
    if $F(B) \succ F(A_{\text{best}})$ {
        $A_{\text{best}} \leftarrow B$
    }
    $t \leftarrow t + 1$
    for each $k \in \{1, \ldots, L\}$ {
        $Pr_k^{(t)} \leftarrow B_k \cdot \alpha + Pr_k^{(t-1)}(1 - \alpha)$
    }
    for each $k \in \{1, \ldots, L\}$ {
        $u \leftarrow$ draw a random number according to $U((0, 1])$
        if $u < p_m$ {
            $u' \leftarrow$ draw a random number according to $U(\{0, 1\})$
            $Pr_k^{(t)} \leftarrow u' \cdot \beta + Pr_k^{(t)}(1 - \beta)$
        }
    }
}
return A_{best}

PBIL: Typical Parameters

learning rate α

- low: emphasizes exploration
- high: emphasizes fine tunig

parameter	co-domain
population size λ	20–100
learning rate α	0.05–0.2
mutation rate p_m	0.001–0.02
mutation constant β	0.05

PBIL: Problemes

- algorithm might learn depencencies between certain single bits
- PBIL considers single bits isolated of each other

example:

population 1					population 2			
1	1	0	0	individual 1	1	0	1	0
1	1	0	0	individual 2	0	1	1	0
0	0	1	1	individual 3	0	1	0	1
0	0	1	1	individual 4	1	0	0	1
0.5	0.5	0.5	0.5	population statistics	0.5	0.5	0.5	0.5

- same population statistics can represent different populations

PBIL: Alternatives

- better techniques for estimating the distribution of beneficial candidate solutions
- especially: modelling of internal dependencies (i.e. with bayesian networks)
- example: *Bayesian optimization algorithm (BOA)*
  - create initial population randomly
  - update population for a given number of iterations by applying selection and variation
  - perform selection as usual
  - for variation, apply Bayesian Network as a model of promising candidate solutions
  - create new candidate solutions by reproducing samples from the Bayesian Network

Particle Swarm Optimization

© Eric T. Schulz http://www.eeb.uconn.edu/courses/eeb296/ © Ariel Bravy http://www.skphoton.com/albums/

- fish or birds are searching for rich food resources in swarms
- orientation based on individual search (cognitive part) and other individuals close to them within the swarm (social part)
- also: living within a swarm reduces the risk of getting eaten by a predator

Particle Swarm Optimization

Particle Swarm Optimization [Kennedy and Eberhart, 1995]

- **motivation:** behaviour of swarms of fish (e.g.) when searching for food: randomly swarming out, but always returning to the swarm to exchange information with the other individuals
- **approach:** use a "swarm" of m candidate solutions instead of single ones
- **preconditions:** $\Omega \subseteq \mathbb{R}^n$ and thus the function f $f : \mathbb{R}^n \to \mathbb{R}$ to be maximized (w.l.o.g.)
- **procedure:** take every candidate solution as a "particle" searching for food at the position $\boldsymbol{x}_i$ with a velocity of $\boldsymbol{v}_i$. $(i = 1, \ldots, m)$

⇒ combine elements of ground-oriented search (e.g. gradient descent approach) and population-based search (e.g. EA)

Particle Swarm Optimization

update for position and velocity of particle i:

$$\mathbf{v}_i(t+1) = \alpha \mathbf{v}_i(t) + \beta_1 \left(\mathbf{x}_i^{(\text{local})}(t) - \mathbf{x}_i(t) \right) + \beta_2 \left(\mathbf{x}^{(\text{global})}(t) - \mathbf{x}_i(t) \right)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)$$

- parameter: β_1, β_2 randomly for every step, α decreasing with t
- $\mathbf{x}_i^{(\text{local})}$ is **local memory** of an individual (particle): the best coordinates being visited by this individual within the search space, i.e.
$$\mathbf{x}_i^{(\text{local})} = \mathbf{x}_i \left(\arg\max_{u=1}^{t} f(\mathbf{x}_i(u)) \right)$$
- $\mathbf{x}^{(\text{global})}$ is **global memory** of the swarm: the best coordinates being visited by any individual of the swarm within the search space (best solution so far), i.e.
$$\mathbf{x}^{(\text{global})}(t) = \mathbf{x}_j^{(\text{local})}(t) \quad \text{mit} \quad j = \arg\max_{i=1}^{m} f\left(\mathbf{x}_i^{(\text{local})} \right)$$

Algorithm 2 Particle swarm optimization

for each particle i $\{$
  $\boldsymbol{x}_i \leftarrow$ choose randomly within search space Ω
  $\boldsymbol{v}_i \leftarrow 0$
$\}$
do $\{$
  for each particle i $\{$
    $y \leftarrow f(\boldsymbol{x}_i)$
    if $y \geq f\left(\boldsymbol{x}_i^{(\text{local})}\right)$ $\{$
      $\boldsymbol{x}_i^{(\text{local})} \leftarrow \boldsymbol{x}_i$
    $\}$
    if $y \geq f\left(\boldsymbol{x}_i^{(\text{global})}\right)$ $\{$
      $\boldsymbol{x}^{(\text{global})} \leftarrow \boldsymbol{x}_i$
    $\}$
  $\}$
  for each particle i $\{$
    $\boldsymbol{v}_i(t+1) \leftarrow \alpha \cdot \boldsymbol{v}_i(t) + \beta_1 \left(\boldsymbol{x}_i^{(\text{local})}(t) - \boldsymbol{x}_i(t)\right) + \beta_2 \left(\boldsymbol{x}^{(\text{global})}(t) - \boldsymbol{x}_i(t)\right)$
    $\boldsymbol{x}_i(t+1) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
  $\}$
$\}$ **while** termination condition is not satisfied

Extensions

- **reduced search space:** if Ω is a proper subset of $\mathbb{R}^n$ (e.g. hypercube $[a, b]^n$), then all particles will be reflected and bounce off the boundaries of the search space
- **local environment of a particle:** use best local memory of a single particle instead of global swarm memory, e.g. particles surrounding the currently updated one
- **automatic parameter adjustment:** e.g. changing the swarm size (particles being much worse than the currently updated one are extinguished)
- **diversity control:** prevent early convergence to suboptimal solutions e.g. by introducing a new random number for updating the speed to increase diversity

Ant Colony Optimization

© PeTA http://www.helpingwildlife.com/ants.asp

© NickLyonMedia http://nicklyon.orchardhostings4.co.uk

- since food has to be fetched from its source and carried to the nest, ants form transportation roads
- to do this, they label all their routes with scents (pheromones) for other ants may then trace their routes
- this way, routes to food sources are minimized

Ant Colony Optimization

Ant Colony Optimization [Dorigo and Stützle, 2004]

motivation: some ant species are able to find shortest route to food sources by placing and tracing pheromones (scents)

- intuitively: short routes are labeled with more pheromone during the same time
- routes are randomly chosen according to the current pheromone distribution: the more pheromone there is, the more probable is it for ants to choose this way
- the amount of pheromone might vary according to the quality and amount of food found

main principle: stigmergy

- ants are communicating implicitly by placing pheromones
- stigmergy (indirect communication by changing the environmental circumstances) allows for globally adapted behaviour due to locally found information

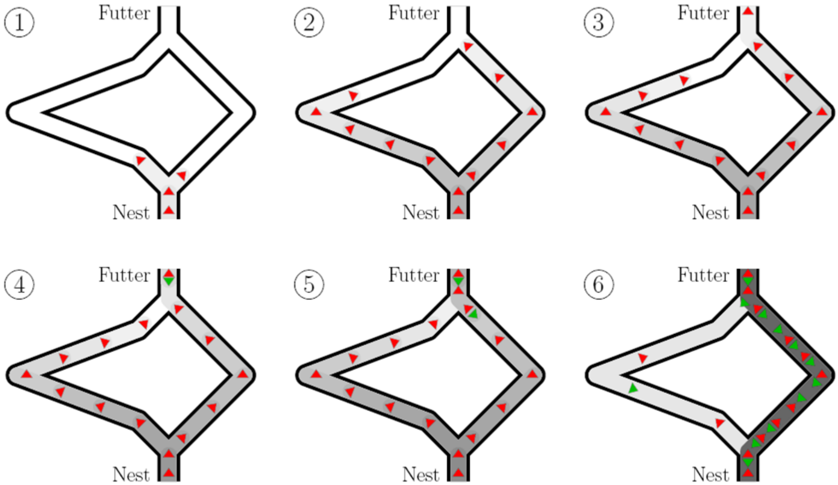
Double-Bridge Experiment [Goss et al., 1989]

- ant nest and food source are connected by 2 bridges that differ in length
- experiment has been run with Argentinian Ants *Iridomyrmex Humilis*: these ants are almost blind (as most other ants, too) so they can't "see" which bridge is shorter
- in most runs: after just several minutes most ants were using the shorter bridge

explanation

- ants travelling the shorter bridge are reaching the food earlier so in the first place the end of the shorter bridge gets more pheromon
- when returning to the nest, choosing the shorter bridge again is more probable for it is labeled with more pheromone now, thus increasing the difference in pheromone even more

Double-Bridge Experiment



Double-Bridge Experiment: Principle

- shorter route is intensified automatically (autocalysis): more pheromon $\longleftrightarrow$ more ants will choose this route
- note: ants are able to find shortest path only because they return on it while placing pheromons again
- when only placing pheromons while running **towards** the food source:
  - at the nest there is no way to decide between both paths as there is no difference in pheromone
  - at the junction of both bridges the ration decreases slowly and finally disappears.
  - by random fluctuation of pheromons the choice for a route might converge towards one of both bridges anyway, but randomly!
- analog (symmetrical situation), if pheromons are only place when returning

Double-Bridge Experiment

- **note:** shorter route is found because of both bridges being available in the very beginning, and not differing in their amount of pheromone
- end of the shorter bridge is reached earlier

$\Rightarrow$ different amount of pheromone on both bridges
$\Rightarrow$ self-intensifying process

- **questions:** What if a new even shorter route is added later by changing the environment?
- Will the ants change for the new route?

answer: No! [Goss et al., 1989]

- once a solution route has been established, the ants will stick to it
- proof: by a second bridge experiment: initializing the experiment with only one (longer) path), later adding a second (shorter) one
- most ants go on using the longer path, only few ants change.

Natural and Artificial Ants

reduce the problem to a search for the best path within a weighted graph

- **problem:** self-intensifying cycles (being visited by an ant a cycle becomes even more probable to be again visited by an ant)
- **solution:** labelling routes only after the ant has completed it's whole tour (thus cycles may be removed from the path)
- **problem:** early convergence to a candidate solution found in the very beginning
- **solution:** pheromone evaporation (not of importance in nature)

useful extensions/improvements

- amount of pheromone dependent on the quality of the solution
- considering heuristics when choosing graph edges (e.g. their weights)

Ant Colony Optimization

- **preconditions:** combinatorial optimization problem with constructive method for creating a solution candidate
- **procedure:** solutions are constructed according to a sequence of random choices, where every choice extends a partial solution
- sequence of choices = path in a decision graph (or construction graph)
- ants ought to explore the paths through a decision graph and find the best (shortest, cheapest) one
- ants label the graph edges with pheromone ⇒ other ants will be guided towards promising solutions
- pheromone “evaporates“ after every iteration so once placed it won’t affect the system for too long (“forgetting“ outdated information)

Application to the TSP

- represent the problem by $n \times n$ matrix $\mathbf{D} = (d_{ij})_{1 \leq i,j \leq n}$
- n cities with distances d_{ij} between city i and j
- note: $\mathbf{D}$ may be asymmetrical, but $\forall i \in \{1, \ldots, n\} : d_{ii} = 0$
- pheromone information as $n \times n$ matrix $\Phi = (\phi_{ij})_{1 \leq i,j \leq n}$
- pheromone value $\phi_{ij} (i \neq j)$ indicates the desirability of visiting city j directly after visiting city i (ϕ_{ii} not used)
- there is no need in keeping Φ symmetrical
- initialize all ϕ_{ij} with the same small value (same amount of pheromone on all edges in the beginning)
- ants run Hamilton tour by labelling the edges of the Hamilton tour with pheromone (with the added pheromone value corresponding to the quality of the found solution)

Constructing a solution

- every ant possesses a “memory“ C where indices of not-yet visited cities are stored
- every visited city is removed from the set C
- there is no such memory in the nature!

1. ant is put randomly to a city where it begins its cycle
2. ant chooses not-yet visited city and goes there: in city i an ant chooses a (not-yet visited) city j with a probability of

$$p_{ij} = \frac{\phi_{ij}}{\sum_{k \in C} \phi_{ik}}.$$

3. repeat step 2 until every city has been visited

Updating the pheromone

1. **evaporation**
   all ϕ_{ij} are reduced by a fraction η (evaporation):
   $$\forall i, j \in \{1, \dots, n\} : \phi_{ij} = (1 - \eta) \cdot \phi_{ij}$$
2. **intensifying a constructed solution:**
   pheromone is put on all edges of the constructed solution corresponding to it's quality::
   $$\forall \pi \in \Pi_t : \phi_{\pi(i)\pi((i \bmod n)+1)} = \phi_{\pi(i)\pi((i \bmod n)+1)} + Q(\pi)$$
   Π_t is the amount used for the tour (permutation) constructed during step t, function of quality: e.g. inverse travelling length
   $$Q(\pi) = c \cdot \left(\sum_{i=i}^{n} d_{\pi(i)\pi((i \bmod n)+1)} \right)^{-1}$$
   „The better the solution, the more pheromone is added.“

Travelling Salesman Problem

Algorithm 3 Ant colony optimization for TSP

initialize all elements ϕ_{ij}, $1 \leq i, j \leq n$ of the matrix, with small ϵ
do {
  for each Ant { /* generate candidate solution */
    $C \leftarrow \{1, \ldots, n\}$ /* set of cities to be visited */
    $i \leftarrow$ draw a hometown randomly from C
    $C \leftarrow C \setminus \{i\}$ /* remove it from the set of not-yet visited cities */
    while $C \neq \emptyset$ { /* while there are not-yet visited cities */
      $j \leftarrow$ draw the next city from C with probability p_{ij}
      $C \leftarrow C \setminus \{j\}$ /* remove it from the set of not-yet visited cities */
      $i \leftarrow j$ /* and move there */.
    }
  }
  update matrix of pheromones Φ according to the fitness of the solution
} **while** termination condition is not satisfied

Extensions and Alternatives

- **prefer nearby cities:** (analogical to next neighbor heuristics)
  move from city i to city j with probability

$$p_{ij} = \frac{\phi_{ij}^{\alpha}\tau_{ij}^{\beta}}{\sum_{k \in C} \phi_{ik}^{\alpha}\tau_{ik}^{\beta}}$$

  with C = set of indices of not-yet visited cities and $\tau_{ij} = d_{ij}^{-1}$
- **tend to choosing the best edge:** (greedy)
  with probability p_{exploit} move from city i to city j_{best} with

$$j_{\text{best}} = \arg\max_{j \in C} \phi_{ij} \quad \text{bzw.} \quad j_{\text{best}} = \arg\max_{j \in C} \phi_{ij}^{\alpha}\tau_{ij}^{\beta}$$

  and use p_{ij} with probability $1 - p_{\text{exploit}}$
- **intensify best known tour:** (elitism)
  label it with extra pheromone (e.g. the fraction of additional ants that pass it)

Extensions and Alternatives

ranking based updates

- place pheromone only on edges of last iteration's best solution (and possibly on the best overall solution, too)
- amount of pheromone depends on the rank of the solution

strict elite principles

- place pheromone only on the last iteration's best solution
- place pheromone only on the best solution found so far

Extensions and Alternatives

minimal/maximal amount of pheromone

- set an upper or lower limit of pheromone for the edges

⇒ sets an upper or lower limit for the probability of choosing an edge

⇒ better search space exploration, but might lead to worse convergence

limited evaporation

- pheromone evaporates only on edges, that have been used during this iteration

⇒ better search space exploration

Improving a tour locally

- considering local improvements of a candidate solution is promising: Before updating the pheromone the generated tour is optimized locally. (simple modifications are checked for giving benefit)
- local optimizations include e.g.:
  - **recombination after removing 2 edges** (2-opt) can be seen as “reversing“ a part of a tour
  - **recombination after removing 3 edges** (3-opt) can be seen as “reversing“ two parts of a tour
  - **limited recombination** (2.5-opt)
  - **exchanging neighboring cities**
  - **permutation of neighboring city-triplets**
- apply „expensive“ local optimization only to the best solution found so far (or found during the last iteration)

General Application to Optimization Problems

- **general idea**
  consider the problem as searching a (decision) graph, with the candidate solutions being described by sets of edges (note: these sets are not required to form paths!)

- **general description:** for each problem below we describe
  - nodes and edges of the decision/construction graph
  - constraints
  - significance of the pheromone on edges (and possibly nodes)
  - useful heuristics
  - generation of a candidate solution

- the algorithmic approach is similar to those used for TSP

General Application to Optimization Problems: TSP

- *nodes and edges of the decision/construction graph*: the cities to be visited, and their weighted connections (weights being distance, time, costs)
- *constraints:* visit every city exactly once
- *meaning of pheromone on the edges:* the desirability of visiting city j right after city i
- *useful heuristics:* distances between the cities, prefer close cities
- *generation of a solution candidate:* starting at a randomly chosen city always progress to another, not-yet visited city

General Application to Optimization Problems

General Assignment Problem

assign n tasks to m working units (workmen/machines): minimizing the sum of assignment costs d_{ij} with respect to the maximal capacity ρ_j for given capacity costs r_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$

- every task and every working unit = node of the construction graph (edges are labled with the costs of assignment d_{ij})
- every task has to be assigned to exactly one working unit without exceeding their capacity
- pheromones upon the edges are used for describing the desirability of assigning a task to a working unit
- inverse absolute or relative r_{ij} oder inverse d_{ij}
- choose edges step by step, not necessarily creating a path. Skip edges of tasks that have already been assigned (penalize candidate solutions that violate constraints (e.g. by raising costs))

General Application to Optimization Problems

Knapsack Problem

Choose a subset of maximal value from a set of n objects with a value of w_i, a weight of g_i, a volume of v_i, etc. $1 \leq i \leq n$ with respect to an upper limit for weight, volume etc.

- every object = node within the construction graph, labeled with their value w_i. Edges are not needed
- upper limit for weight, volume etc. has to be respected
- pheromone: assigned only to nodes, describing the desirability of choosing the corresponding object
- ratio between an object and it's relative weight, volume etc. if necessary with respect to the limits
- choose nodes step by step whilst making sure to not exceed the limits

Convergence of the Search

consider "standard behaviour" with the following attributes:

- pheromone evaporating on all edges with a constant factor
- placing pheromone only on the best candidate solution found so far (strict elite principle)
- $\exists$ lower bound $\phi_{\min}$ for pheromone amount on the edges, which is not to be exceeded

- standard procedure converges in probability towards a solution, i.e. the probability for finding a solution goes to 1 with $t \to \infty$

- when the lower bound $\phi_{\min}$ for pheromone values is approaching 0 "sufficiently slow" ($\phi_{\min} = \frac{c}{\ln(t+1)}$ with a number of increments t and a constant c), it can be shown that for $t \to \infty$ the probability for every ant generating a solution will approach 1.

Summary

- swarm and population based algorithms: **heuristics for solving optimization problems**
- purpose: finding a good approximation of the solution
- attempt to reduce the **problem of local optima** (by improving exploration of the search space)
- important: **exchange of information** between individuals (depending on the principle: different types of algorithms)
- **particle swarm optimization**
  - optimization of a function with real agruments
  - exchange of information by watching the neighbors
- **ant colony optimization**
  - search for best routes (abstract: within a decision graph)
  - exchange of information: manipulation of the environment (stigmergy)

Outline

1. Swarm- And Population-Based Optimization

2. Organic Computing

3. Summary

Motivation

In the future independent systems will

- be able to communicate,
- adapt to their environment automatically,
- be used in many different fields.

Examples for such systems:

- administration of peer-to-peer-networks
- learning traffic controls
- robotic area scouts
- automation of processes in factories
- management of renewable energy resources
- self-repairing faulty systems in automobiles

Organic Computing

- goal: independent organization, configuration and repair of those complex IT systems
- solution: *Organic Computing*
  - adapt to (environmental) changes,
  - inspired by patterns in nature

problems:

- Controlling these systems becomes increasingly complex as they develop an emergent behaviour, i.e. a behaviour that they have not shown before
- emergence can be favorable, if the system is able to react correctly. Otherwise it might be fatal.
  Consider a learning traffic control system that switches all traffic lights to green because of noticing that letting cars pass is reducing traffic jams...

Organic Computing: More Than Just Optimization

- ants take on roles such as soldiers or workers
- e.g. if the number of soldiers drops below a certain threshold, some workers will become soldiers

$\Rightarrow$ sensors could take on new tasks when others fail

- systems could solve tasks easier and more efficiently after having performed on them several times before

- since 2004 the DFG supports fields regarding the topic Organic Computing
- literature: [Würtz, 2008] also online at `http://www.springerlink.com/content/978-3-540-77656-7`

Outline

Genetic and Evolutionary Algorithms

representation: (classical) $\{0,1\}^L$ with fixed length L (also $\mathbb{R}^L$ and $\mathcal{S}_n$, decoding)

mutation: bit flipping, uniformly distributed real-valued mutation, special operations for permutation

recombination: k-point- and uniform crossover, arithmetical crossover, order-based recombination

selection: parental selection, fitnessproportional or tournament selection

population: mid-sized populations

features: theoretical basis in Schema Theorem (*next up in the lecture*)

Local Search

representation: arbitrary

mutation: arbitrary

recombination: none

selection: improvements always, degradiation with a certain probability

population: one individual

features: early convergence is a central problem

Tabu Search

representation: close to phenotype

mutation: non-invertable because of Tabu-lists

recombination: none

selection: best individual

population: one parental, several children

features: best found individual is stored additionally

Memetic Algorithm

representation: arbitrary

mutation: combined with local search

recombination: arbitrary

selection: arbitrary

population: arbitrary

features: arbitrary

Differential evolution

representation: $\mathbb{R}^L$

mutation: mixed operator

recombination: mixed operator

selection: child replaces parental if it is superior

Population: small/mid-sized

features: mutation makes use of population information

Scatter Search

representation: $\mathbb{R}^L$ and others

mutation: none

recombination: subset operator and combination

selection: selection of the best

population: mid-sized

features: many variants, deterministic procedure

Cultural Algorithm

representation: $\mathbb{R}^L$ and others

mutation: makes use of conviction space

recombination: none

selection: environmental selection

population: mid-sized

features: conviction space stores prescriptive and situation-based information

Population-Based Incremental Learning

representation: $\{0, 1\}^L$

mutation: changing the population statistics

recombination: implicitly

selection: best child individual enters statistics

population: is replaced by population statistics

features: whenever individuals are needed, they are drawn from the statistics

Ant colony optimization

representation: several different

mutation: every ant generates one solution candidate

recombination: none

selection: quality determines influence on global pheromones

population: quantity of ants during one iteration

features: global amount of pheromones represents candidate solutions similar to statistics in PBIL

Particle Swarm Optimization

representation: $\mathbb{R}^L$

mutation: based on lethargy and neighbors

recombination: none

selection: based on the best (population/own memory)

population: small/mid-sized

features: synchronously searching the search space

literature for this lecture

Dorigo, M. and Stützle, T. (2004).
Ant Colony Optimization.
MIT Press, Cambridge, MA, USA.

Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. M. (1989).
Self-organized shortcuts in the argentine ant.
Naturwissenschaften, 76:579–581.

Kennedy, J. and Eberhart, R. (1995).
Particle swarm optimization.
In *Proceedings of the IEEE International Conference on Neural Networks*, page 1942–1948, Perth, Australia. IEEE Press.

Würtz, R. P., editor (2008).
Organic Computing.
Understanding Complex Systems. Springer Berlin / Heidelberg.