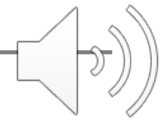
Introduction to Metaheuristics

- *Heuristics*
  - "to find" (from ancient Greek "εὑρίσκειν")

- *Meta-*
  - An abstraction from another concept
    - beyond, in an upper level
    - used to complete or add to
  - E.g., meta-data = "data about data"

- *Metaheuristics*
  - "A heuristic around heuristics"

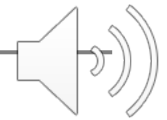


Introduction to Metaheuristics

- *Heuristics*
  - "to find" (from ancient Greek "εὑρίσκειν")

- *Meta-*
  - An abstraction from another concept
    - beyond, in an upper level
    - used to complete or add to
  - E.g., meta-data = "data about data"

- *Metaheuristics*
  - "A heuristic around heuristics"

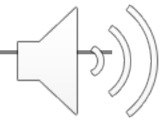


Introduction to Metaheuristics

- *Heuristics*
  - "to find" (from ancient Greek "εὑρίσκειν")

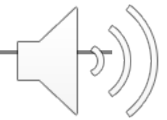
- *Meta-*
  - An abstraction from another concept
    - beyond, in an upper level
    - used to complete or add to
  - E.g., meta-data = "data about data"

- *Metaheuristics*
  - "A heuristic around heuristics"



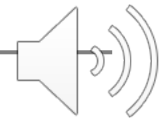
Introduction to Metaheuristics

- Metaheuristics…
  - …usually incorporate mechanisms to avoid getting trapped in confined areas of the search space
  - …usually are non-deterministic
  - …are not problem specific (but their subordinate heuristics are)
  - …may use some form of memory to better guide the search



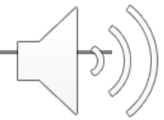
Introduction to Metaheuristics

- Some popular frameworks
  - Genetic Algorithms
  - Simulated Annealing
  - Tabu Search
  - Scatter Search
  - Ant Colony Optimization
  - Particle Swarm Optimization
  - Iterated Local Search
  - Variable Neighborhood Search
  - Adaptive Memory Programming
  - …



Introduction to Metaheuristics

- Metaheuristics…
  - …can address both discrete- and continuous-domain optimization problems
  - …are strategies that "guide" the search process
  - …range from simple local search procedures to complex adaptive learning processes

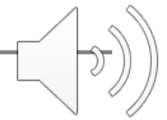


Introduction to Metaheuristics

- Metaheuristics…
  - …efficiently explore the search space in order to find good (near-optimal) feasible solutions
  - …provide no guarantee of global or local optimality
  - …are agnostic to the unexplored feasible space (i.e., no "bound" information)
  - …lack a metric of "goodness" of solution (often stop due to an external time or iteration limit)
  - …are not based on some algebraic model

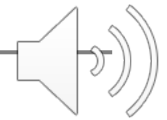
  unlike exact methods!

  - …are often used in conjunction with an exact method



Introduction to Metaheuristics

- Metaheuristics…
  - …is a relatively new field (started in the ‘80s or so)
    - becomes possible because we can now afford vast amounts of computation
    - inspired from “AI,” rather than “pure math”
    - lack of theoretical “rigor” (no proofs, theorems, etc. for people to pursue)



Metaheuristics classification

Trajectory-based
(S-metaheuristics)

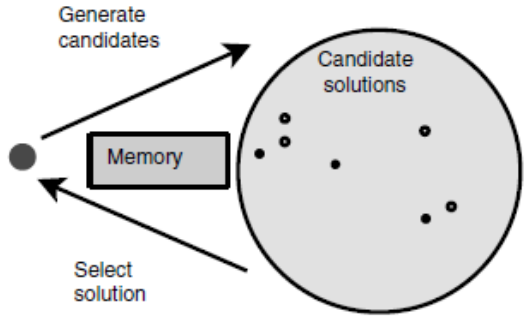


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.

Population-based
(P-metaheuristics)

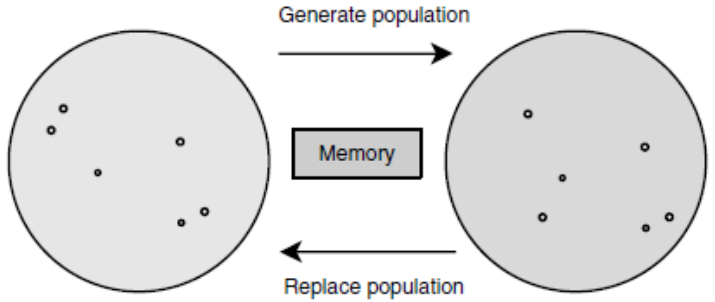
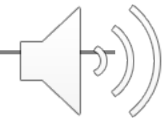


FIGURE 3.1 Main principles of P-metaheuristics.

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.



TRAJECTORY METHODS

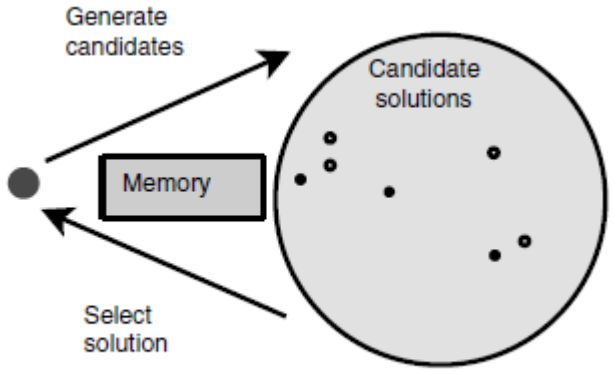
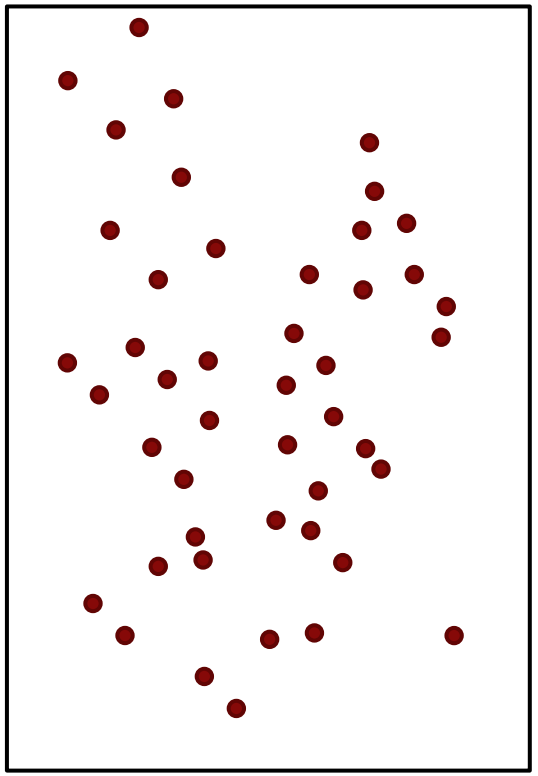


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



TRAJECTORY METHODS

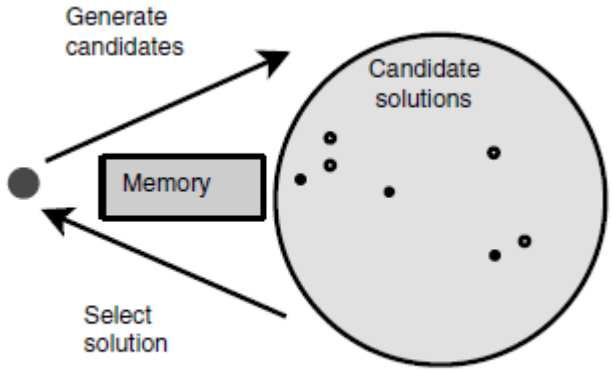
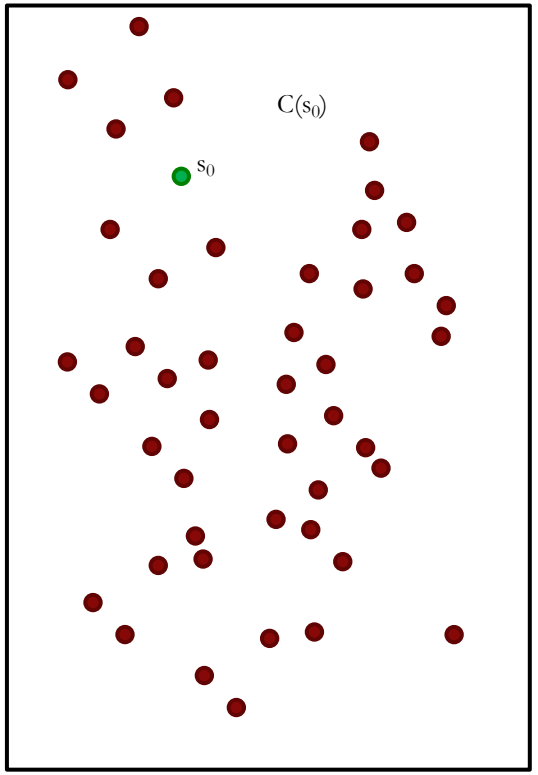


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



TRAJECTORY METHODS

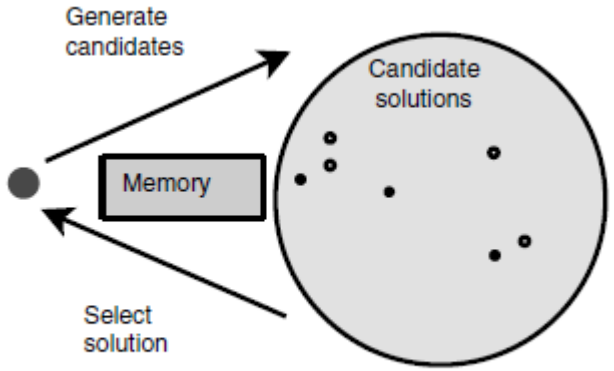
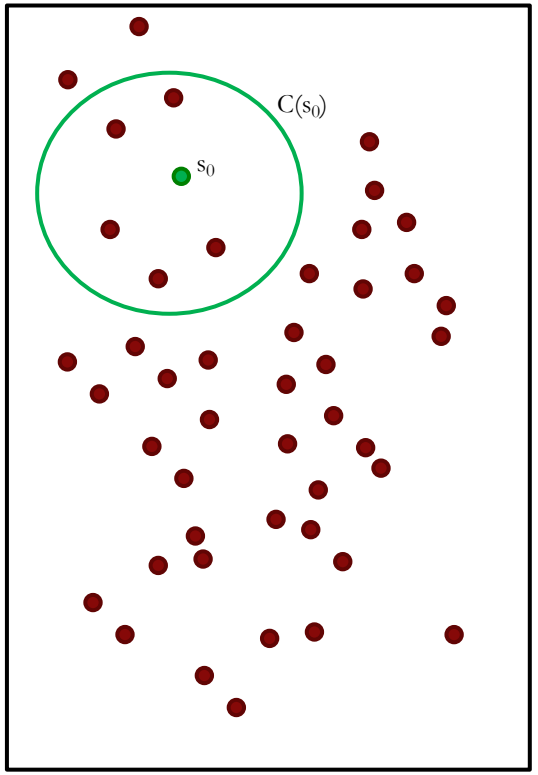


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



TRAJECTORY METHODS

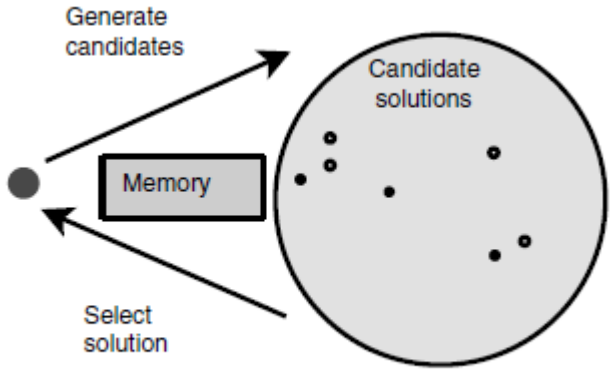
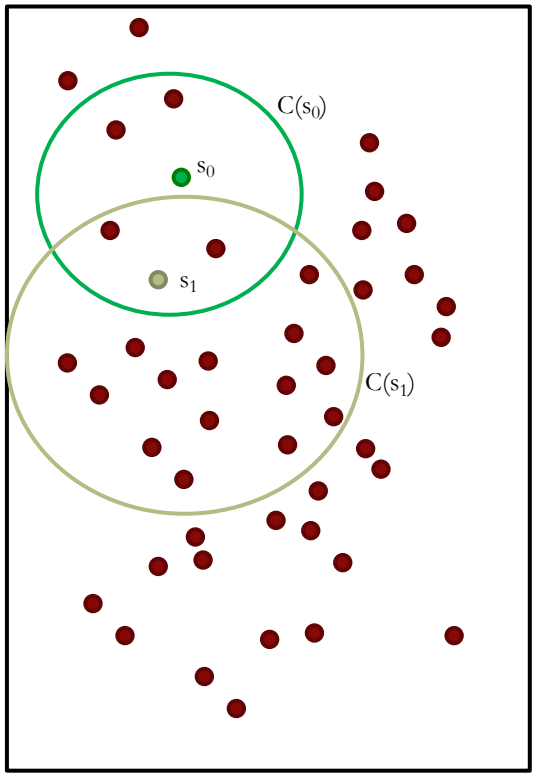


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



TRAJECTORY METHODS

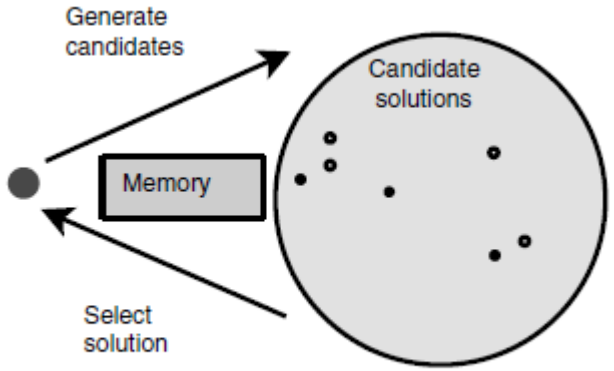
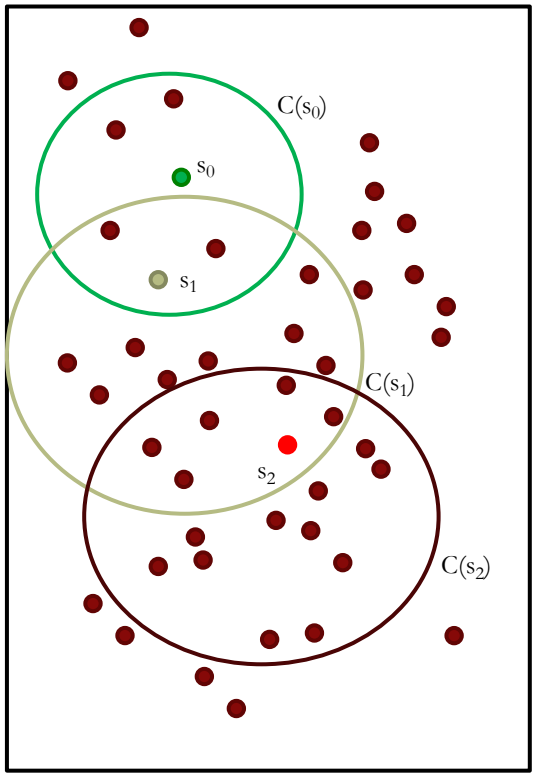


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



Stop because of no improvement in region $C(s_2)$

TRAJECTORY METHODS

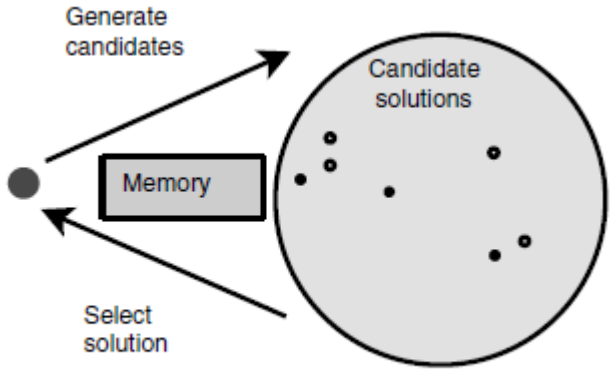
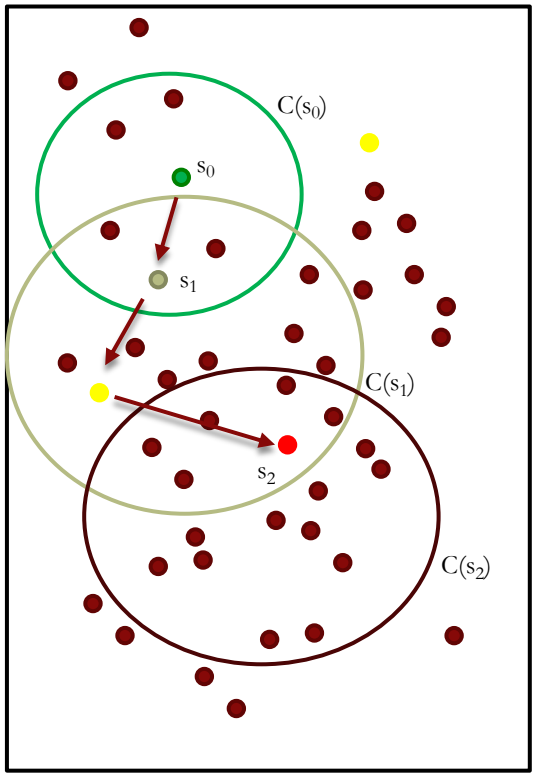


FIGURE 2.1 Main principles of single-based metaheuristics.

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.



Turns out two global optima in this problem, but none was identified

- One was missed during search of region $C(s_1)$
- One was far away from searched space

POPULATION-BASED METHODS

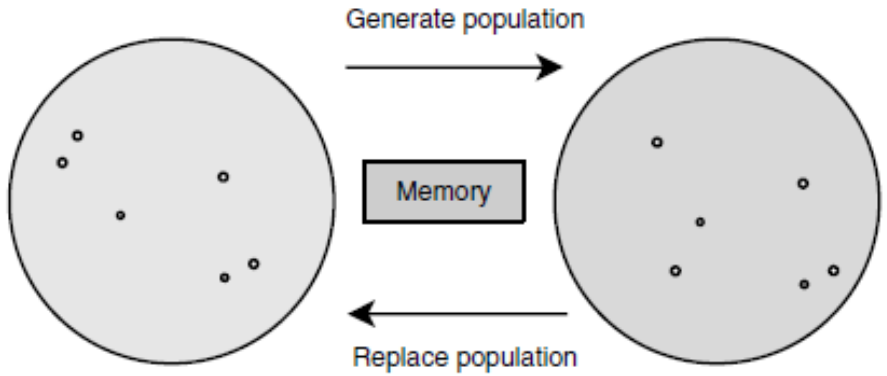
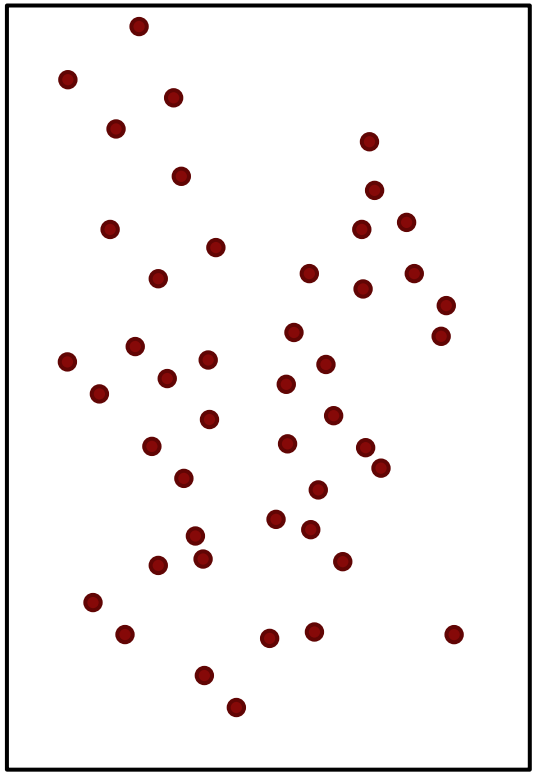


FIGURE 3.1 Main principles of P-metaheuristics.

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.



POPULATION-BASED METHODS

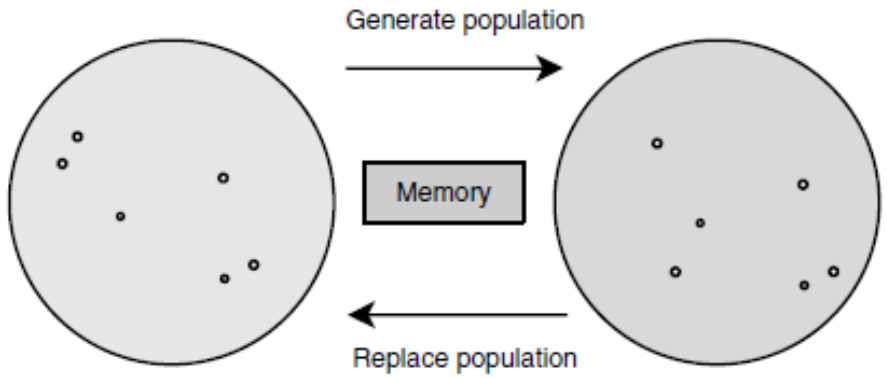
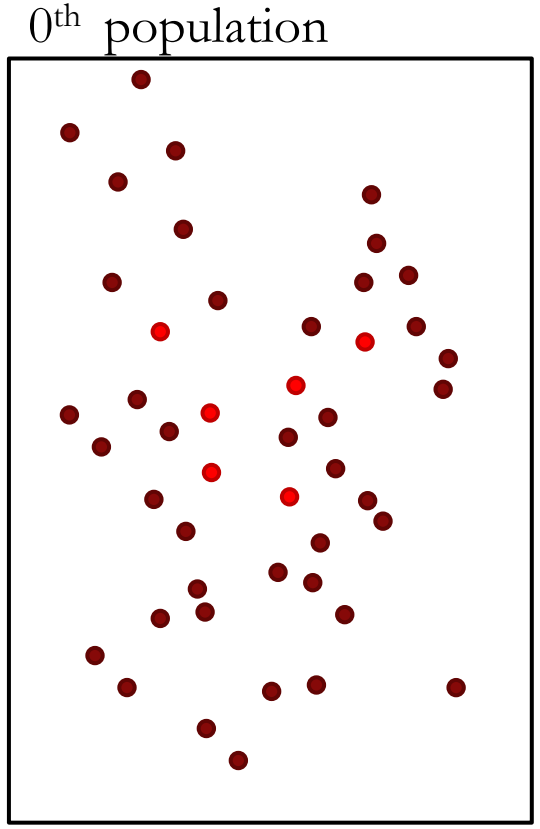


FIGURE 3.1 Main principles of P-metaheuristics.

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.



POPULATION-BASED METHODS

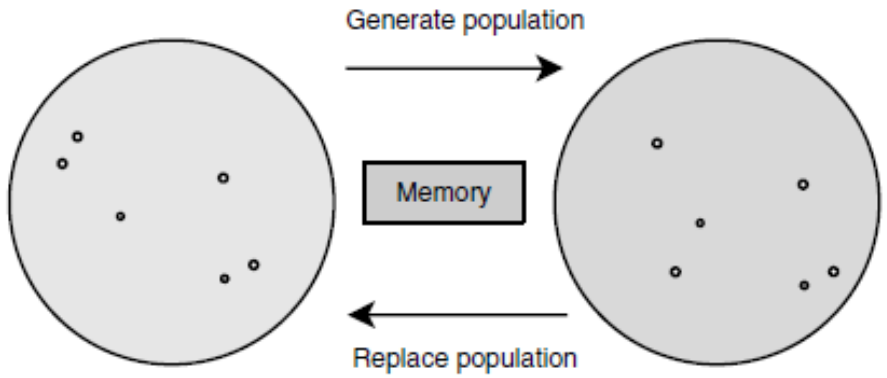
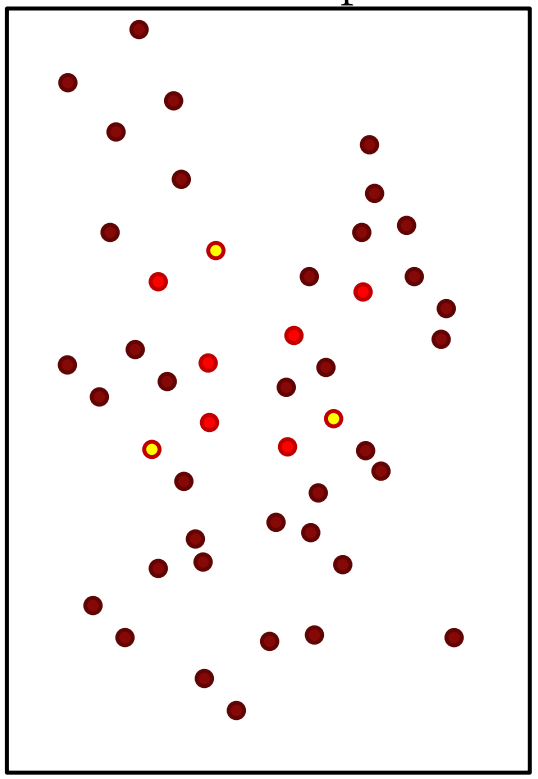


FIGURE 3.1 Main principles of P-metaheuristics.

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.

Get some new points



POPULATION-BASED METHODS

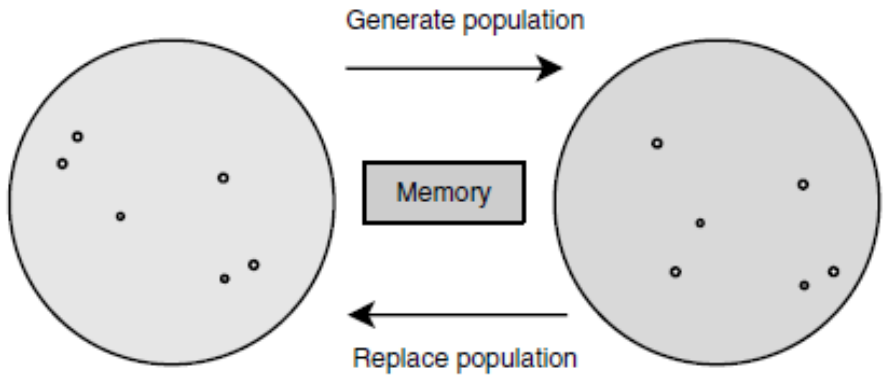
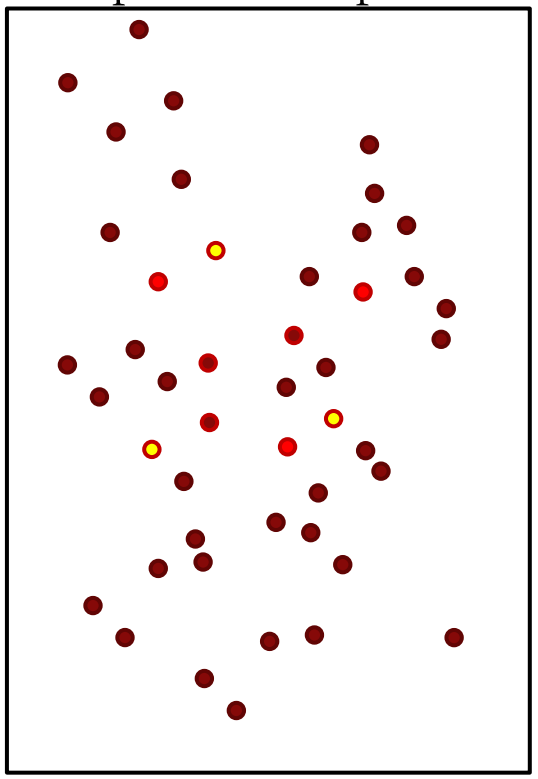


FIGURE 3.1 Main principles of P-metaheuristics.

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.

Drop some old points



POPULATION-BASED METHODS

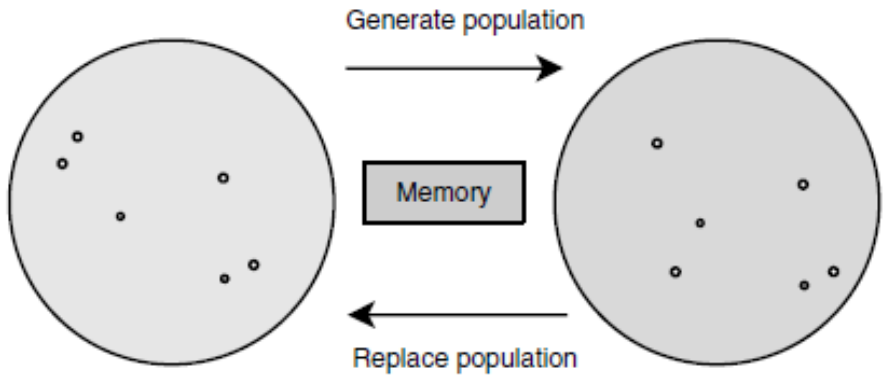
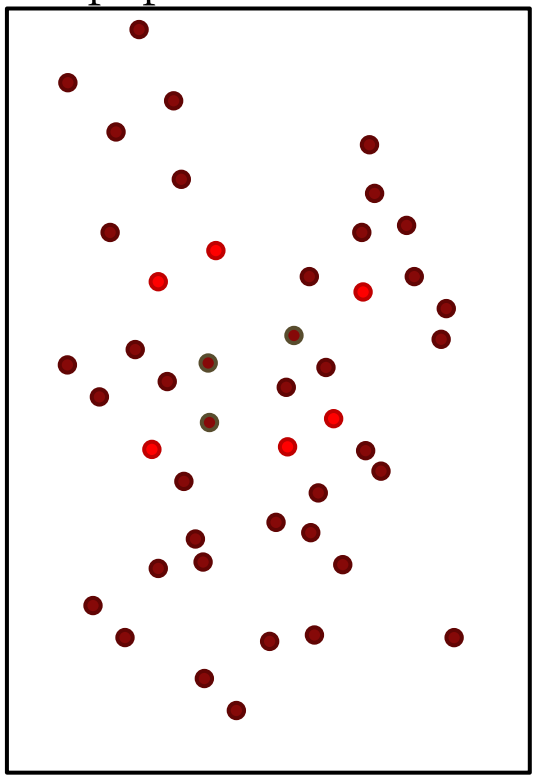


FIGURE 3.1 Main principles of P-metaheuristics.

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.

1st population



POPULATION-BASED METHODS

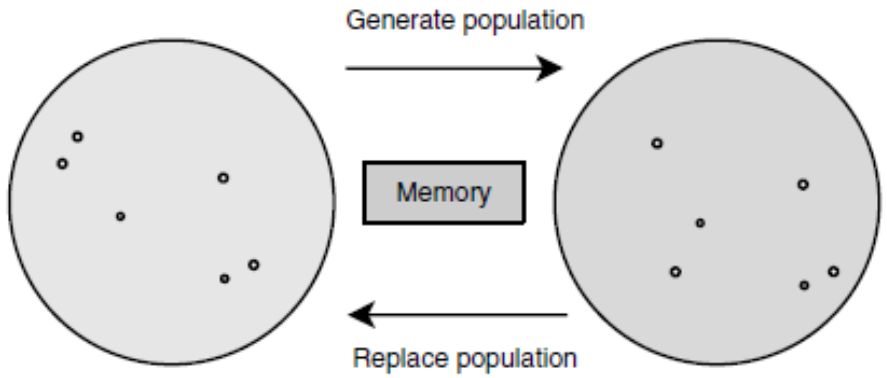
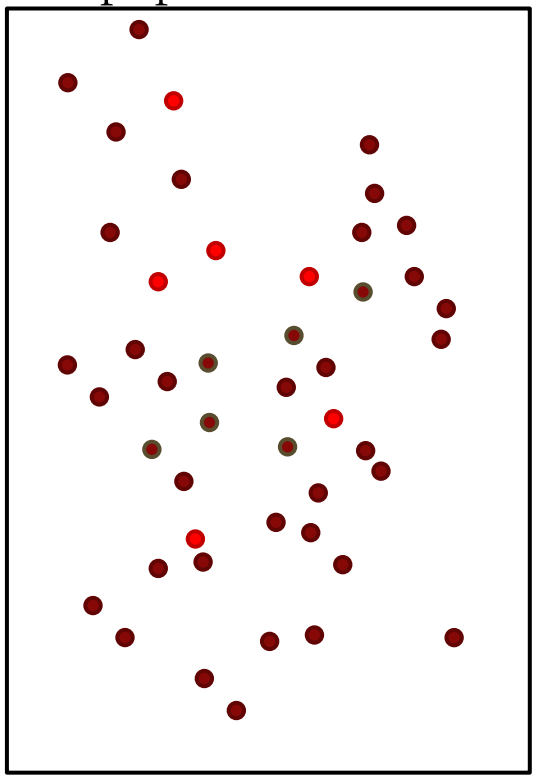


FIGURE 3.1 Main principles of P-metaheuristics.

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.

2nd population



POPULATION-BASED METHODS

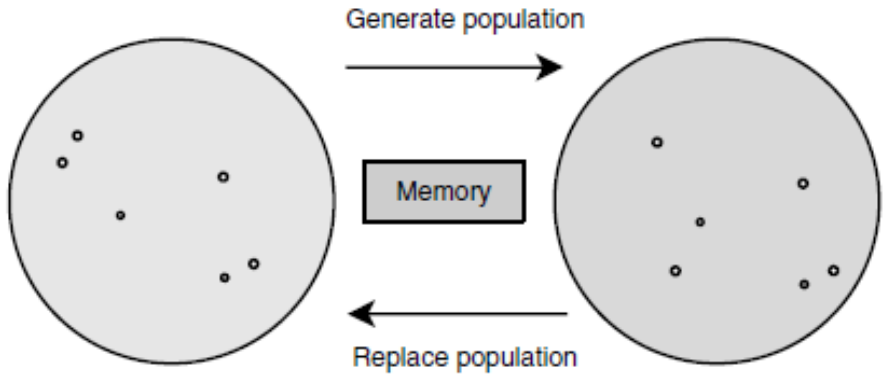
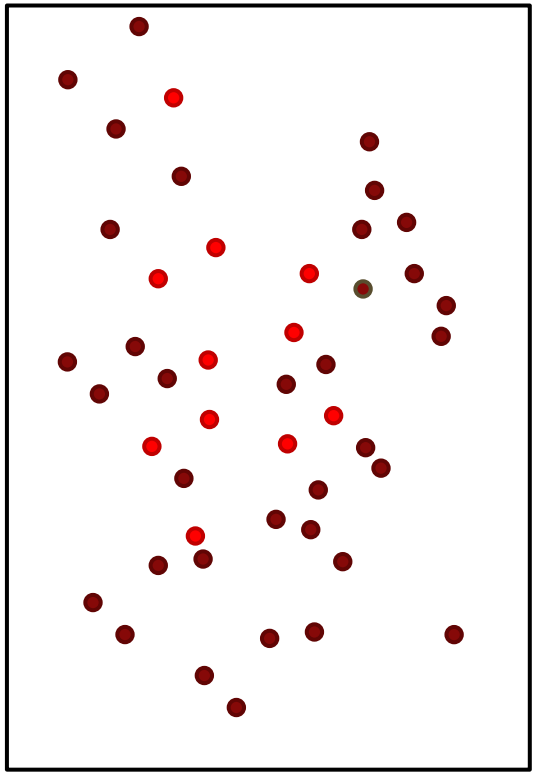


FIGURE 3.1 Main principles of P-metaheuristics.

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.



Again, optimum may or may not have been sampled

- Typically, the incumbent always remains in the population, so need only focus on last generation

POPULATION-BASED METHODS

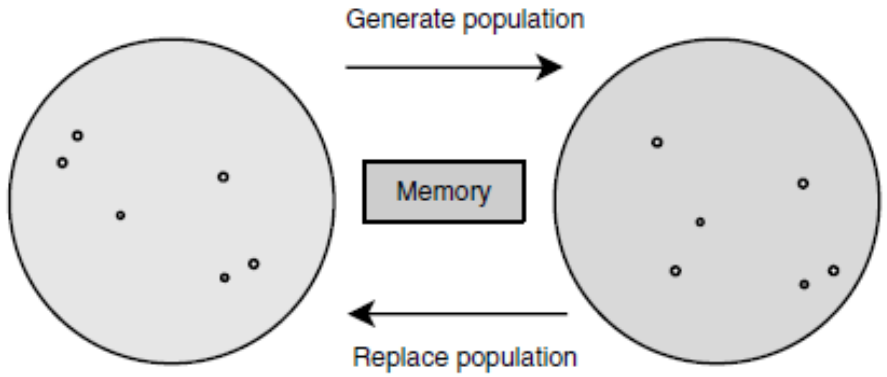
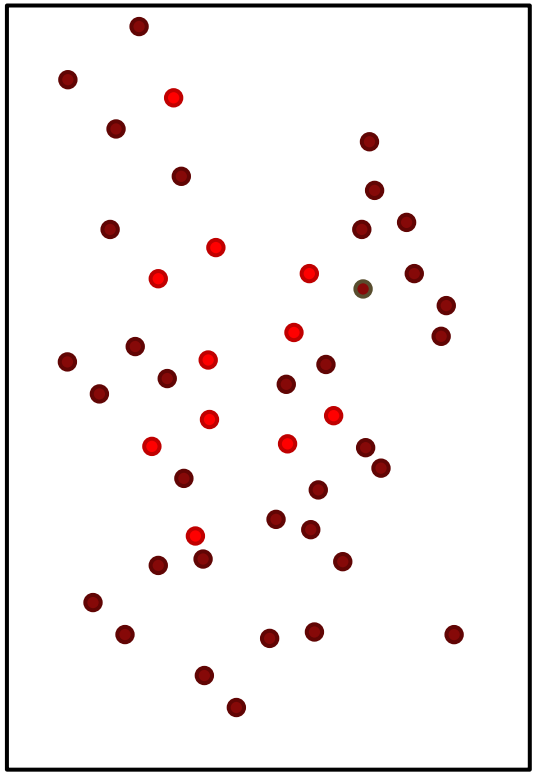


FIGURE 3.1 Main principles of P-metaheuristics.

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.



Again, optimum may or may not have been sampled

- Typically, the incumbent always remains in the population, so need only focus on last generation