IE552

Lecture Notes 6

Ant Colony, Particle Swarm, Scatter Search, and Variable Neighborhood Search

Swarm Intelligence

- Algorithms inspired from the collective behavior of species such as ants, bees, wasps, termite, fish, and birds are referred as swarm intelligence algorithms.
- Swarm intelligence originated from the social behavior of those species that compete for foods.
- The main characteristics of swarm-intelligence-based algorithms are simple and nonsophisticated agents, they *cooperate* by an indirect communication, and do movements in the decision space.
- Among the most successful swarm intelligence inspired optimization algorithms are ant colony and particle swarm optimization.

Ant Colony Optimization

- The basic idea in ant colony optimization algorithms (ACO) is to imitate the cooperative behavior of real ants to solve optimization problems.
- ACO metaheuristics have been proposed by M. Dorigo (1992).
- They can be seen as multiagent systems in which each single agent is inspired by the behavior of a real ant.
- Traditionally, ACO have been applied to combinatorial optimization problems and they have achieved widespread success in solving different problems (e.g., scheduling, routing, assignment)

Ant Colony Optimization

- It is quite interesting that ants, using collective behavior, perform complex tasks such as transportation of food and finding shortest paths to the food sources.
- ACO algorithms mimic the principle that using very simple communication mechanism, an ant colony is able to find the shortest path between two points.

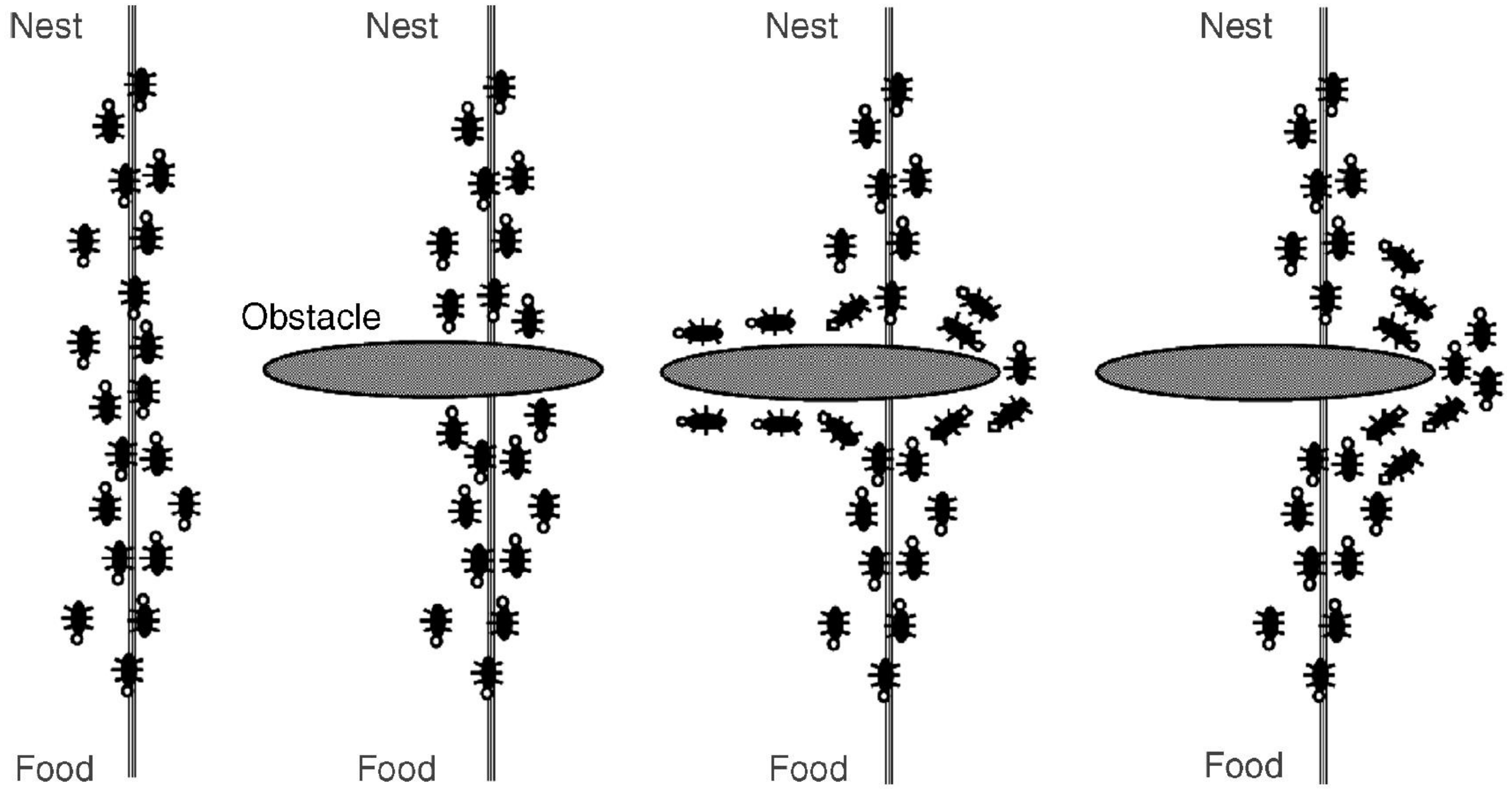


FIGURE 3.32 Inspiration from an ant colony searching an optimal path between the food and the nest.

Ant Colony Optimization

- Figure 3.32 illustrates an experiment done by Goss et al. with a real colony of Argentine ants (Iridomyrmex humilis).
- Notice that those ants cannot see very well. The colony has access to a food source linked by two paths to the colony’s nest.
- During their trips, a chemical trail (pheromone) is left on the ground. The pheromone is an olfactive and volatile substance.
- The role of this trail is to guide the other ants toward the target point. The larger the amount of pheromone on a particular path, the larger the probability that the ants will select the path.
- For a given ant, the path is chosen according to the smelt quantity of pheromone.

Ant Colony Optimization

- Furthermore, this chemical substance has a decreasing action over time (evaporation process) and the quantity left by one ant depends on the amount of food (reinforcement process).
- As shown in Fig. 3.32, when facing an obstacle, there is an equal probability for every ant to choose the left or right path.
- As the left trail is shorter than the right one and so requires less travel time, the ant will end up leaving a higher level of pheromone.
- The more the ants take the left path, the higher the pheromone trail.
- Hence, there is an emergence of the shortest path.

Algorithm 3.12 Template of the 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.
```

Solution construction

- The construction of solutions is done according to a probabilistic state transition rule.
- Artificial ants can be considered as stochastic greedy procedures that construct a solution in a probabilistic manner by adding solution components to partial ones until a complete solution is derived.
- The target optimization problem can be seen as a decision graph (or construction graph) where an ant will construct a path.
- Usually, this iterative process takes into account <u>pheromone trails and problem-dependent heuristic information</u>.
- **Pheromone trails:**
  - They memorize the characteristics of “good” generated solutions, which will guide the construction of new solutions by the ants.
  - They change dynamically during the search to reflect the acquired knowledge.
  - It represents the memory of the whole ant search process.

Pheromone update

- The update of the pheromone is carried out using the generated solutions. A global pheromone updating rule is applied in two phases:
- An *evaporation phase* where the pheromone trail decreases automatically. Each pheromone value is reduced by a fixed proportion:

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \quad \forall i, j \in [1, n]$$

where ρ ∈[0, 1] represents the reduction rate of the pheromone.

- The goal of the evaporation is to avoid for all ants a premature convergence toward “good” solutions and then to encourage the diversification in the search space (exploration).

Pheromone update

- *Reinforcement phase* where the pheromone trail is updated according to the generated solutions. Three different strategies may be applied
  - **Online step-by-step pheromone update:** The pheromone trail ij is updated by an ant at each step of the solution construction.
  - **Online delayed pheromone update:** The pheromone update of is applied once an ant generates a complete solution. For instance, each ant will update the pheromone information with a value that is proportional to the quality of the solution found.
  - **Off-line pheromone update:** The pheromone train update is applied once *all* ants generate a complete solution. Could be quality based, rank based, worst or elitist pheromone update

Algorithm 3.13 Ant colony algorithm for the TSP problem (ACO–TSP).

```
Initialize the pheromone information ;
Repeat
   For each ant Do
      Solution construction using the pheromone trails:
         $S = \{1, 2, \ldots, n\}$ /* Set of potentially selected cities */
         Random selection of the initial city $i$ ;
         Repeat
            Select new city $j$ with probability $p_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}}$ ;
            $S = S - \{j\}$ ; $i = j$ ;
         Until $S = \emptyset$
   End For
   Update the pheromone trail:
      For $i, j \in [1, n]$ Do
         $\tau_{ij} = (1 - \rho)\tau_{ij}$ /* Evaporation */ ;
      For $i \in [1, n]$ Do
         $\tau_{i\pi(i)} = \tau_{i\pi(i)} + \Delta$ /* $\pi$: best found solution */ ;
Until Stopping criteria
Output: Best solution found or a set of solutions.
```

Ant Colony for TSP

- **Pheromone trails:** A pheromone τ_{ij} will be associated with each edge (i, j) of the graph G.
- The pheromone information can be represented by an n × n matrix where each element τ_{ij} of the matrix expresses the desirability to have the edge (i, j) in the tour.
- The pheromone matrix is generally initialized by the same values.
- During the search, the pheromone will be updated to estimate the utility of any edge of the graph.

Ant Colony for TSP

- **Solution construction:** Each ant will construct a tour in a stochastic way. Given an initial arbitrary city i, an ant will select the next city j with the probability

$$p_{ij} = \frac{\tau_{ij}}{\sum_{k \in S} \tau_{ik}}, \quad \forall j \in S$$

- where the set S represents the not yet visited solutions of the graph G.
- The ants may use a randomly selected initial city in the construction phase.

Ant Colony for TSP

- The additional problem-dependent heuristic is defined by considering the values η_{ij} equal to $1/d_{ij}$ where d_{ij} represents the distance between the cities i and j.
- The higher the heuristic value η_{ij} , the shorter the distance d_{ij} between cities i and j.
- Computing the decision transition probabilities, p_{ij} is performed as follows:

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}}, \quad \forall j \in S$$

Ant Colony for TSP

- Then, the pheromone update procedure has to be specified. For instance, each ant will increment the pheromone associated with the selected edges in a manner that is proportional to the quality of the obtained tour π:

$$\tau_{i\pi(i)} = \tau_{i\pi(i)} + \Delta, \qquad \forall i \in [1, n]$$

- where $\Delta = 1/f(\pi)$ (1/length of the tour)
- Then, good tours will emerge as the result of the cooperation between ants through the pheromone trails.

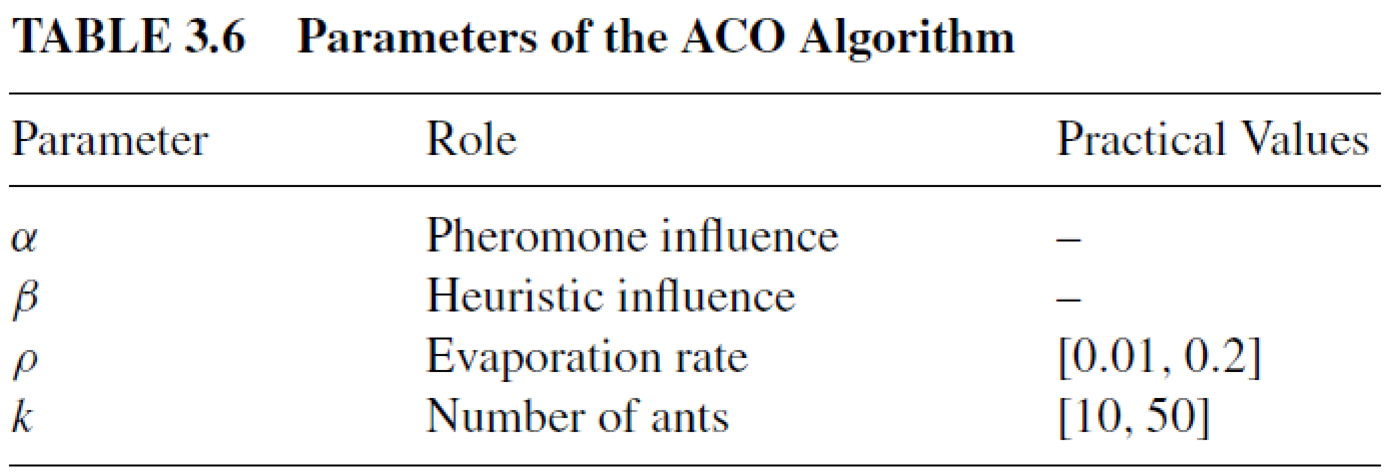


TABLE 3.6 Parameters of the ACO Algorithm

Parameter	Role	Practical Values
α	Pheromone influence	–
β	Heuristic influence	–
ρ	Evaporation rate	[0.01, 0.2]
k	Number of ants	[10, 50]

- For a small example of TSP check the following link

https://www.youtube.com/watch?v=783ZtAF4j5g&t=55s&ab_channel=AliMirjalili

Particle Swarm Optimization

- It is another stochastic population-based metaheuristic inspired from swarm intelligence.
- It mimics the social behavior of natural organisms such as bird flocking and fish schooling to find a place with enough food.
- Indeed, in those swarms, a coordinated behavior using local movements emerges without any central control.
- Originally, PSO has been successfully designed for continuous optimization problems.

Particle Swarm Optimization

- In the basic model, a swarm consists of N particles flying around in a D dimensional search space.
- Each particle i is a candidate solution to the problem, and is represented by the vector x_i in the decision space.
- A particle has its own position and velocity, which means the flying direction and step of the particle (Fig. 3.33).

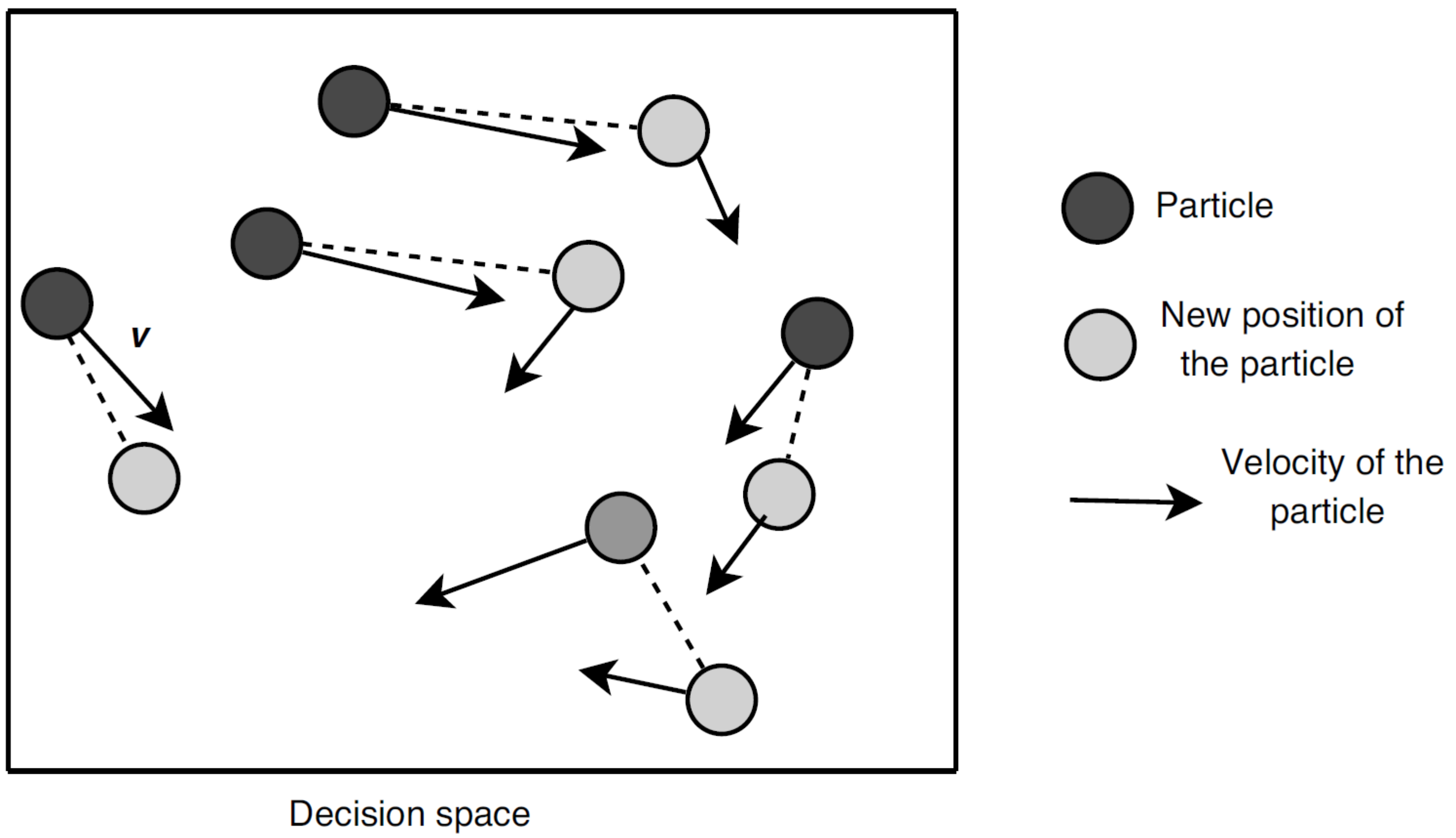


FIGURE 3.33 Particle swarm with their associated positions and velocities. At each iteration, a particle moves from one position to another in the decision space. PSO uses no gradient information during the search.

Particle Swarm Optimization

- Optimization takes advantage of the cooperation between the particles.
- The success of some particles will influence the behavior of their peers.
- Each particle successively adjusts its position x_i toward the global optimum according to the following two factors:
  - the best position visited by itself (pbesti) denoted as $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$and
  - the best position visited by the whole swarm (gbest) (or lbest, the best position for a given subset of the swarm) denoted as $p_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$.
- The vector $(p_g - x_i)$ represents the difference between the current position of the particle i and the best position of its neighborhood.

Particles Neighborhood

- A neighborhood must be defined for each particle.
- This neighborhood denotes the social influence between the particles. There are many possibilities to define such a neighborhood. Traditionally, two methods are used:
  - **gbest Method:** In the global best method, the neighborhood is defined as the whole population of particles.
  - **lbest Method:** In the local best method, a given topology is associated with the swarm. Hence, the neighborhood of a particle is the set of directly connected particles. The neighborhood may be empty in which the articles are isolated

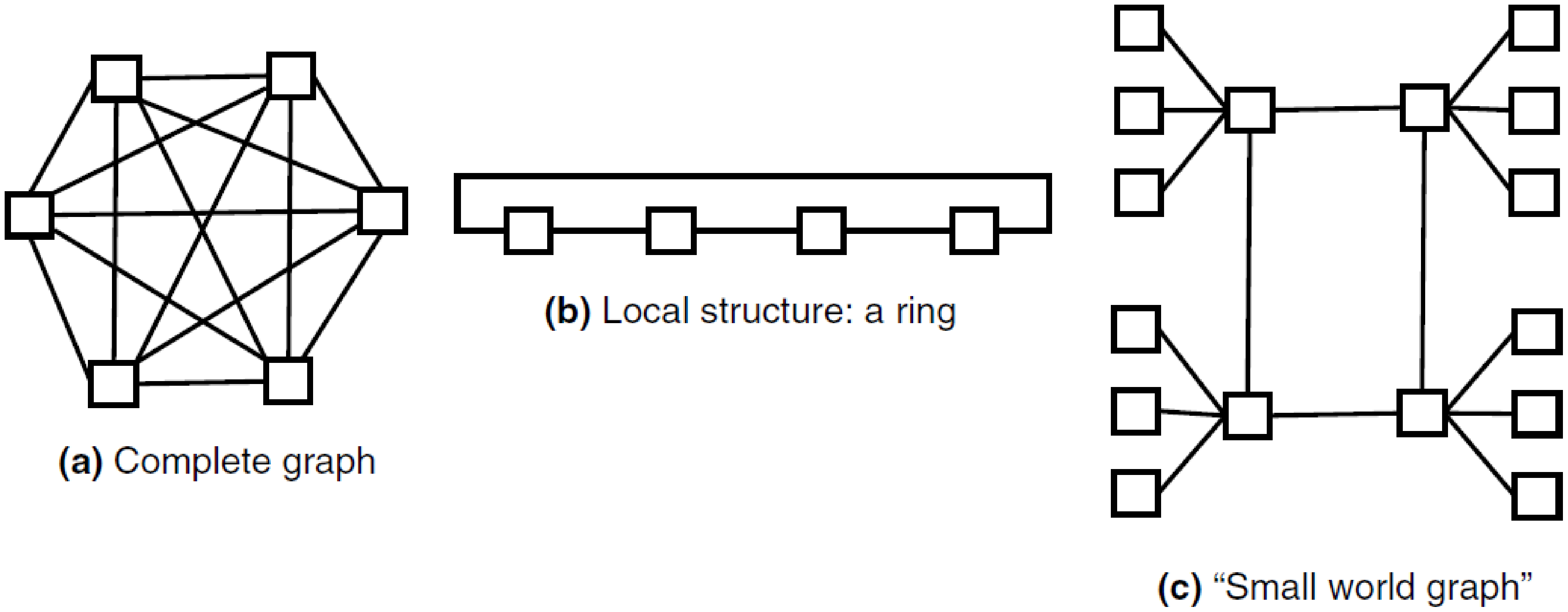


FIGURE 3.34 Neighborhood associated with particles. (a) gbest Method in which the neighborhood is the whole population (complete graph). (b) lbest Method where a noncomplete graph is used to define the neighborhood structure (e.g., a ring in which each particle has two neighbors). (c) Intermediate topology using a small world graph.

Vectors

A particle is composed of three vectors:

- The *x*-vector records the current position (location) of the particle in the search space.
- The *p*-vector records the location of the best solution found so far by the particle.
- The *v*-vector contains a gradient (direction) for which particle will travel in if undisturbed.
- Two fitness values: The *x*-fitness records the fitness of the *x*-vector, and the *p*-fitness records the fitness of the *p*-vector.

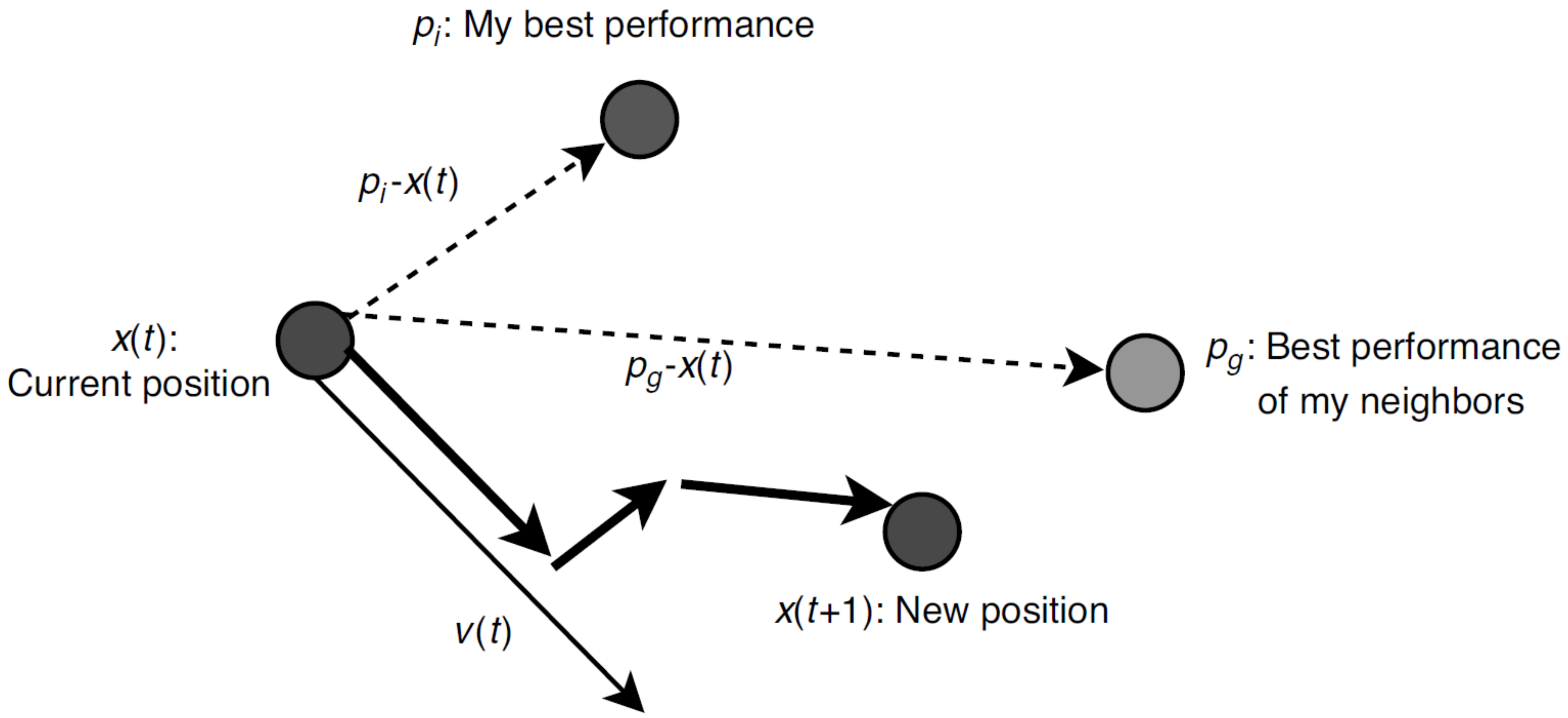


FIGURE 3.35 Movement of a particle and the velocity update.

Algorithm 3.14 Template of the particle swarm optimization 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
```

- https://www.youtube.com/watch?v=DzcZ6bP4FGw&ab_channel=ChurchillCompSciTalks for simulations

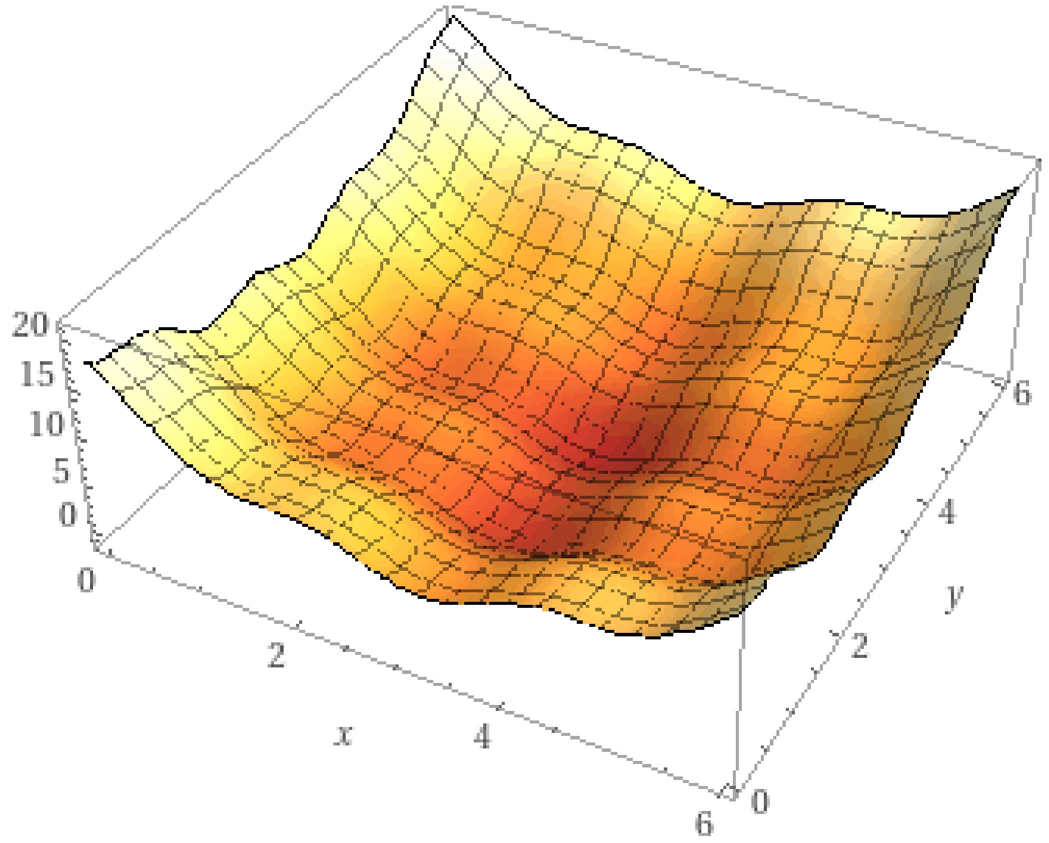
Particle Swarm Optimization Continued

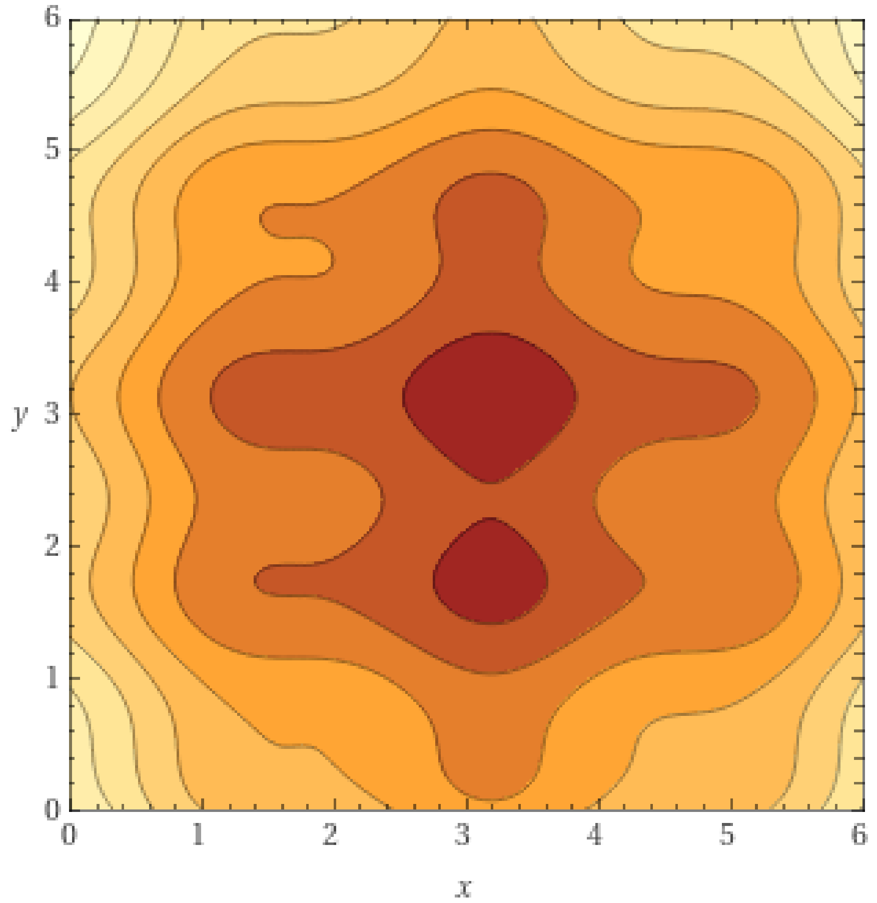
- **Particle Swarm Optimization** was proposed by Kennedy and Eberhart in 1995.
- “a school of fish or a flock of birds that moves in a group can profit from the experience of all other members”
- We simulate the movement of a flock of birds, each bird helps us find the optimal solution in a high-dimensional solution space and the best solution found by the flock becomes the best solution.
- PSO

Example

We would like to minimize the following function:

$$f(x,y)(x-3.14)^2 + (y-2.72)^2 + \sin(3x+1.41) + \sin(4y-1.73)$$





- We start with a number of random points on the plane (call them **particles**) and let them look for the minimum point in random directions.
- At each step, every particle should search around the minimum point it ever found as well as around the minimum point found by the entire swarm of particles.
- After certain iterations, we consider the minimum point of the function as the minimum point ever explored by this swarm of particles.

- At iteration t, let $x_i(t)$ be the location of particle i, and $v_i(t)$ be its velocity
- We update the velocity using the current best of particle i and the global best solution

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

- Then we move the next location

$$x_i(t) = x_i(t-1) + v_i(t)$$

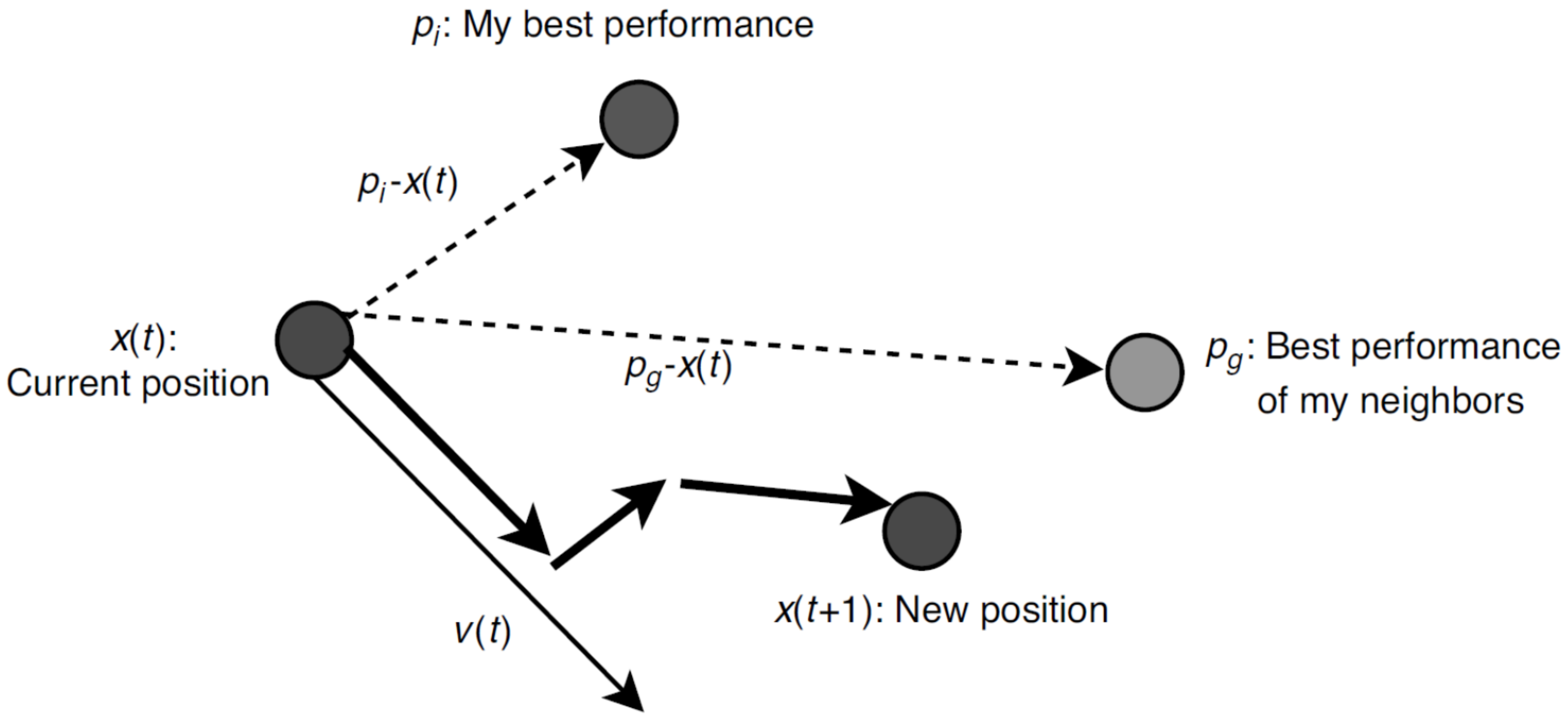


FIGURE 3.35 Movement of a particle and the velocity update.

Code and animation for minimizing the following function with pso

$$f(x,y)(x-3.14)^2 \; + \; (y-2.72)^2 + \sin(3x+1.41) + \sin(4y-1.73)$$

https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

PSO for Discrete Problems

- Unlike ACO algorithms, PSO algorithms are applied traditionally to continuous optimization problems.
- Some adaptations must be made for discrete optimization problems. They differ from continuous models in
  - Mapping between particle positions and discrete solutions:

    Many discrete representations such as binary encodings and permutations can be used for a particle position. For instance, in the binary mapping, a particle is associated with an n-dimensional binary vector.
  - Velocity models:

    The velocity models may be real valued, stochastic, or based on a list of moves. In stochastic velocity models for binary encodings, the velocity is associated with the probability for each binary dimension to take value 1.

PSO for Discrete Problems

- Velocity models for discrete optimization problems have been generally inspired from mutation and crossover operators of EAs.

Example 3.21 Geometric PSO.

- Based on the geometric framework of recombination operators.
- The location of each particle i is represented by a binary vector $x_i = (x_{i1}, x_{i2}, \dots, x_{iN})$ where each element x_{ij} (with $j \in \{1, N\}$) takes a binary value 0 or 1.
- The key issue of the GPSO is the concept of particle movement. In this approach, instead of using the notion of velocity, a three-parent mask-based crossover (3PMBCX) operator is applied to each particle to “move” it.

PSO for Discrete Problems

Example 3.21 Geometric PSO.

- 3PMBCX operator, given three parents a, b, and c in $\{0,1\}^n$, generates

randomly a crossover mask of length n with symbols from the alphabet a, b, and c.

- Then, the offspring is constructed by filling each element with the value from the parent appearing in the crossover mask at the corresponding position.
- For particle i, three parents take part in the 3PMBCX operator: the current position x_i, the social best position g_i, and the historical best position found h_i of this particle
- The weight values w_1, w_2, and w_3 indicate for each element in the crossover mask the probability of having values from the parents x_i, g_i, or h_i, respectively.
- w_1, w_2, and w_3 are nonnegative and add up to 1.

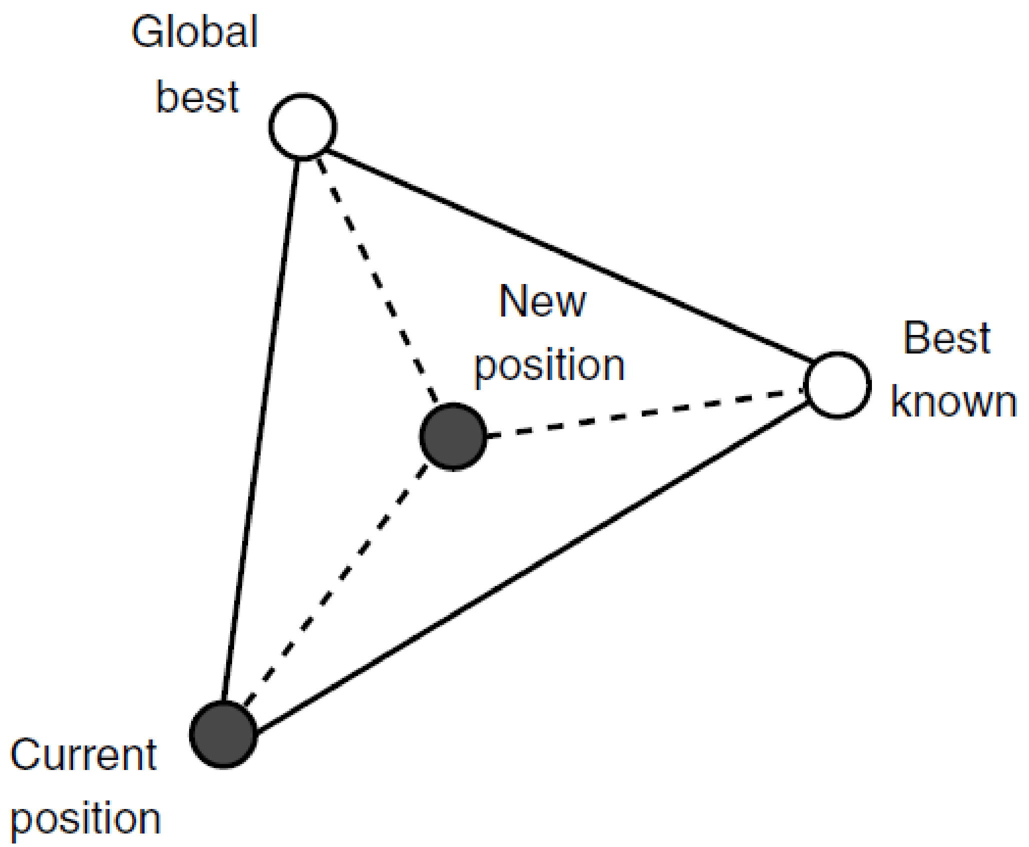


FIGURE 3.36 Geometric crossover in the GPSO algorithm.

TABLE 3.7 Parameters of the PSO Algorithm

Parameter	Role	Practical Values
n	Number of particles	[20, 60]
τ_1, τ_2	Acceleration coefficients	≤ 2.0
k	Neighborhood size	$\left[2, n \cdot \frac{n-1}{2}\right]$
w	Inertia weight	[0.4, 0.9]

Scatter Search

- Scatter search has its origin in the paper of F. Glover (1977).
- SS is a deterministic strategy that has been applied successfully to some combinatorial and continuous optimization problems.
- SS is a evolutionary and population metaheuristic that recombines solutions selected from a reference set to build others.
- The method starts by generating an initial population Pop satisfying the criteria of diversity and quality.
- The reference set (RefSet) of moderate size is then constructed by selecting good representative solutions from the population.

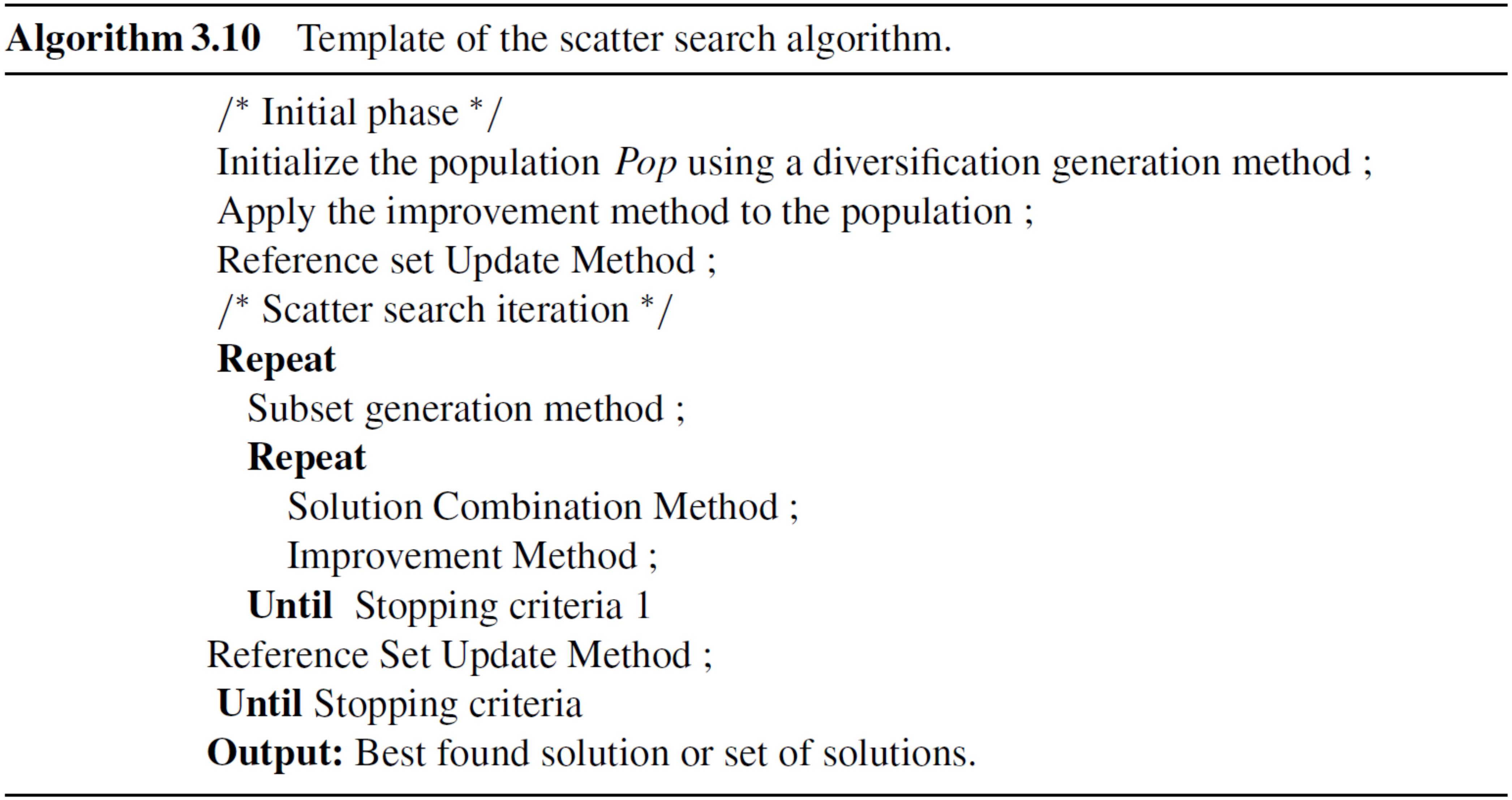
Scatter Search

- The selected solutions are combined to provide starting solutions to an improvement procedure based on a S-metaheuristic.
- According to the result of such procedure, the reference set and even the population of solutions are updated to incorporate both high-quality and diversified solutions. The process is iterated until a stopping criterion is satisfied.
- The SS approach involves different procedures allowing to generate the initial population, to build and update the reference set, to combine the solutions of such set, to improve the constructed solutions, and so on.

Scatter Search

- SS uses explicitly strategies for both search intensification and search diversification.
- It integrates search components from P-metaheuristics and S-metaheuristics.
- The algorithm starts with a set of diverse solutions, which represents the reference set (initial population)
- This set of solutions is evolved by means of recombination of solutions as well as by the application of local search (or another S-metaheuristic).

Scatter Search



Algorithm 3.10 Template of the scatter search algorithm.

```
/* 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.
```

The design of a scatter search algorithm is generally based on the following five methods (Fig. 3.29) :

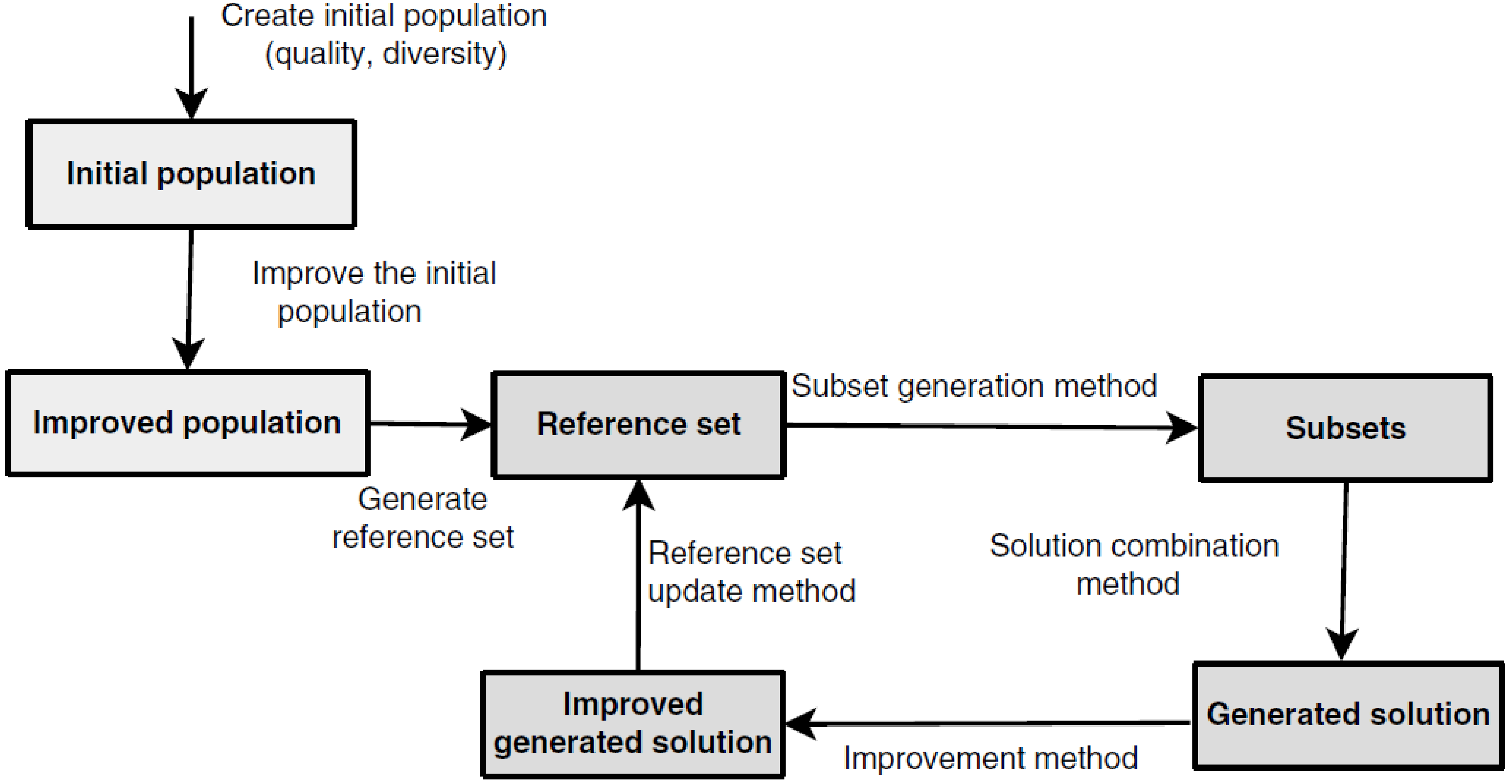


FIGURE 3.29 Search components of scatter search algorithms.

Variable Neighborhood Search

- Recall that there are several ways to classify metaheuristics
- Single solution based vs. population based
- Single solution based: local search, simulated annealing, tabu search
- Population based: genetic algorithm, ant colony, particle swarm

- We will go back to the first category and learn about Variable Neighborhood Search (VNS)

Escaping from Local Optima

- In general, local search is a very easy method to design and implement and gives fairly good solutions very quickly. This is why it is a widely used optimization method in practice.

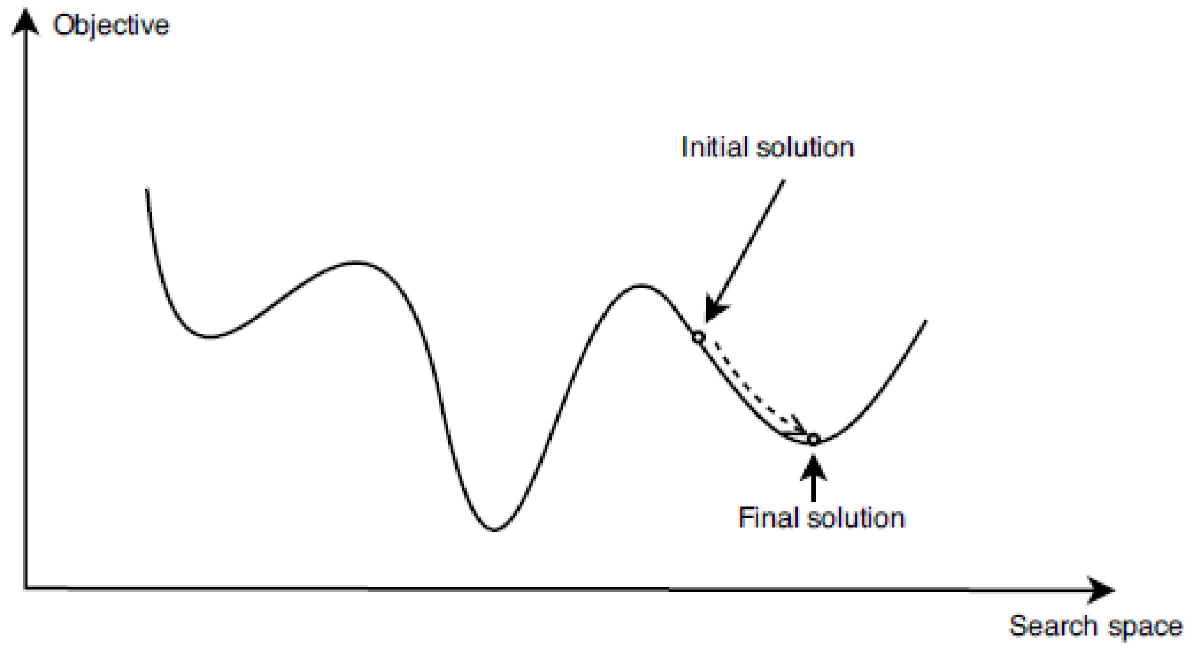


FIGURE 2.22 Local search (steepest descent) behavior in a given landscape.

- One of the main disadvantages of LS is that it converges toward local optima.
- Moreover, the algorithm can be very sensitive to the initial solution.
- No means to estimate the relative error from the global optimum and the number of iterations performed may not be known in advance.

Escaping from Local Optima

- Local search works well if there are not too many local optima in the search space or the quality of the different local optima is more or less similar.
- If the objective function is highly multimodal, which is the case for the majority of optimization problems, local search is usually not an effective method to use.
- As the main disadvantage of local search algorithms is the convergence toward local optima, many alternatives algorithms have been proposed to avoid becoming stuck at local optima.

Escaping from Local Optima

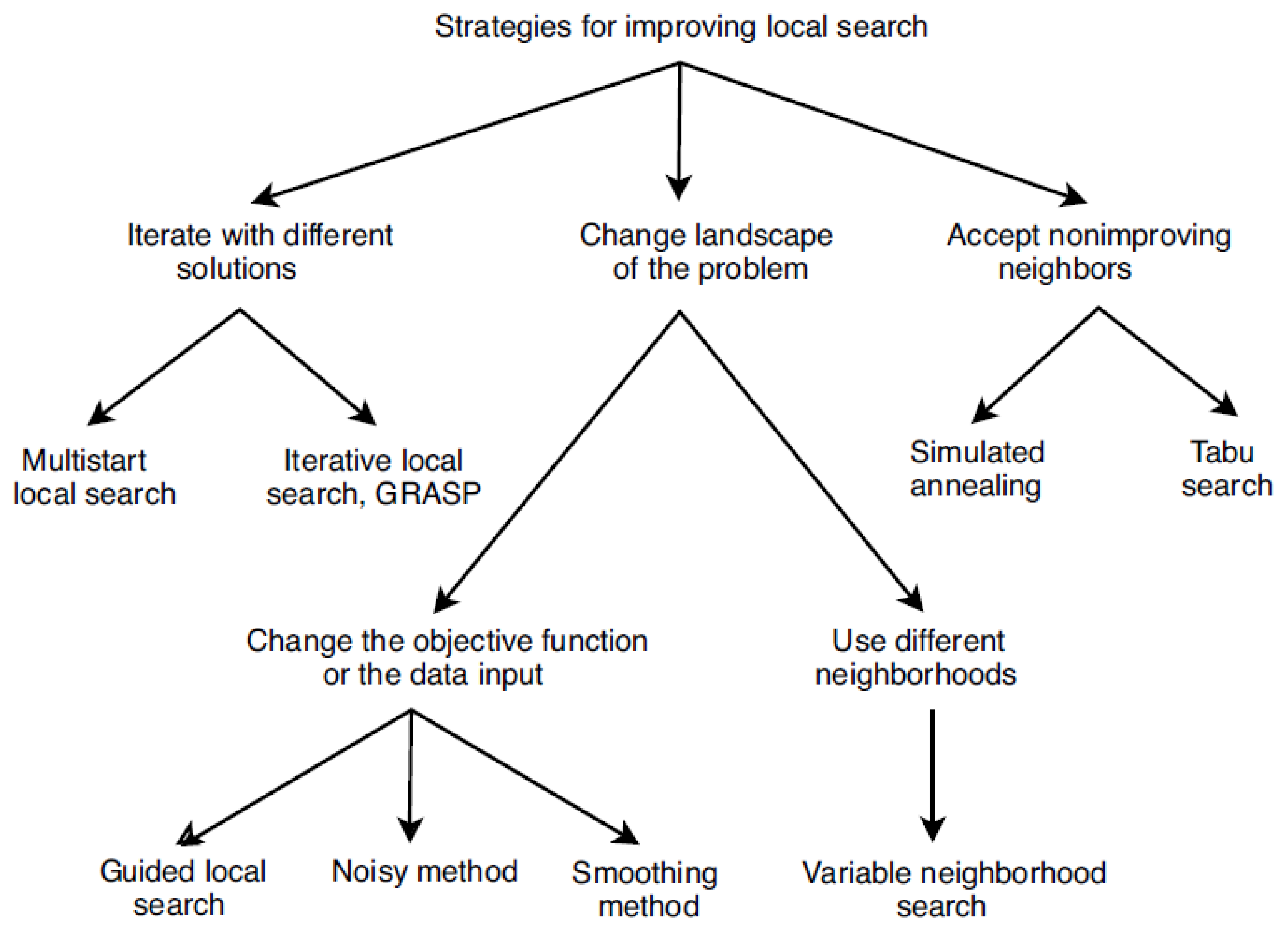

Strategies for improving local search
Iterate with different solutions
Change landscape of the problem
Accept nonimproving neighbors
Multistart local search
Iterative local search, GRASP
Simulated annealing
Tabu search
Change the objective function or the data input
Use different neighborhoods
Guided local search
Noisy method
Smoothing method
Variable neighborhood search

FIGURE 2.24 S-metaheuristic family of algorithms for improving local search and escaping from local optima.

Variable Neighborhood Search

- Variable neighborhood search has been recently proposed by P. Hansen and N. Mladenovic in 1997.
- The basic idea of VNS is to successively explore a set of predefined neighborhoods to provide a better solution.
- It explores either at random or systematically a set of neighborhoods to get different local optima and to escape from local optima.
- VNS exploits the fact that using various neighborhoods in local search may generate different local optima and that the global optima is a local optima for a given neighborhood.
- Indeed, different neighborhoods generate different landscapes.

Variable Neighborhood Search

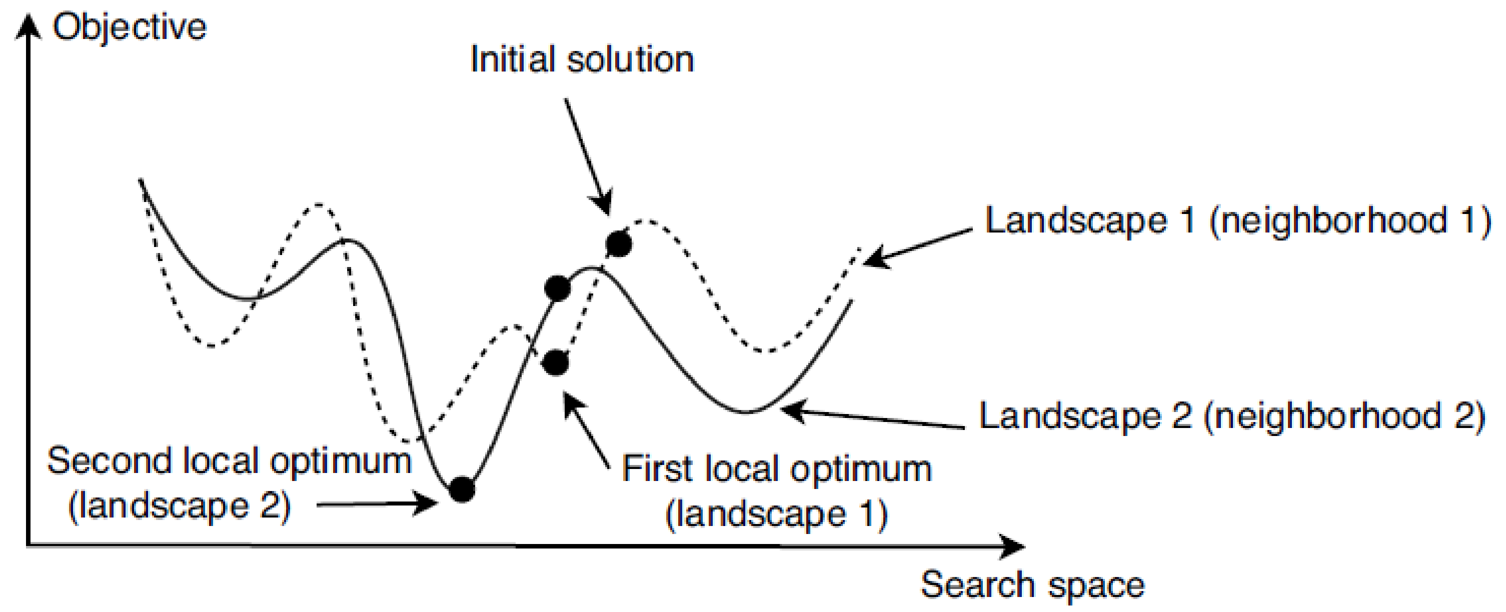


FIGURE 2.30 Variable neighborhood search using two neighborhoods. The first local optimum is obtained according to the neighborhood 1. According to the neighborhood 2, the second local optimum is obtained from the first local optimum.

Variable neighborhood descent

- The VNS algorithm is based on the variable neighborhood descent, which is a deterministic version of VNS.
- VND uses successive neighborhoods in descent to a local optimum.
- First, one has to define a set of neighborhood structures N_l $(l = 1, \ldots, l_{max})$.
- Let N_1 be the first neighborhood to use and x the initial solution.
- If an improvement of the solution x in its current neighborhood $N_l(x)$ is not possible, the neighborhood structure is changed from N_l to N_{l+1}.
- If an improvement of the current solution x is found, the neighborhood structure returns to the first one $N_1(x)$ to restart the search
- This strategy will be effective if the different neighborhoods used are complementary in the sense that a local optima for a neighborhood N_i will not be a local optima in the neighborhood N_j.

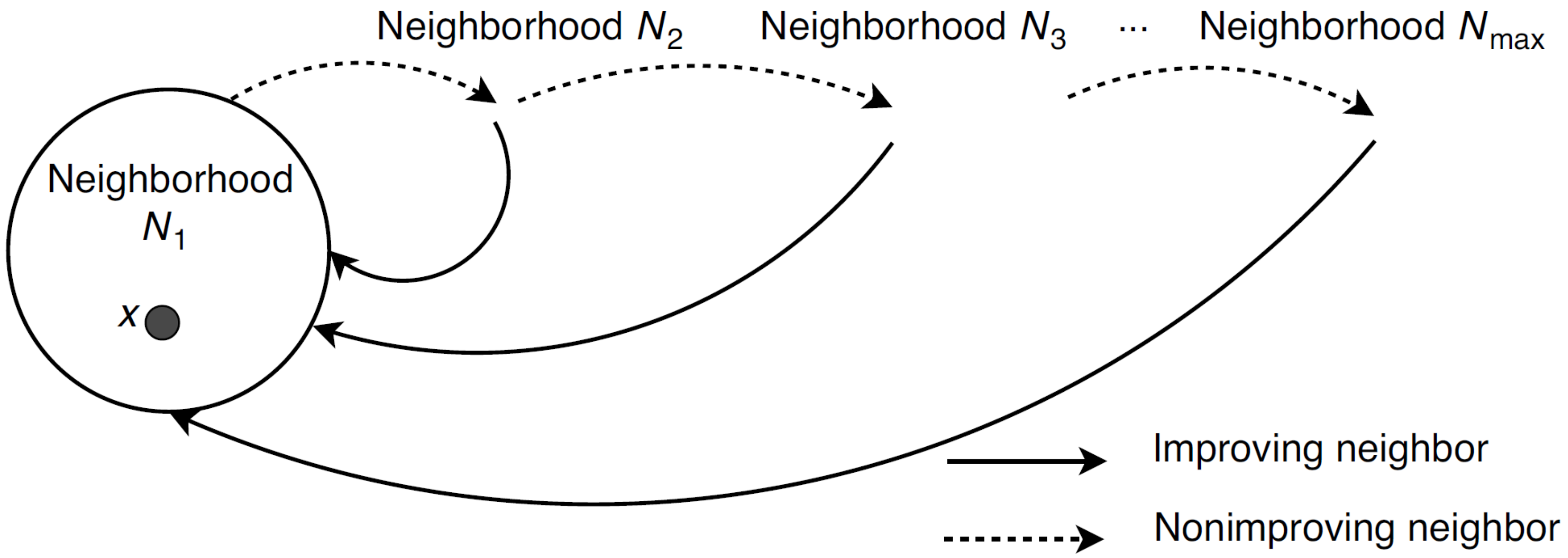


FIGURE 2.31 The principle of the variable neighborhood descent algorithm.

Algorithm 2.11 Template of the variable neighborhood descent algorithm.

```
**Input**: a set of neighborhood structures $N_l$ for $l = 1, \ldots, l_{max}$.
$x = x_0$ ; /* Generate the initial solution */
$l = 1$ ;
While $l \leq l_{max}$ Do
  Find the best neighbor $x'$ of $x$ in $N_l(x)$ ;
  If $f(x') < f(x)$ Then $x = x'$ ; $l = 1$ ;
  Otherwise $l = l + 1$ ;
Output: Best found solution.
```

Neighborhoods

- The design of the VND algorithm is mainly related to the selection of neighborhoods and the order of their application.
- The complexity of the neighborhoods in terms of their exploration and evaluation must be taken into account.
- The larger are the neighborhoods, the more time consuming is the VND algorithm.
- Concerning the application order, the most popular strategy is to rank the neighborhoods following the increasing order of their complexity (e.g., the size of the neighborhoods $|N_l(x)|$).

General Variable Neighborhood Search

- VNS is a stochastic algorithm in which, first, a set of neighborhood structures N_k $(k = 1, \ldots, n)$ are defined.
- Then, each iteration of the algorithm is composed of three steps: shaking, local search, and move.
- At each iteration, an initial solution is shaked from the current neighborhood N_k.
- For instance, a solution x' is generated randomly in the current neighborhood $N_k(x)$.
- A local search procedure is applied to the solution x' to generate the solution x''.
- The current solution is replaced by the new local optima x'' if and only if a better solution has been found (i.e., $f\ (x'') < f\ (x)$).

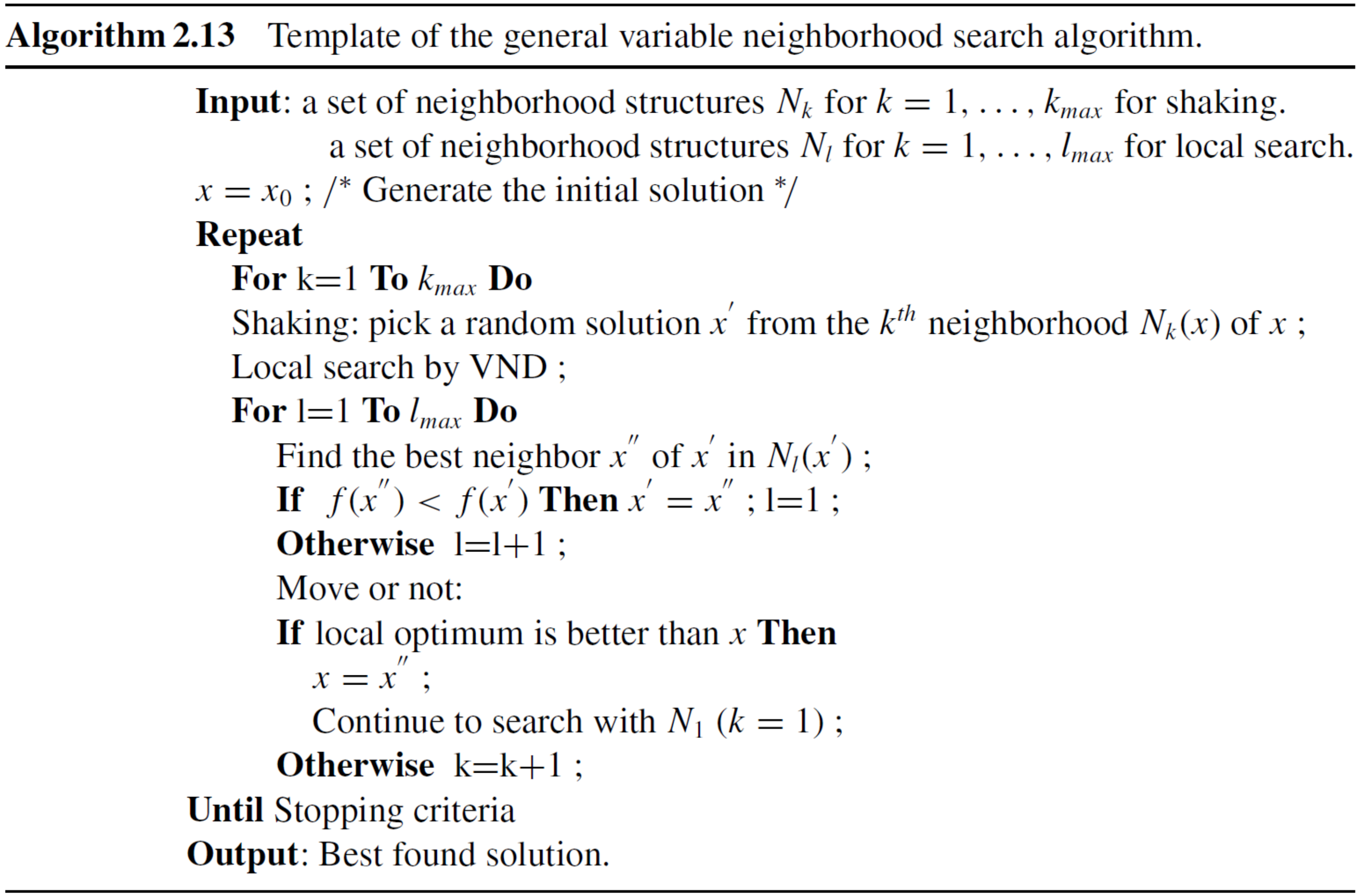
General Variable Neighborhood Search

- The same search procedure is thus restarted from the solution x'' in the first neighborhood N_1.
- If no better solution is found (i.e., $f\ (x'')\ \geq\ f\ (x)$), the algorithm moves to the next neighborhood N_{k+1}, randomly generates a new solution in this neighborhood, and attempts to improve it.
- Let us notice that cycling is possible (i.e., $x''\ =\ x$).
- The template of the basic variable neighborhood search is as follows:

Algorithm 2.12 Template of the basic variable neighborhood search algorithm.

```
**Input**: a set of neighborhood structures $N_k$ for $k = 1, \ldots, k_{max}$ for shaking.
$x = x_0$ ; /* Generate the initial solution */
**Repeat**
 $k = 1$ ;
  **Repeat**
    Shaking: pick a random solution $x^{'}$ from the $k^{th}$ neighborhood $N_k(x)$ of $x$ ;
    $x''$ =local search($x'$) ;
    **If** $f(x'') < f(x)$ **Then**
        $x = x^{''}$ ;
        Continue to search with $N_1$ ; $k = 1$ ;
    **Otherwise** k=k+1 ;
  **Until** $k = k_{max}$
**Until** Stopping criteria
**Output**: Best found solution.
```

- A more general VNS algorithm where the simple local search procedure is replaced by the VND algorithm:



Algorithm 2.13 Template of the general variable neighborhood search algorithm.

```
Input: a set of neighborhood structures $N_k$ for $k = 1, \ldots, k_{max}$ for shaking.
           a set of neighborhood structures $N_l$ for $k = 1, \ldots, l_{max}$ for local search.
$x = x_0$ ; /* Generate the initial solution */
Repeat
  For k=1 To $k_{max}$ Do
  Shaking: pick a random solution $x'$ from the $k^{th}$ neighborhood $N_k(x)$ of $x$ ;
  Local search by VND ;
  For l=1 To $l_{max}$ Do
     Find the best neighbor $x''$ of $x'$ in $N_l(x')$ ;
     If $f(x'') < f(x')$ Then $x' = x''$ ; l=1 ;
     Otherwise l=l+1 ;
     Move or not:
     If local optimum is better than $x$ Then
        $x = x''$ ;
        Continue to search with $N_1$ $(k = 1)$ ;
     Otherwise k=k+1 ;
Until Stopping criteria
Output: Best found solution.
```

General Variable Neighborhood Search

- In addition to the design of a simple local search (see Section 2.3) or a VND algorithm, the design of the VNS algorithm is mainly related to the selection of neighborhoods for the shaking phase.
- Usually, nested neighborhoods are used, where each neighborhood $N_k(x)$ contains the previous one $N_{k-1}(x)$:

$$N_1(x) \subset N_2(x) \subset \cdots \subset N_k(x) \; \forall x \in S$$

General Variable Neighborhood Search

- A compromise must be found between intensification of the search and its diversification through the distribution of work between the local search phase and the shaking phase.
- An increased work in the local search phase will generate better local optima (more intensification), whereas an increased work in the shaking phase will lead to potentially better regions of the search space (more diversification).

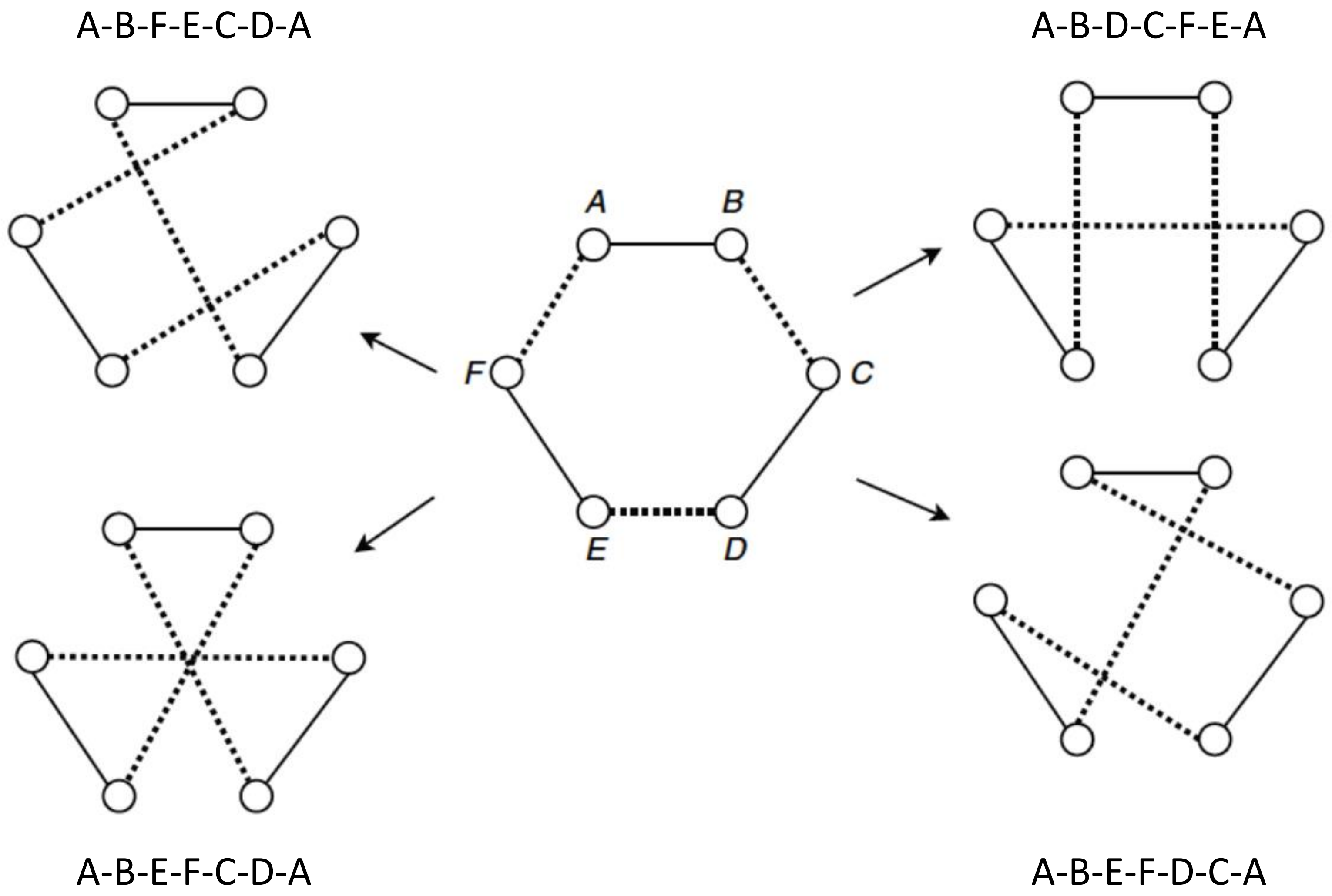
General Variable Neighborhood Search

Example 2.34 Nested neighborhoods for the shaking phase.

For many combinatorial optimization problems, there is a natural definition of nested neighborhoods. For instance, for routing problems such as the traveling salesman problem and the vehicle routing problem, the k-opt operator can be used for different values of k

$(k = 2, 3, \dots)$.

3-opt example



Parameters of VNS

- As in local search, VNS requires a small number of parameters.
- For the shaking phase, the single parameter is the number of neighborhood structures k_{max}.
- If large values of k_{max} are used (i.e., very large neighborhoods are considered), VNS will be similar to a multistart local search.
- For small values of k_{max}, VNS will degenerate to a simple local search algorithm.