1

Multi-Objective Evolutionary Algorithms and Evolutionary Large-Scale Optimization

1.1 Introduction

The word *optimization* initially appeared in the mid-19th century, meaning "to make the best or most of." Today, optimization has deeply penetrated into our thoughts and actions: engineers aim to achieve the best performance of their designs; businessmen seek to maximize the profits of their deals; manufacturers expect the highest efficiency of their production processes; investors try to minimize the risk of investment while maximizing rate of return, etc. With sophisticated formulations and modelings, such optimization-driven targets become *optimization problems*.

With problems always come solvers, as known as *algorithms* in the context of optimization. In the early age, the conventional optimization algorithms were often gradient based, e.g., Newton's method and the hill climbing method, assuming that the problems to be solved were convex (or at least differentiable) – and indeed – problems were supposed to be well defined by mathematicians. Nonetheless, the information age commencing from the mid-20th century has led modernized science, engineering, industry, and modernized optimization as well. While enjoying the benefits of modernization, we are facing emerging challenges – optimization problems are increasingly complex, far beyond the scope of conventional optimization algorithms.

Scientists have always been dedicated to looking into nature to find answers to research questions, so as in the case of solving optimization problems. The *nature-inspired optimization algorithms* are exactly a family of modern algorithms for global optimization. Facing the challenging features of complex optimization problems (e.g., nonlinearity, non-differentiability, or multi-modality), nature-inspired optimization algorithms benefit from unique advantage of robustness – treating optimization problems as *black-boxes*. Particularly, inspired by the mechanisms of biological evolution, the evolutionary

Evolutionary Large-Scale Multi-Objective Optimization and Applications: Integrating Evolutionary Computation, Machine Learning, and Data Science,
First Edition. Xingyi Zhang, Ran Cheng, Ye Tian, and Yaochu Jin.

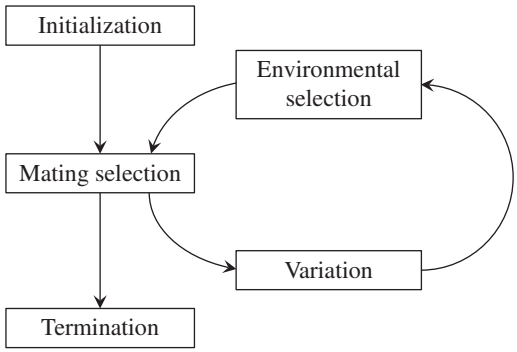

Figure 1.1 The generic flowchart of an *Evolutionary Algorithm* (*EA*).

algorithms (EAs) [1] are among the most representative niches of the family. Generally speaking, EAs encompass genetic algorithms (GAs), evolutionary strategies, evolutionary programming, and genetic programming.

As illustrated in Fig. 1.1, a generic flowchart of EAs comprises five main components: initialization, mating selection, variation, environmental selection, and termination. Normally, an EA starts with a randomly initialization population of candidate solutions. Then, the population of candidate solutions is iteratively updated via the main loop consisting of mating selection, variation, and environmental selection. Specifically, mating selection randomly picks up pairwise (parent) candidate solutions for generating new (offspring) candidate solutions via variation, and environmental selections decide which candidate solutions can survive to the next iteration (generation). Eventually, when a certain condition (e.g., the maximum iteration number) is satisfied, the EA will terminate and output the candidate solutions in the final population as the approximate optimal solution(s) to the optimization problem being solved.

During the past three decades, EAs have achieved remarkable success in solving complex optimization problems; recently, however, they are facing a new challenge brought by the surging development of information sciences – the *scalability*. Just as many other computer algorithms, the nature-inspired optimization algorithms suffer from a serious *curse of dimensionality*, i.e., as the scale of the optimization problems increases, the performance of the algorithms substantially deteriorates. Now, we are stepping forward into new adventures – scaling up EAs for solving *large-scale optimization problems*.

Before moving toward the main course of large-scale optimization problems, we might start with an appetizer – general optimization problems. Generally, an optimization problem often consists of three essential components: objective(s), decision variable(s), and constraint(s): an objective depicts the performance of the system under study, and a decision variable is a certain characteristic in correlation with the objective(s); the spaces where the objectives and decision variables are

defined are often known as objective space and decision space, respectively, and a constraint specifies the limitation(s) in either the objective space or decision space. Correspondingly, an optimization problem can be mathematically formulated as[1]:

$$\begin{aligned} &\min_{\mathbf{x}} \ \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})), \\ &\text{s.t.} \quad \mathbf{x} \in X, \quad \mathbf{f} \in Y, \end{aligned} \tag{1.1}$$

where $X \subset \mathbb{R}^n$ is the decision space and $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in X$ is the decision vector; $Y \subset \mathbb{R}^m$ is the objective space and $\mathbf{f} \in Y$ is the objective vector. The decision vector $\mathbf{x}$ is composed of n decision variables while the objective vector $\mathbf{f}$ is composed of m objective functions that map $\mathbf{x}$ from X to Y.

Considering the number of objectives, optimization problems can be intrinsically categorized into two types: single-objective optimization problems (SOPs) (i.e., $m = 1$) and multi-objective optimization problems (MOPs) (i.e., $m > 1$). For an SOP, the optimal solution is often a single decision vector minimizing (or maximizing) the objective. For an MOP, since the multiple objectives are often conflicting with each other, it is impossible to find any single solution minimizing (or maximizing) all objectives simultaneously; instead, the optima to an MOP is often a set of solutions trading-off between different objectives, known as the Pareto optimal solutions. Specially, an MOP with more than three objectives is also known as a many-objective optimization problem (MaOP) due to its particular challenges [2].

As illustrated in Fig. 1.2, optimization problems can be large scale in terms of either objectives or decision variables, intrinsically comprising four types: (i) SOPs with a large number of decision variables (LSSOPs), (ii) MOPs with a large number of decision variables (LSMOPs), (iii) MaOPs with a normal scale of decision variables, and (iv) MaOPs with a large number of decision variables (LSMaOPs).

For SOPs, the optimization target is quite straightforward – searching for the global optimum solution $\mathbf{x}^*$ which minimizes the objective function $f(\mathbf{x}^*)$. By contrast, for MOPs/MaOPs, due to the possible conflicts between different objectives, no single solution is able to minimize all objectives simultaneously. As a consequence, there exist a set of optimal solutions trading-off between different objectives, namely, the *Pareto optimality* – defining a situation where no solution can be better off without making at least another solution worse off. In other words, the optimality of MOPs/MaOPs is defined upon the pairwise relationship (the *dominance relationship*) among candidate solutions considering the trade-offs between different optimization objectives. To be specific, a solution

1 This book only considers minimization problems, while maximization problems can be equivalently transformed into minimization problems by negating the objective functions.

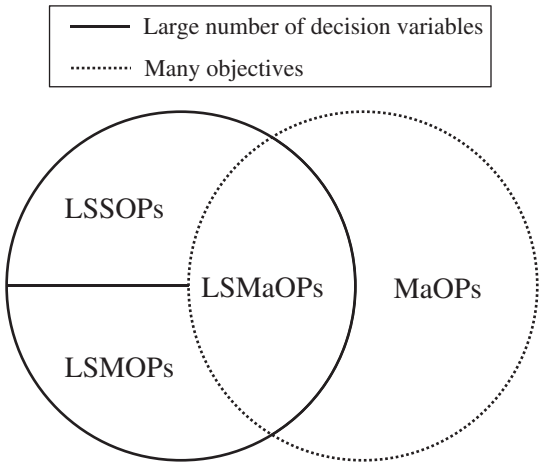

Figure 1.2 This figure illustrates the classification of large optimization problems involved in this chapter. Generally, the problems can have either a large number of decision variables or many objectives, which can be further categorized into four classes: single-objective problems with a large number of decision variables (LSSOPs), multi-objective optimization problems with a large number of decision variables (LSMOPs), many-objective optimization problems (MaOPs), and many-objective optimization problems with a large number of decision variables (LSMaOPs).

$\mathbf{x}_1$ is said to *dominate* another solution $\mathbf{x}_2$ (denoted as $\mathbf{x}_1 \succ \mathbf{x}_2$) iff

$$\begin{cases} \forall i \in 1, 2, \ldots, m : f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2), \\ \exists j \in 1, 2, \ldots, m : f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2). \end{cases} \tag{1.2}$$

If a solution $\mathbf{x}^*$ cannot be dominated by any other feasible solutions, $\mathbf{x}^*$ is said to be *Pareto optimal* and the union of all $\mathbf{x}^*$ is called *Pareto set (PS)*, while the union of $\mathbf{f}(\mathbf{x}^*)$ is termed *Pareto front (PF)*. An illustrative example of Pareto optimality is given in Fig. 1.3.

For most continuous MOPs/MaOPs, the PS (PF) often consists of an infinite number of Pareto optimal solutions. Therefore, in practice, only a representative subset of the PS (PF) can be approximated using a limited number of *non-dominated solutions*, which are often the candidate solutions not dominated by any other in the final solution set obtained by an algorithm.

In this chapter, we mainly focus on problems of type (ii) and type (iv), i.e., LSMOPs and LSMaOPs. Since MaOPs are essentially MOPs of a special case, for simplicity, we unified the abbreviations of both LSMOPs and LSMaoPs to be LSMOPs. Specifically, we refer to research adopting EAs to solve large-scale optimization problems as *evolutionary large-scale optimization*. The rest of this chapter is organized as follows: Section 1.2 briefly introduces multi-objective evolutionary algorithms (MOEAs), Section 1.3 overviews evolutionary large-scale optimization, and Section 1.4 summarizes this chapter.

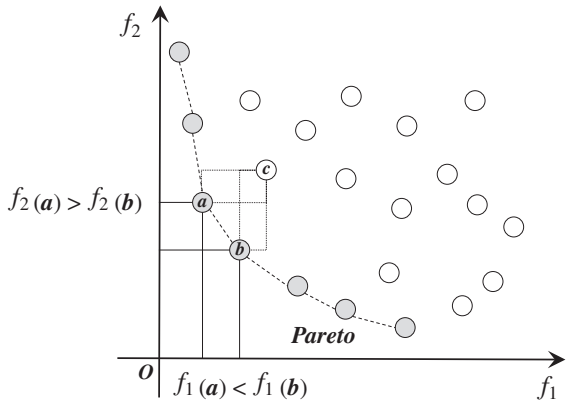

Figure 1.3 An illustrative example to *Pareto optimality*. The images of a number of candidate solutions (**a**, **b**, **c**, etc.) are spreading in the objective space. **a** and **b**, non-dominated to each other, are among the Pareto optimal solutions on the PF (PS). **c** is dominated by both **a** and **b**.

1.2 Multi-Objective Evolutionary Algorithms (MOEAs)

Dating back to the GA proposed by John Holland in 1975 [3], the EAs, generally referring to the family of soft computing paradigms inspired by natural evolution mechanisms, have witnessed booming development during the past half-century. Given an optimization problem, an EA iteratively maintains a population of candidate solutions undergoing evolutionary operators such as crossover, mutation, and selection. As a consequence, the population is expected to move toward optimum to the optimization problem, just as species adapting to the natural environments.

1.2.1 MOEAs for Solving MOPs

Considering the population-based character, EAs are intrinsically suited for solving MOPs – to obtain a set of candidate solutions trading-off between multiple optimization objectives. Since Schaffer proposed the vector-evaluated GA [4] in 1984, a large number of MOEAs have been proposed. In general, MOEAs proposed for solving MOPs can be categorized into three groups: dominance-based ones, performance-indicator-based ones, and decomposition[2] -based ones.

Dominance-based MOEAs have been prevalent over the past two decades. Early dominance-based non-elitism MOEAs include the non-dominated sorting GA [5], the niched-Pareto GA [6], and the multi-objective genetic algorithm [7]. It was found out later on that the use of elitism strategies can substantially improve the convergence of MOEAs. As a result, several popular elitism MOEAs have been developed, including the elitist non-dominated sorting genetic algorithm (NSGA-II) [8], the strength Pareto evolutionary algorithm 2

2 Decomposition in multi- or many-objective optimization specifically refers to the decomposition in the objective space.

(SPEA2) [9], and the Pareto envelope-based selection algorithm (PESA) [10, 11]. Although dominance-based MOEAs are very powerful for solving bi-objective or three-objective MOPs, their convergence performance dramatically degrades when the number of objectives is larger than three, mainly due to the loss of selection pressure.

Decomposition-based approaches divide a complex MOP into a number of sub-problems and solve them in a collaborative manner. There are mainly two types of decomposition-based approaches. In the first type of decomposition-based approaches, an MOP is decomposed into a group of single-objective problems (SOPs), including the weighted aggregation-based approaches in the early days [12, 13], and the more recent multi-objective evolutionary algorithm based on decomposition (MOEA/D), where more explicit collaboration strategies between the solutions of the subproblems were introduced. A number of variants of MOEA/D have been proposed for enhancing the selection strategy for each subproblem to strike a better balance between convergence and diversity [14–16]. In the second type of decomposition-based approaches, an MOP is decomposed into a group of sub-MOPs. For instance, MOEA/D-M2M [17, 18] divides the whole PF into a group of segments, and each segment can be regarded as a subproblem.

Since performance indicators are used to measure the solution qualities, it is quite intuitive to also use them as selection criteria to guide the search of the algorithms to obtain high-quality solutions. Among various performance indicators, the hypervolume indicator (or the S metric) is most widely applied in various MOEAs, due to the fact that it is the only metric known to be strictly monotonic to the Pareto dominance [19]. The first well-known hypervolume indicator-based algorithm was proposed by Emmerich *et al.* in 2005, known as the s-metric selection evolutionary multi-objective algorithm (SMS-EMOA) [20]. In SMS-EMOA, a $(\mu + 1)$ evolutionary strategy is adopted to generate one offspring candidate solution from a number of μ parents. To select a number of μ new candidate solutions for the next generation, the hypervolume contribution of each candidate solution on the last non-dominated front is calculated, and the one with the smallest hypervolume contribution is removed. Apart from the hypervolume indicator, some other performance indicators have also been adopted to guide the search of MOEAs. Zitzler and Künzli have proposed a binary indicator ($I_{\epsilon+}$)-based evolutionary algorithm (IBEA) [21], where $I_{\epsilon+}$ is considered to be compliant with the Pareto dominance. Tian *et al.* have proposed an IGD-NS-based multi-objective evolutionary algorithm (AR-MOEA) [22], where a reference point adaptation method is developed to adjust the reference points for the calculation of IGD-NS [23] at each generation.

Despite the various MOEAs as introduced above, all of them have more or less inherited the merits from NSGA-II – either the framework or the main algorithm components. On the one hand, the simple and flexible framework of NSGA-II

allows users to replace the main algorithm components with their tailored ones; on the other hand, the main algorithm components in NSGA-II can be easily modified as required. More recently, NSGA-II has also shed light on the literature of evolutionary large-scale multi-objective optimization, and it is highly related to the works to be presented in this book. Hence, before moving to the rest of chapters of this book, the readers are encouraged to go deep into the details of NSGA-II as given in the following.

As presented in Algorithm 1.1, the main loop of NSGA-II follows the workflow of a typical EA (Fig. 1.1), involving three main components: mating selection, variation, and environmental selection. More specifically, the environmental selection in NSGA-II consists of another two subcomponents: the non-dominated sorting (Line 2) and the crowding distance sorting (Line 3), which are two tailored sorting methods considering convergence criterion and diversity criterion when performing environmental selection, respectively. In the following, we will detail each component in NSGA-II one by one.

Algorithm 1.1: Procedure of NSGA-II

Input: N (population size)
Output: P (final population)

```
1 P ← Randomly generate N solutions;
2 Front ← NondominatedSorting(P); //Algorithm 1.2
3 Crowd ← CrowdingDistance(P, Front); //Algorithm 1.3
4 while termination criterion not fulfilled do
5 |   P' ← MatingSelection(P, N, Front, Crowd); //Algorithm 1.4
6 |   O ← Variation(P'); //Algorithm 1.5
7 |   [P, Front, Crowd] ← EnvironmentalSelection(P ∪ O, N);
  |     //Algorithm 1.6
8 return P;
```

In evolutionary multi-objective optimization, the candidate solutions in the final population of an MOEA are expected to be an approximation to the PF of the MOP to be solved, and the motivation of non-dominated sorting in NSGA-II is exactly to meet the basic requirement (i.e., Pareto dominance, the *convergence criterion*) of the candidate solutions.

Basically, the *non-dominated sorting* method (Algorithm 1.2) is motivated to partition the candidate solutions into a number of *non-dominated fronts* (fronts for short) according to the dominance relationship as defined by Eq. (1.2). Consequently, as illustrated in Fig. 1.4, the candidate solutions inside each front

Algorithm 1.2: *NondominatedSorting*(P)

Input: P (population)
Output: *Front* (non-dominated front number of each solution)

```
//Efficient non-dominated sort with sequential search
P ← Sort the solutions in P in an ascending order of the first objective values;
Front ← A 1 × |P| vector of zeros;
maxF ← 0; //Maximum non-dominated front number
while not all Front_i > 0 do
    maxF ← maxF + 1;
    for i = 1, ..., |P| do
        if Front_i == 0 then
            Dominated ← false; //Whether solution P_i is
              dominated
            for j = i − 1, ..., 1 do
                if Solution P_j dominates solution P_i then
                    Dominated ← true;
                    break;
            if Dominated == false then
                Front_i ← maxF;
    return Front;
```

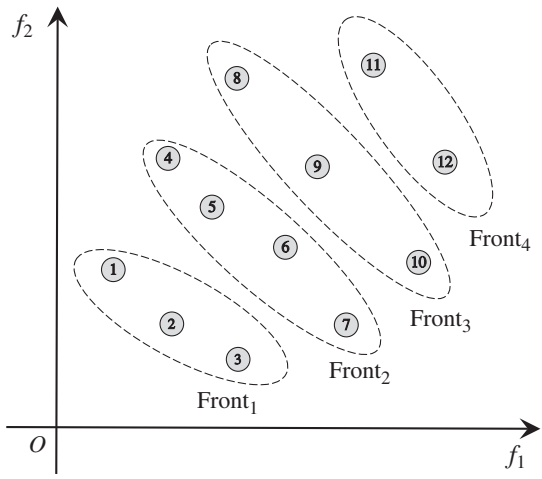

Figure 1.4 An illustrative example to *non-dominated sorting*. The 12 candidate solutions are partitioned into four fronts by non-dominated sorting. The candidate solutions inside each front are indexed with the same front number. The candidate solutions inside $Front_1$ dominate those on the other three fronts.

are non-dominated to each other and share the same ranking index, while those in the last front (i.e., the one with the minimum front number) dominate all the rest.

In order to obtain a population of candidate solutions as an approximation to the PF of a given MOP, apart from the convergence criterion as considered in non-dominated sorting, it is equally important to take the *diversity criterion* into consideration. Specifically, in evolutionary multi-objective optimization, the diversity criterion often refers to degree of how uniformly/evenly the candidate solutions are spreading over the PF. It is due to the fact that, without any preference, the decision-maker can be interested in diverse candidate solutions diversely trading-off between different objectives.

The crowding distance sorting (Algorithm 1.3) in NSGA-II is exactly motivated to maintain diversity of the candidate solutions. As indicated by the name of the sorting method, crowding distance sorting aims to rank candidate solutions according to their crowding degrees (i.e., how much each candidate

Algorithm 1.3: *CrowdingDistance*(*Front*)

Input: P (population), *Front* (non-dominated front number of each solution)
Output: *Crowd* (crowding distance of each solution)

1 $Crowd \leftarrow$ A $1 \times |P|$ vector of zeros;
2 $maxF \leftarrow 0$; //`Current non-dominated front number`
3 **while** *not all* $Crowd_i > 0$ **do**
4 $\quad maxF \leftarrow maxF + 1$;
5 $\quad I \leftarrow \{i = 1, \ldots, |P| | Front_i = maxF\}$;
6 $\quad$ **for** $m = 1, \ldots, M$ **do**
7 $\quad\quad fmax = max_{i \in I} f_m(P_i)$; //$f_m(P_i)$ `is the` m`th objective value of the` i`th solution in` P
8 $\quad\quad fmin = min_{i \in I} f_m(P_i)$;
9 $\quad\quad I' \leftarrow$ Sort the solutions in I in an ascending order of the mth objective values;
10 $\quad\quad$ **for** $i = 1, \ldots, |I'|$ **do**
11 $\quad\quad\quad$ **if** $i == 1$ *or* $i == |I'|$ **then**
12 $\quad\quad\quad\quad Crowd_{I'_i} = \inf$;
13 $\quad\quad\quad$ **else**
14 $\quad\quad\quad\quad Crowd_{I'_i} = Crowd_{I'_i} + \frac{f_m(P_{I'_{i+1}}) - f_m(P_{I'_{i-1}})}{fmax - fmin}$;
15 $\quad$ **return** *Crowd*;

is crowded with respect of its closest neighbors). Eventually, the candidate solutions primarily with the best convergence (i.e., those on or as close to the last non-dominated front as possible) and secondarily best diversity (those having the least crowding degrees) will be selected.

The *mating selection* in NSGA-II is very similar to the conventional tournament selection (Algorithm 1.4) as adopted in single-objective GAs [24], except that it performs ranking according to two (instead of one) criteria – non-dominate front indexing ($Fit1$, ranking results by Algorithm 1.2) and crowding indexing ($Fit1$, ranking results by Algorithm 1.3). As indicated by the name itself, tournament selection is designed to select candidate solutions in a population via random "tournaments." Consequently, the winner of each tournament (i.e., the one with the best fitness value) is selected to go through *variation* (i.e., crossover/mutation).

Algorithm 1.4: *MatingSelection*($P, N, Fit1, Fit2$)

Input: P (population), N (number of selected parents), $Fit1$ (primary fitness of solutions), $Fit2$ (secondary fitness of solutions)
Output: P' (a set of parents)

```
P' ← ∅;
for i = 1, ..., N do
    [i, j] ← Randomly select two indexes from {1, ..., |P|};
    if Fit1_i < Fit1_j then
        P' ← P' ∪ {P_i};
    else if Fit1_i > Fit1_j then
        P' ← P' ∪ {P_j};
    else if Fit2_i < Fit2_j then
        P' ← P' ∪ {P_i};
    else
        P' ← P' ∪ {P_j};
return P';
```

It is worth noting that tournament selection (as well as any other mating selection method) is not necessarily an indispensable component of an MOEA, while it is expected to be more diverse, yet high-quality (in terms of fitness) candidate solutions when adopting it. The key mechanism is the random pairing of the candidate solutions according to probabilistic (but not deterministic) measurement of them. From a more general perspective, such kind

of randomization mechanism is widely seen and shown to be effective when coupled with MOEAs.

The *variation* operation (Algorithm 1.5) in NSGA-II adopts the classic *simulated binary crossover* and *polynomial mutation*, which are also widely adopted in real-coded genetic algorithms (RCGAs) [25]. Essentially, a variation operation is expected to generate high-quality new candidate solutions toward future evolution. As in the case of EA, crossover and mutation are the two most commonly seen variation operations. On the one hand, the crossover operation randomly recombines a pair of candidate solutions by exchange part of each; on the other hand, the mutation operation randomly modifies variable(s) of a candidate solution.

Algorithm 1.5: *Variation*(P')

Input: P' (a set of parents)
Output: O (offspring population)

```
1 O ← ∅;
2 while P' ≠ ∅ do
3     [x, y] ← Randomly select two parents from P';
4     P' ← P' \ {x, y};
5     [p, q] ← Perform simulated binary crossover and polynomial mutation
      based on x and y;
6     O ← O ∪ {p, q};
7 return O;
```

Since NSGA-II was initially proposed for solving MOPs with continuous search space, the variation operation using classic simulated binary crossover and polynomial mutation works pretty efficiently in generating candidate solutions for real-coded small-scale MOPs. From the probabilistic point of view, a variation operation can be seen as a process of sampling new data toward optimum in the decision space. Hence, a variation operation in high-dimensional space may also face challenges of *curse of dimensionality*, thus losing efficiency when dealing with LSMOPs. To this end, this book (Chapters 3 and 4) will introduce some novel variation operators tailored for solving LSMOPs.

Since *environmental selection* substantially determines where the candidate solutions evolve toward, it plays a vital role in an EA. Particularly, in an MOEA, environmental selection must be delicately designed to balance convergence criterion and diversity criterion, aiming to obtain a population of candidate solutions

Algorithm 1.6: *EnvironmentalSelection*(P, N)

Input: P (combined population), N (population size)
Output: P (population for next generation), *Front* (non-dominated front number of each solution), *Crowd* (crowding distance of each solution)

1 $Front \leftarrow NondominatedSorting(P)$; `//Algorithm 1.2`
2 $Crowd \leftarrow CrowdingDistance(P, Front)$; `//Algorithm 1.3`
3 $i \leftarrow 1$;
4 **while** $\{P_j | Front_j = i\} \neq \emptyset$ **do**
5 $\quad F_i \leftarrow \{P_j | Front_j = i\}$; `//Set of solutions in the` i`th front`
6 $\quad i \leftarrow i + 1$;
7 $k \leftarrow \text{argmin}_i |F_1 \bigcup \cdots \bigcup F_i| \geq N$;
8 $F_k \leftarrow$ Delete $|F_1 \bigcup \cdots \bigcup F_k| - N$ solutions from F_k with the smallest *Crowd*;
9 $P \leftarrow F_1 \bigcup \cdots \bigcup F_k$;
10 *Front* $\leftarrow$ Retain the elements in *Front* whose corresponding solutions are in P;
11 *Crowd* $\leftarrow$ Retain the elements in *Crowd* whose corresponding solutions are in P;
12 **return** P, *Front*, and *Crowd*;

uniformly spreading over the approximated PF. To this end, the environmental selection in NSGA-II (Algorithm 1.6) consists of two subcomponents: the non-dominated sorting (Line 1) and the crowding distance sorting (Line 2), as introduced above, respectively.

Essentially, the reason for using two sorting methods together is to simultaneously take into consideration the convergence criterion (Pareto dominance) and the diversity criterion (crowding distance). Thus, the environmental selection can be seen as *multi-index sorting* procedure. As the *primary sorting*, the non-dominated sorting divides the candidate solutions into a number of fronts by considering the convergence criterion of Pareto optimality. Ideally, if the number of candidate solutions in the last front exactly equals the predefined number of candidate solutions to be selected (e.g., N out of $2N$), the environmental selection will end up here. However, it is very likely that the number of candidate solutions in the last front may exceed or not reach the total number of candidates to be selected and passed to the next generation. Hence, the environmental selection will continue to perform crowding distance sorting as the *secondary sorting* until the predefined number of selected candidate solutions is met (Lines 4–9).

1.2.2 MOEAs for Solving MaOPs

MOEAs have been shown to perform well on a wide range of MOPs with two or three objectives; however, MOEAs have experienced substantial difficulties when they are adopted to tackle MOPs with more than three objectives, i.e., the MaOPs [26–29]. As a result, MaOPs have attracted increasing attention in evolutionary optimization.

One major reason behind the failure of most conventional MOEAs in solving MaOPs can be attributed to the loss of selection pressure, i.e., the pressure for the population to converge toward the PF, when dominance is adopted as a criterion for selecting individuals with a limited population size [30]. For example, the elitist NSGA-II [8] and the SPEA2 [9], both of which use dominance-based selection, will fail to work properly when applied to the optimization of MaOPs. It is due to the fact that most candidate solutions generated in a population of a limited size are non-dominated, thus making the dominance-based selection criterion hardly possible to distinguish the candidate solutions, even in a very early stage of the search.

Another important reason for the degraded performance of MOEAs on MaOPs is the difficulty in maintaining a good population diversity in a high-dimensional objective space. Generally speaking, since the PFs of most continuous MOPs are piecewise continuous [31, 32], it is practically unlikely to approximate all Pareto optimal solutions. Instead, most MOEAs aim to find a set of evenly distributed representative solutions to approximate the PF. When the number of objectives is two or three, where the PF is typically a one-dimensional curve or two-dimensional surface, maintaining a good diversity of the solutions is relatively straightforward. As the dimension of the objective space increases, it becomes increasingly challenging to maintain a good population diversity, as the candidate solutions distribute very sparsely in the high-dimensional objective space. This consequently causes immense difficulties to the diversity management strategies widely used in MOEAs, e.g., the crowding distance-based diversity method in NSGA-II [33–35].

To enhance the performance of most traditional MOEAs in solving MaOPs, a number of approaches have been proposed [2, 36, 37], which can be roughly divided into three classes.

The first class is the *modified dominance*-based approaches. The basic idea is to enhance the convergence pressure by modifying the Pareto dominance to enlarge the dominating areas, such that some of the original non-dominated solutions become dominated by others. An early work of this class is the ϵ-dominance proposed by Laummans *et al.* [38], where the original dominance relation can be relaxed to a certain extent by tuning the parameter $\epsilon \in (0, 1)$. The bigger the ϵ is set, the more relaxed the dominance relation is, and vice versa. And, based on ϵ-dominance, Deb *et al.* have proposed a steady-state algorithm called ϵ-MOEA

[39]. In addition, some extended variants of the ϵ-dominance have also been proposed. In [40], Aguirre and Tanaka have proposed an ϵ-ranking multi-objective optimizer (ϵ R-EMO), where an adaptive ϵ-ranking procedure is performed on top of a space partitioning strategy. Another representative modified dominance is known as the α-domination, proposed by Ikeda *et al*. in [26]. Given two solutions, the α-domination allows a solution to be dominated by the other if it is slightly superior in one objective but substantially inferior in others. The α-domination has been successfully integrated into some MOEAs for solving MOPs, such as the guided multi-objective evolutionary algorithm (G-MOEA) [41]. As a combination of ϵ-dominance and α-dominance, Batista *et al*. have proposed another modified dominance, namely, the cone ϵ-dominance [42]. It has been reported that the cone ϵ-dominance is promising for solving MaOPs with up to 50 objectives [29]. Recently, Yang *et al*. have proposed a grid based evolutionary algorithm (GrEA) [43], where a grid dominance is designed on top of some adaptively constructed grid coordinates in the objective space. The grid dominance has an attractive feature that allows one solution to dominate another if it is slightly inferior in some objectives but substantially superior in some others, which is useful in distinguishing the non-dominated solutions in many-objective optimization.

The second class is the *diversity management enhancement*-based approaches. Since in many-objective optimization, the distribution of the candidate solutions becomes very sparse in the high-dimensional objective space, diversity management becomes particularly meaningful to alleviate the negative impact of the excessive diversity on the population convergence. An early work of this class is from Adra and Fleming, who have proposed a diversity management strategy to quantitatively evaluate the degree of the population diversity according to the distribution of the candidate solutions [34], and if the population is found excessively diverse, the diversity promotion mechanism in the selection procedure will be deactivated. Another more recent work of this class is the shift-based density estimation (SDE) proposed by Li *et al*. [44], where the basic idea is to penalize poorly converged solutions by assigning them a high-density value, such that only candidate solutions with both good convergence and distribution can survive to the next generation. In [45], Wang *et al*. have proposed to coevolve preferences with a population of candidate solutions for diversity management in many-objective optimization. Most recently, another representative algorithm falling into this class is the NSGA-III [46], where the diversity is managed by a set of predefined reference points. In addition, the recently proposed knee point-driven evolutionary algorithm (KnEA) also belongs to this category, where a knee point-based diversity secondary selection strategy is adopted on top of traditional non-dominated sorting [47].

While the first two classes aim to improve the performance of MOEAs on MaOPs, the third class, known as the *objective reduction* approaches, tries to

transform MaOPs to MOPs by reducing the number of redundant or irrelevant objectives, such that the problems can be directly solved with traditional MOEAs [48, 49]. In general, there are two types of objective reduction approaches: offline approaches and online approaches. Offline objective reduction approaches can be seen as preprocessing techniques, which work independently of specific MOEAs, but require a set of approximate Pareto optimal solutions as *a priori* knowledge. In [50], Brockhoff and Zitzler have presented both deterministic and stochastic algorithms for objective reduction based on the objective conflicts measured by ϵ-dominance. Sexane *et al.* have proposed to use principal component analysis (PCA) [51] and maximum variance unfolding (MVU) [52] for linear and nonlinear objective reduction, respectively [49]. Singh *et al.* have proposed to use corner solutions for objective reduction, which are obtained using a Pareto corner search evolutionary algorithm (PCSEA) [48]. Online objective reduction approaches are usually embedded in an existing MOEA and perform as part of the evolutionary procedure, where the idea is to update the reduced objectives iteratively using the solutions obtained by an MOEA during the optimization procedure. In [50], Brockhoff and Zitzler have proposed to embed the PCA approach into NSGA-II to build an online objective reduction approach, known as the PCA-NSGA-II. In [53], Jaimes *et al.* have proposed an online objective reduction algorithm to find a k-sized objective subset based on correlations between the objectives. Recently, Guo *et al.* have proposed to adopt the partitioning around medoids (PAM) [54] clustering algorithm to detect the correlations between objectives and divide them into different clusters [55].

Apart from the three classes of approaches, the decomposition-based approaches and the performance indicator-based approaches, which were originally proposed for solving MOPs, are also applicable to many-objective optimization. However, although neither of these two types of approaches are dominance based, they still suffer from some other issues.

For decomposition-based approaches, it is always required to predefine a set of weight (or reference) vectors before running the algorithms. In high-dimensional objective space, it becomes particularly challenging to specify a proper vector set to well approximate the PF. To remedy this issue, Li *et al.* have proposed an MOEA based on both dominance and decomposition, known as the MOEA/DD [56].

For performance indicator-based approaches, the major issue is the significant *curse of dimensionality* regarding the computational efficiency [57]. For example, the runtime of one generation in SMS-EMOA is as high as $O(N^{M/2+1})$, where N is the population size and M is the number of objectives. To tackle this issue, Bader and Zitzler have proposed a hypervolume estimation algorithm (HypE) [19], where the Monte Carlo sampling method is adopted to obtain the approximate hypervolume values for many-objective cases. Similar to SMS-MOEA, the selection in HypE is still based on the hypervolume contribution of the candidate

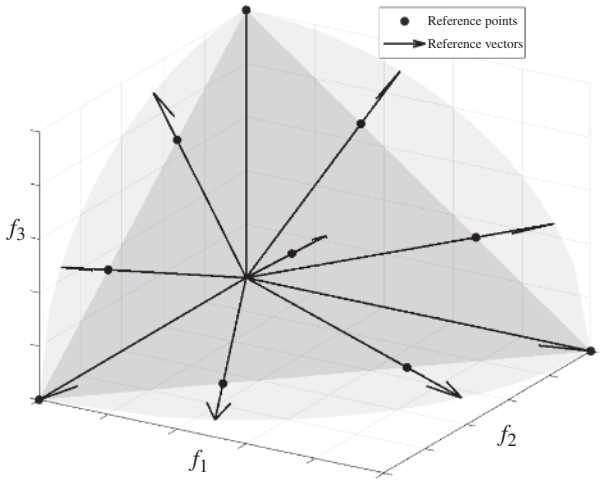

Figure 1.5 An illustration of how to generate the uniformly distributed reference vectors in a three-objective space. In this case, 10 uniformly distributed reference points are first generated on a hyperplane, and then they are mapped to a hypersphere to generate the 10 reference vectors.

solutions, while a minor difference is that the contribution is calculated based on a randomly chosen subset of the non-dominated solution set, where the size of the subset equals the number of solutions to be removed. In addition, some faster hypervolume calculation methods have been proposed recently, e.g., the hypervolume by slicing objectives (HSO) [58] and the walking fish group (WFG) hypervolume algorithm [59] (Fig. 1.5).

In summary, the approaches for solving MaOPs are either tailored on the basis of conventional MOEAs (originally proposed for solving MOPs) or adopting some strategies transforming MaOPs to MOPs. Moving toward LSMaOPs, an MOEA may suffer from the curse of dimensionality in both decision space and objective space, thus posing stiff challenges in convergence/diversity management of the candidate solutions. Recently, Cheng *et al.* have proposed a reference vector guided evolutionary algorithm (RVEA) [60] for solving MaOPs. RVEA inherited the generic elitism-selection-based framework (similar to that of NSGA-II) while adopting delicately tailored environmental selection inspired by both decomposition- and indicator-based MEOAs. Thanks to the generic framework and powerful convergence/diversity management capability, RVEA has proven to have certain potentials in scaling up to LSMaOPs, as will be elaborated in the rest of this book. Hence, as preliminary knowledge, we will introduce the details of RVEA in the following.

As summarized in Algorithm 1.7, RVEA adopts the framework of traditional elitism selection-based EAs – very similar to that of NSGA-II (Algorithm 1.1).

Algorithm 1.7: Procedure of RVEA

Input: N (population size)
Output: P (final population)

1 $P \leftarrow$ Randomly generate N solutions;
2 $V^0 \leftarrow$ *UniformSampling*(N, M); `//Algorithm 1.8`
3 $V \leftarrow V^0$;
4 **while** *termination criterion not fulfilled* **do**
5 $\quad P' \leftarrow$ Randomly select N solutions from P;
6 $\quad O \leftarrow$ *Variation*(P'); `//Algorithm 1.5`
7 $\quad [P, Fitness] \leftarrow$ *EnvironmentalSelection*$(P \cup O, V, N)$; `//Algorithm 1.9`
8 $\quad V \leftarrow$ *VectorAdaptation*(P, V, V^0); `//Algorithm 1.10`
9 **return** P;

To begin with, RVEA samples a set of reference vectors (Line 2). As illustrated in Fig. 1.5, the method of generating reference vectors in RVEA is inspired by the method of generating reference points for decomposition-based methods, using the *simplex-lattice design* in the case of requiring that the reference vectors uniformly distributed (Algorithm 1.8). Despite the deep connections between reference points and reference vectors, their working mechanisms are substantially different: a reference point specifies a region in the objectives space where

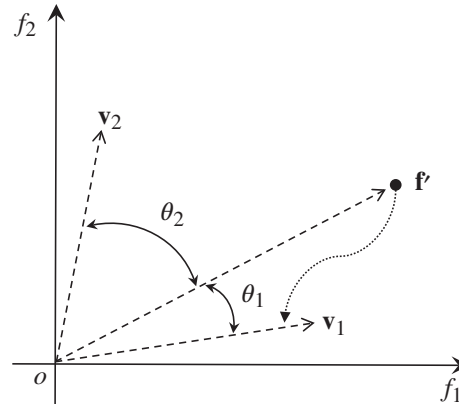

Figure 1.6 An example showing how to associate an individual with a reference vector. In this example, $\mathbf{f}'$ is a translated objective vector, $\mathbf{v}_1$ and $\mathbf{v}_2$ are two unit reference vectors. θ_1 and θ_2 are the angles between $\mathbf{f}'$ and $\mathbf{v}_1, \mathbf{v}_2$, respectively. Since $\theta_1 < \theta_2$, the candidate solution related to $\mathbf{f}'$ is associated with reference vector $\mathbf{v}_1$.

Algorithm 1.8: *UniformSampling*(N, M)

Input: N (number of vectors), M (number of objectives)
Output: V (uniform vector set)

1 $H \leftarrow \text{argmax}_h C_{h+M-1}^{M-1} \leq N$; //Number of divisions on each objective
2 $X \leftarrow$ All the $(M-1)$-combinations of $\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H+M-2}{H}\}$;
3 **for** *each* $x_{ij} \in X$ **do**
4 $\quad x_{ij} \leftarrow x_{ij} - \frac{j-1}{H}$; //$x_{ij}$ is the jth element of the ith combination
5 $V \leftarrow$ A $C_{H+M-1}^{M-1} \times M$ matrix of zeros;
6 **for** *each* $v_{ij} \in V$ **do**
7 $\quad$ **if** $j == 1$ **then**
8 $\quad\quad v_{ij} = x_{ij}$;
9 $\quad$ **else if** $j == M$ **then**
10 $\quad\quad v_{ij} = 1 - x_{i(j-1)}$;
11 $\quad$ **else**
12 $\quad\quad v_{ij} = x_{ij} - x_{i(j-1)}$;
13 **return** V;

a decision-maker can be interested; by contrast, a reference vector specifies a direction that a candidate solution is expected to converge along, probably toward a unique Pareto optimal solution on the PF, as illustrated in Fig. 1.6.

In the main loop of RVEA (Lines 4–8), the variation component is also the same as that in NSGA-II. The main differences between RVEA and NSGA-II lie in the other two key components:*environmental selection* (Line 7) and *vector adaptation* (Line 8).

The *environmental selection* (Algorithm 1.9) in RVEA is designed to perform single-objective selections inside the subpopulations comprising the candidate solutions associated with their closest reference vectors (Line 4). As the selection criterion, a scalarization function known as the angle-penalized distance (APD) is proposed to aggregate the convergence criterion and the diversity criterion:

$$d_{t,i,j} = (1 + P(\theta_{t,i,j})) \cdot \|\mathbf{f}_{t,i} - \mathbf{z}_t^{min}\|, \tag{1.3}$$

where $d_{t,i,j}$ denotes the APD value of solution $p_{t,i}$ with respect to reference vector $\mathbf{v}_{t,j}$ in the tth generation; $\|\mathbf{f}_{t,i} - \mathbf{z}_t^{min}\|$, as the convergence criterion, is the

Algorithm 1.9: *EnvironmentalSelection*(P, V, N)

Input: P (combined population), V(uniform vector set), N (population size)
Output: P (population for next generation)

```
//Objective value translation
```
$\mathbf{z}^{ideal} \leftarrow$ Minimum objective values in P;
for $i = 1, \ldots, |P|$ **do**
  $\mathbf{f}'(P_i) \leftarrow \mathbf{f}(P_i) - \mathbf{z}^{ideal}$;

```
//Population partition
```
$Associate \leftarrow$ A $1 \times |P|$ vector of zeros; `//The index of vector that each solution belongs to`
for $i = 1, \ldots, |P|$ **do**
  for $j = 1, \ldots, |V|$ **do**
    $\theta_{ij} \leftarrow \langle \mathbf{f}'(P_i), V_j \rangle$; `//Angle between the solution and the vector`
  $Associate_i \leftarrow \text{argmin}_j \theta_{ij}$;

```
//Angle-penalized distance based selection
```
$P_{next} \leftarrow \emptyset$;
for $j = 1, \ldots, |V|$ **do**
  $I \leftarrow \{i = 1, \ldots, |P| | Associate_i = j\}$;
  for *each* $i \in I$ **do**
    Calculate the angle-penalized distances using Eqs. (1.3)–(1.5);
  $k \leftarrow \text{argmin}_{i \in I} d_{ij}$;
  $P_{next} \leftarrow P_{next} \cup \{P_k\}$;
$P \leftarrow P_{next}$;
return P;

Euclidean distance from the candidate solution $p_{t,i}$ to the ideal vector $\mathbf{z}_t^{min}$; and $P(\theta_{t,i,j})$, as the diversity criterion, is a penalty function related to the angle $\theta_{t,i,j}$ between candidate solution $p_{t,i}$ and reference vector $\mathbf{v}_{t,j}$:

$$P(\theta_{t,i,j}) = M \cdot \left(\frac{t}{t_{max}}\right)^{\alpha} \cdot \frac{\theta_{t,i,j}}{\gamma_{\mathbf{v}_{t,j}}}, \tag{1.4}$$

with

$$\gamma_{\mathbf{v}_{t,j}} = \min_{i \in \{1,\ldots,N\}, i \neq j} \langle \mathbf{v}_{t,i}, \mathbf{v}_{t,j} \rangle, \tag{1.5}$$

where M is the number of objectives, t_{max} is the predefined maximum number of generations, $\gamma_{\mathbf{v}_{t,j}}$ is the smallest angle value between reference vector $\mathbf{v}_{t,j}$, and α is a user-defined parameter controlling the changing rate of the diversity criterion with respect to the number of generations. It is worth noting that the design of penalty function $P(\theta_{t,i,j})$ is out of several empirical observations. First, since there are only a limited number of candidate solutions sparsely distributed in the high-dimensional objective space, it is particularly important to enhance the convergence and diversity of the population at the early stage and late stages of the evolutionary search, respectively; hence, $P(\theta_{t,i,j})$ is designed to be linked to the generation number t. Second, due to the sparse distribution of the candidate solutions, the angles between the candidate solutions and reference vectors can vary a lot; hence, the angles are normalized into $[0, 1]$ with $\frac{\theta_{t,i,j}}{\gamma_{\mathbf{v}_{t,j}}}$. Third, the more objectives there are, the more sparsely the candidate solutions are distributed; hence, $P(\theta_{t,i,j})$ is designed to be related with the number of objectives M.

The *vector adaption* (Algorithm 1.10) in RVEA is designed to adjust the distribution of the reference vectors according to the different ranges of objectives on the PFs. For dominance-based algorithms such as the NSGA-II, normalization of objective functions does not change the partial orders of dominance relations, and thus has no influence on the selection process; by contrast, in RVEA, since the selection criterion is related to the specific objective function values of the candidate solutions, dynamically normalizing the objective values during the

Algorithm 1.10: *VectorAdaptation*(P, V, V^0)

Input: P (population), V (uniform vector set), V^0 (original vector set)
Output: V (adapted uniform vector set)

1 **if** *t is a multiple of* $0.1t_{max}$ **then**
  `//t is the current generation number and` t_{max} `is the maximum generation number`
2 $\mathbf{z}^{ideal} \leftarrow$ Minimum objective values in P;
3 $\mathbf{z}^{nadir} \leftarrow$ Maximum objective values in P;
4 **for** $i = 1, \ldots, |V|$ **do**
5 Normalize the reference vectors in P using Eq. (1.6);
6 **return** V;

evolutionary search will substantially perturb the convergence of the algorithm. Therefore, instead of normalizing the objective functions, RVEA adapts the reference vectors in the following manner:

$$\mathbf{v}_{t+1,i} = \frac{\mathbf{v}_{0,i} \circ \left(\mathbf{z}_{t+1}^{max} - \mathbf{z}_{t+1}^{min}\right)}{\|\mathbf{v}_{0,i} \circ \left(\mathbf{z}_{t+1}^{max} - \mathbf{z}_{t+1}^{min}\right)\|}, \tag{1.6}$$

where $\circ$ operator denotes the Hadamard product that element-wisely multiplies two vectors (or matrices) of the same size, $i = 1, \ldots, N$, $\mathbf{v}_{0,i}$ denotes the ith uniformly distributed reference vector, which is generated in the initialization stage (in Step 1 in Algorithm 1.7) and $\mathbf{v}_{t+1,i}$ denotes the ith adapted reference vector for the next generation $t+1$, $\mathbf{z}_{t+1}^{max} = (z_{t+1,1}^{max}, z_{t+1,2}^{max}, \ldots, z_{t+1,m}^{max})$ and $\mathbf{z}_{t+1}^{min} = (z_{t+1,1}^{min}, z_{t+1,2}^{min}, \ldots, z_{t+1,m}^{min})$ denote the nadir point and the ideal point estimated with the candidate solutions in population P_{t+1}, respectively. Besides, in order to control the frequency of the reference vector adaptation, a predefined parameter f_r is introduced.

1.3 Evolutionary Large-Scale Optimization

Adapting to the environment is an important trait of natural evolution; hence, it is intuitive to take such adaptability as a unique advantage when applying EAs – treating the problems to be optimized as *black-box*. Indeed, EAs work in a way requiring no specific information of the problems at hand; however, when facing large-scale problems, leaving all labors to EAs would probably lead to prohibitively expensive computational costs due to the enormous search space. Therefore, the literature of evolutionary large-scale optimization [61, 62] has been mainly dedicated to accelerating the search speed of EAs by transforming black-box problems to be *gray-box*, mainly on the basis of *a priori* knowledge or *a posteriori* information as mined from the problems. For such purposes, the *cooperative coevolution (CC)* provides a general approach coupled with *decision variable grouping* techniques.

1.3.1 Decision Variable Grouping

The intuitive idea of decision variable grouping is borrowed from classical operation research – *divide-and-conquer*, aiming to decompose a large-scale optimization problems into a set of subproblems of smaller scales [63]. Specifically, an optimization problem $f(\mathbf{x})$ can be decomposed into subproblems by considering the *separability* of the decision vector (i.e., vector comprising the decision variables) $\mathbf{x}$, mainly in two cases.

Ideally, an optimization problem $f(\mathbf{x})$ can be decomposed into subproblems to be optimized fully independently *iff* the original objective function $f(\mathbf{x})$ is *fully separable* with respect to any decision variable x_k related to each subproblem (Case 1):

$$\underset{\mathbf{x}}{\operatorname{argmin}}\ f(\mathbf{x}) = \left(\underset{x_k}{\operatorname{argmin}}\ f(\mathbf{x}), \underset{\forall x_j j \neq k}{\operatorname{argmin}}\ f(\mathbf{x}) \right). \tag{1.7}$$

However, the subproblems can only be optimized component-wise independently *iff* $f(\mathbf{x})$ is *partially separable* with respect to any decision variable x_k related to each subproblem (Case 2):

$$\underset{\mathbf{x}}{\operatorname{argmin}}\ f(\mathbf{x}) = \left(\underset{\mathbf{x}_1}{\operatorname{argmin}}\ f(\mathbf{x}_1, \ldots), \ldots, \underset{\mathbf{x}_m}{\operatorname{argmin}}\ f(\ldots, \mathbf{x}_m) \right), \tag{1.8}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are disjoint sub-vectors of $\mathbf{x}$, and the interactions only exist among decision variables inside each sub-vector, but the decision variables in different sub-vectors do not interact with each other.

The *separability* properties as defined above provide a theoretical basis for the development of decision variable grouping techniques. Practically, in order to check the separability of a given function, it requires *variable interaction detection* among each pair of decision variables in the decision vector:

$$\begin{aligned} &f(\mathbf{x})|_{x_i=a_2, x_j=b_1} < f(\mathbf{x})|_{x_i=a_1, x_j=b_1} \wedge \\ &f(\mathbf{x})|_{x_i=a_2, x_j=b_2} > f(\mathbf{x})|_{x_i=a_1, x_j=b_2}, \text{ with} \end{aligned} \tag{1.9}$$

$$f(\mathbf{x})|_{x_i=a_2, x_j=b_1} \triangleq f(x_1, \ldots, x_{i-1}, a_2, \ldots, x_{j-1}, b_1, \ldots, x_d),$$

where x_i and x_j are known to be *interacted* with each other, and thus not *separable*.

On the basis of separability properties coupled with variable interaction detection, a number of decision variable grouping techniques have been developed. As the most representative work along this line, the differential grouping [64] proposed to permutate each pair of decision variables according to Eq. (1.9) and checked the variable interactions with respect to a threshold ϵ; consequently, the interacted decision variables are divided into the same group to be a subproblem. Considering to the sensitivity to parameter ϵ and poor accuracy in detecting interacted variables on functions with overlapping components in differential grouping, the global differential grouping (GDG) [65] and the eXtended Differential Grouping (XDG) [66] were proposed. More recently, the improved DG (DG2) was further proposed to reduce the number of function evaluations (FEs) required for pairwise variable interaction detection [67].

Despite the theoretical soundness of differential grouping, it intrinsically suffers from the burdens of requiring a large number of FEs for pairwise variable interaction detection. By contrast, some other variable grouping techniques attempted to circumvent explicit variable interaction detection. For instance, the random

Algorithm 1.11: $(x^\star, f^\star) = \texttt{CC}(f)$

```
1  /*Main Framework of CC*/
2  P ← randomized initial population;
3  c ← randomized initial context vector;
4  //grouping stage
5  G = VariableGrouping(f);
6  //optimization stage
7  while Termination Condition is Not Satisfied do
8      for κ = 1 to |G|
9      do
10         (P, c) = Optimizer(P, c, G_κ);
11 x* = c ; f* = f(x*) ;
12 return (x*, f*);
```

static/dynamic grouping techniques proposed to randomly divide the decision variables into k even groups dynamically [68, 69].

1.3.2 Cooperative Coevolution

The cooperative coevolutionary (CC) framework was initially proposed by Potter and De Jong [70] as a general framework to optimize the subcomponents of a given problem in a collaborative manner. As introduced in the Section 1.3.1, large-scale optimization problems can be decomposed into a set of subproblems of smaller scales via decision variable grouping; hence, it is intuitively reasonable to adopt CC framework for the optimization of the subproblems as decomposed [71, 72].

As shown in Algorithm 1.11, the main framework of CC consists of two main stages: the grouping stage and the optimization stage. At the grouping stage, the original optimization problem is decomposed into a set of subproblems using the methods as introduced in Section 1.3.1

At the optimization stage, the decision variables in each nonseparable group (i.e., $\mathcal{G}_\kappa$) are iteratively optimized in a coevolutionary manner, where the optimizer can be any derivative-free optimization algorithm supporting *variable-wise* update of the decision vectors. In other words, the CC framework requires that an optimization algorithm can flexibly determine which decision variable(s) to be independent updated. As a representative example, the particle swarm optimization (PSO) has proven to be well suitable as an embedded optimizer in a CC framework [72] due to its *pseudo-gradient* characteristic.

PSO is one powerful and widely used swarm intelligence paradigm introduced by Kennedy and Eberhart in 1995 [73], originally intended for simulating social

behaviors such as bird flocking, and later simplified to be applicable for solving optimization problems. PSO iteratively maintains a swarm of particles, each of which has a pair of position and velocity. In PSO, it is assumed that each particle is able to memorize the best position found in history, i.e., the best position that has ever been found by the whole swarm, termed *global best*, and the best position that has ever been found by each particle, known as *personal best*. To find the global optimum of the optimization problem, the particles learn from the personal best and global best positions. To be specific, the learning mechanisms in the canonical PSO can be summarized as follows:

$$\begin{aligned}\mathbf{v}_i^{t+1} = \omega\mathbf{v}_i^t + c_1\mathbf{r}_1^t \circ (\mathbf{pbest}_i^t - \mathbf{x}_i^t) \\ + \ c_2\mathbf{r}_2^t \circ (\mathbf{gbest}^t - \mathbf{x}_i^t),\end{aligned} \tag{1.10}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \tag{1.11}$$

where t is the generation number, $\mathbf{v}_i^t$ and $\mathbf{x}_i^t$ represent the velocity and position of the ith particle, respectively; ω is termed inertia weight [74], c_1 and c_2 are the acceleration coefficients [75], $\mathbf{r}_1^t$ and $\mathbf{r}_2^t$ are two vectors randomly generated within $[0, 1]^n$, with n being the dimension of the search space; $\mathbf{pbest}_i^t$ and $\mathbf{gbest}^t$ denote the personal best of the ith particle and the global best of the swarm, respectively. The operator $\circ$ is the Hadamard product which element-wisely multiplies two vectors of the same size.

1.4 Summary

This chapter begins with a brief overview of how optimization problems changed along with the modernization of science, engineering, and industry. As the optimization problems became increasingly complex, researchers attempted to look into nature for possible solutions – to develop the nature-inspired optimization algorithms, e.g., EAs, a representative family.

Then, this chapter summarized different types of large-scale optimization problems, in terms of either a large number of decision variables or objectives. This book is mainly focused on the MOPs with a large number of decision variables, i.e., large-scale multi-/many-objective optimization problems (LSMOPs).

On the one hand, to tailor EAs for solving LSMOPs, the related works are highly related to MOEAs. While conventional MOEAs were developed to solve problems with relatively small-scale decision variables, the core mechanisms behind them have founded further development toward evolutionary large-scale multi-objective optimization. On the other hand, when dealing with large-scale decision variables, ideas are also borrowed from the single-objective optimization literature. Particularly, the divide-and-conquer strategy is among the most representative and effective ones to deal with large-scale decision variables. When embedded with EAs, the divide-and-conquer strategy is often

implemented inside the CC framework on top of the decision variable grouping methods.

The rest of this book is organized as follows: Chapter 2 presents some background knowledge of evolutionary large-scale multi-objective optimization; Chapters 3 and 4 present several representative EAs for general and sparse large-scale multi-objective optimization, respectively; Chapters 5–9 present several representative applications of evolutionary large-scale multi-objective optimization in areas of community detection in complex networks, logistic scheduling, power systems, radiotherapy planning, and deep learning.

References

1 F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.

2 B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: a survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–35, 2015.

3 J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.

4 J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, 2014, pp. 93–100.

5 N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

6 J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multi-objective optimization," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, 1994, pp. 82–87.

7 C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms*, vol. 93, July, 1993, pp. 416–423.

8 K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

9 E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

10 D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multi-objective optimization," in *Proceedings of International Conference on Parallel Problem Solving from Nature*, 2000, pp. 839–848.

11 D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, 2001, pp. 283–290.

12 Y. Jin, M. Olhofer, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how?" in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 1042–1049.

13 T. Murata, H. Ishibuchi, and M. Gen, "Specification of genetic search directions in cellular multi-objective genetic algorithms," in *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization*, 2001, pp. 82–95.

14 Z. Wang, Q. Zhang, M. Gong, and A. Zhou, "A replacement strategy for balancing convergence and diversity in MOEA/D," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, 2014, pp. 2132–2139.

15 K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching-based selection in evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2013.

16 K. Li, S. Kwong, Q. Zhang, and K. Deb, "Interrelationship-based selection for decomposition multi-objective optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2076–2088, 2014.

17 H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multi-objective optimization problem into a number of simple multi-objective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2013.

18 L. Chen, H.-L. Liu, C. Lu, Y.-m. Cheung, and J. Zhang, "A novel evolutionary multi-objective algorithm based on S metric selection and M2M population decomposition," in *Proceedings of the Asia Pacific Symposium on Intelligent and Evolutionary Systems*, 2015, pp. 441–452.

19 J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

20 M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proceedings of International Conference on Evolutionary Multi-Criterion Optimization*, 2005, pp. 62–76.

21 E. Zitzler and S. Künzli, "Indicator-based selection in multi-objective search," in *Proceedings of International Conference on Parallel Problem Solving from Nature*, 2004, pp. 832–842.

22 Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609–622, 2018.

23 Y. Tian, X. Zhang, R. Cheng, and Y. Jin, "A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 5222–5229.

24 B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, pp. 193–212, 1995.

25 R. B. Agrawal, K. Deb, and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 3, pp. 115–148, 1994.

26 K. Ikeda, H. Kita, and S. Kobayashi, "Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal?" in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, vol. 2, 2001, pp. 957–962.

27 D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler, "On the effects of adding objectives to plateau functions," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 591–603, 2009.

28 O. Schutze, A. Lara, and C. A. Coello Coello, "On the influence of the number of objectives on the hardness of a multiobjective optimization problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 444–455, 2010.

29 L. S. Batista, F. Campelo, F. G. Guimar aes, and J. Ramírez, "A comparison of dominance criteria in many-objective optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2011, pp. 2359–2366.

30 D. W. Corne and J. D. Knowles, "Techniques for highly multiobjective optimisation: some nondominated points are better than others," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 773–780.

31 Y. Jin and B. Sendhoff, "Connectedness, regularity and the success of local search in evolutionary multi-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003, pp. 1910–1917.

32 Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multi-objective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

33 R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

34 S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 183–195, 2010.

35 M. Li, S. Yang, and X. Liu, "Diversity comparison of Pareto front approximations in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2568–2584, 2014.

36 H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: a short review," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, 2008, pp. 2419–2426.

37 Z. He and G. G. Yen, "Ranking many-objective evolutionary algorithms using performance metrics ensemble," in *Proceedings of 2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 2480–2487.

38 M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

39 K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions," *Evolutionary Computation*, vol. 13, no. 4, pp. 501–525, 2005.

40 H. Aguirre and K. Tanaka, "Space partitioning with adaptive ϵ-ranking and substitute distance assignments: a comparative study on many-objective mnk-landscapes," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 2009, pp. 547–554.

41 J. Branke, T. Kaußler, and H. Schmeck, "Guidance in evolutionary multi-objective optimization," *Advances in Engineering Software*, vol. 32, no. 6, pp. 499–507, 2001.

42 L. S. Batista, F. Campelo, F. G. Guimaraes, and J. A. Ramírez, "Pareto cone ϵ-dominance: improving convergence and diversity in multiobjective evolutionary algorithms," in *Proceedings of the 6th International Conference Evolutionary Multi-Criterion Optimization*, 2011, pp. 76–90.

43 S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, 2013.

44 M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 348–365, 2013.

45 R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 474–494, 2012.

46 K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2013.

47 X. Zhang, Y. Tian, and Y. Jin, "A knee point driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 761–776, 2015.

48 H. K. Singh, A. Isaacs, and T. Ray, "A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 539–556, 2011.

49 D. K. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang, "Objective reduction in many-objective optimization: linear and nonlinear algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 77–99, 2012.

50 D. Brockhoff and E. Zitzler, "Objective reduction in evolutionary multi-objective optimization: theory and applications," *Evolutionary Computation*, vol. 17, no. 2, pp. 135–166, 2009.

51 I. T. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, p. 276, 2003.

52 K. Q. Weinberger and L. K. Saul, "An introduction to nonlinear dimensionality reduction by maximum variance unfolding," in *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006, pp. 1683–1686.

53 A. L. Jaimes, C. A. Coello Coello, and J. E. U. Barrientos, "Online objective reduction to deal with many-objective problems," in *Proceedings of IEEE International Conference on Evolutionary Multi-criterion Optimization*, 2009, pp. 423–437.

54 L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. Wiley, 1990.

55 X. Guo, X. Wang, M. Wang, and Y. Wang, "A new objective reduction algorithm for many-objective problems: employing mutual information and clustering algorithm," in *Proceedings of International Conference on Computational Intelligence and Security*, 2012, pp. 11–16.

56 K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2014.

57 K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner, "Approximation-guided evolutionary multi-objective optimization," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2011, p. 1198.

58 L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

59 L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, 2012.

60 R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

61 M. N. Omidvar, X. Li, and X. Yao, "A review of population-based metaheuristics for large-scale black-box global optimizationPart I," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 802–822, 2021.

62 M. N. Omidvar, X. Li, and X. Yao, "A review of population-based metaheuristics for large-scale black-box global optimizationPart II," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 823–843, 2021.

63 M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, vol. 316, pp. 419–436, 2015.

64 Y. Ling, H. Li, and B. Cao, "Cooperative co-evolution with graph-based differential grouping for large scale global optimization," in *Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 2016, pp. 95–102.

65 Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software*, vol. 42, no. 2, pp. 1–24, 2016.

66 Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 313–320.

67 M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

68 M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

69 R. Liu, J. Liu, Y. Li, and J. Liu, "A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 55, p. 100684, 2020.

70 M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of IEEE International Conference on Parallel Problem Solving from Nature*, 1994, pp. 249–257.

71 Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

72 X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2011.

73 J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

74 Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.

75 R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.