Chapter 2
What is Really Multi-objective Optimization?

2.1 Introduction

Optimization refers to the process of finding an optimal set from the set of all possible solutions for a given problems. An algorithm is normally developed call optimization algorithm to find such a solution. Regardless of the specific structure, such algorithms required to compare two solutions at some stage to decide which one is better. An objective function (often called fitness, cost, etc) is used to evaluate the merit of each solution. In case of using one objective function, the proglem is single objective and the solutions can be compared using relational operators.

On the other hard, when we want to optimize more than one objective functions, the solution cannot be compared with relations operators. In this case, we might optimizer each objectives independently, but the problem is that the objectives are often in conflict. To solve such problems using optimization algorithms there are different methods in the literature [1]. This chapter covers different classes of multi-objective optimization techniques and essential definitions in this field.

2.2 Essential Definitions

Before introducing multi-objective optimization, it is important to know how a single-objective optimization problem is formulated. In the field of optimization, most of the definitions are presented for minimization problems without the loss of generality. This means that changing relational operators to their opposite leads to formulate a maximization problem. In this book, we provide both definitions to highlight the differences. A single-objective optimization can be formulated as a minimization problem as follows:

$$Minimize: \quad f(\overrightarrow{x}) \tag{2.1}$$

S. Mirjalili and J. S. Dong, *Multi-Objective Optimization using Artificial Intelligence Techniques*, SpringerBriefs in Computational Intelligence,
https://doi.org/10.1007/978-3-030-24835-2_2

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.2}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.3}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.4}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

A single-objective optimisation can be formulated as a maximization problem as follows:

$$Maximize: \quad f(\overrightarrow{x}) \tag{2.5}$$

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.6}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.7}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.8}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

In the above formulations, there is a vector that store multiple variables (often called parameters or decision variable) store all the variables of the problem. This vector is passed into the objective function that returns a number as the results. In a multi-objective problem, however, there are more than one objective function to be called using the vector. We store those Multi-objective optimization. A multi-objective optimization can be formulated as a minimization problem as follows:

$$Minimize: \quad \overrightarrow{F(\overrightarrow{x})} = \{f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_o(\overrightarrow{x})\} \tag{2.9}$$

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.10}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.11}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.12}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

A multi-objective optimization can be formulated as a maximization problem as follows:

$$Maximize: \quad \overrightarrow{F(\overrightarrow{x})} = \{f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_o(\overrightarrow{x})\} \tag{2.13}$$

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.14}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.15}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.16}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

As discussed above, a new operator is required to compare two solutions when we are dealing with multiple objectives. This operator was proposed by Pareto called Pareto optimal dominance [2]. According to this operator, one solution is better than another if it shows equation objective values on all objectives and at least is better in one of the objectives. An example is price and quality. These two objectives are in conflict when a shopper is looking for an item. As the quality goes down the price goes down too. In reality, most shoppers are looking for the best trade-offs between these two objective. A product with high quality and high prices is not better than a similar product with low quality and low price. It depends on the personal preferences and shopping habits to chose any of them. These two products are both good when minimizing the price and maximizing the quality. In this example, however, a cheap high-quality produce is definitely better than an expensive low-quality product.

The mathematical definition of Pareto dominance and Pareto optimality, which is often called Pareto optimality is defined as follows for a minimization problem [2]:

Suppose that there are two vectors such as: $\overrightarrow{x} = (x_1, x_2, ..., x_k)$ and $\overrightarrow{y} = (y_1, y_2, ..., y_k)$. Vector $\overrightarrow{x}$ dominates vector $\overrightarrow{y}$ (denote as $\overrightarrow{x} \prec \overrightarrow{y}$) iff:

$$\forall i \in (1, 2, ..., o)$$

$$[f_i(\overrightarrow{x}) \leq f_i(\overrightarrow{y})] \wedge [\exists i \in 1, 2, ..., o : f_i(\overrightarrow{x}) < f_i(\overrightarrow{y})]$$

A solution $\overrightarrow{x} \in X$ is called Pareto-optimal iff:

$$\{\nexists \overrightarrow{y} \in X | \overrightarrow{y} \prec \overrightarrow{x}\}$$

The mathematical definition of Pareto dominance and Pareto optimality, which is often called Pareto optimality is defined as follows for a maximization problem:

Suppose that there are two vectors such as: $\overrightarrow{x} = (x_1, x_2, ..., x_k)$ and $\overrightarrow{y} = (y_1, y_2, ..., y_k)$. vector $\overrightarrow{x}$ dominates vector $\overrightarrow{y}$ (denote as $\overrightarrow{x} \succ \overrightarrow{y}$) iff:

$$\forall i \in (1, 2, ..., o)$$

$$[f_i(\overrightarrow{x}) \geq f_i(\overrightarrow{y})] \wedge [\exists i \in 1, 2, ..., o : f_i(\overrightarrow{x}) > f_i(\overrightarrow{y})]$$

A solution $\overrightarrow{x} \in X$ is called Pareto-optimal iff:

$$\{\nexists \overrightarrow{y} \in X | \overrightarrow{y} \succ \overrightarrow{x}\}$$

In the example above, there might be thousands and thousands of products with different quality and price that are all good choices for a shopper. The set of all non-dominated solutions for a given problem is called Pareto optimal solution set. This set includes the solutions that represents the best trade-offs between the objectives. The Pareto optimal set is defined as follows:

The set of all Pareto-optimal solutions for a minimization problem:

$$PS := \{\overrightarrow{x}, \overrightarrow{y} \in X | \nexists \overrightarrow{y} \prec \overrightarrow{x}\}$$

The set of all Pareto-optimal solutions for a maximization problem:

$$PS := \{\overrightarrow{x}, \overrightarrow{y} \in X | \nexists \overrightarrow{y} \succ \overrightarrow{x}\}$$

There is another set that is popular in the field of multi-objective optimization called Pareto optimal front. This set has the same number of solutions as to the Pareto optimal set. However, we store the objective values for all objectives of every solution in this set. In other words. Pareto optimal front is the projection of the Pareto optimal solution when considering the objectives only. This set is defined as follows:

$$\forall i \in (1, 2, ..., o)$$

$$PF := \{f_i(\overrightarrow{x}) | \overrightarrow{x} \in PS\}$$

To visually see how these definitions work, Fig. 2.1 is given. This figure shows the Pareto optimal fronts for a bi-objective problem. Four possible cases in which the two objective can be minimized or maximized are consider. In each case the solutions highlighted in the black regions dominate other solutions in the white areas since they show equal objective value in both objectives and better in at least one of them.

2.3 A Classification *f* Multi-objective Optimization Algorithms

There are different classifications for multi-objective optimization algorithms. Since the focus of this book is on optimization using metaheuristics and evolutionary algorithms, the most well-regarded classification in this area is discussed in this section.

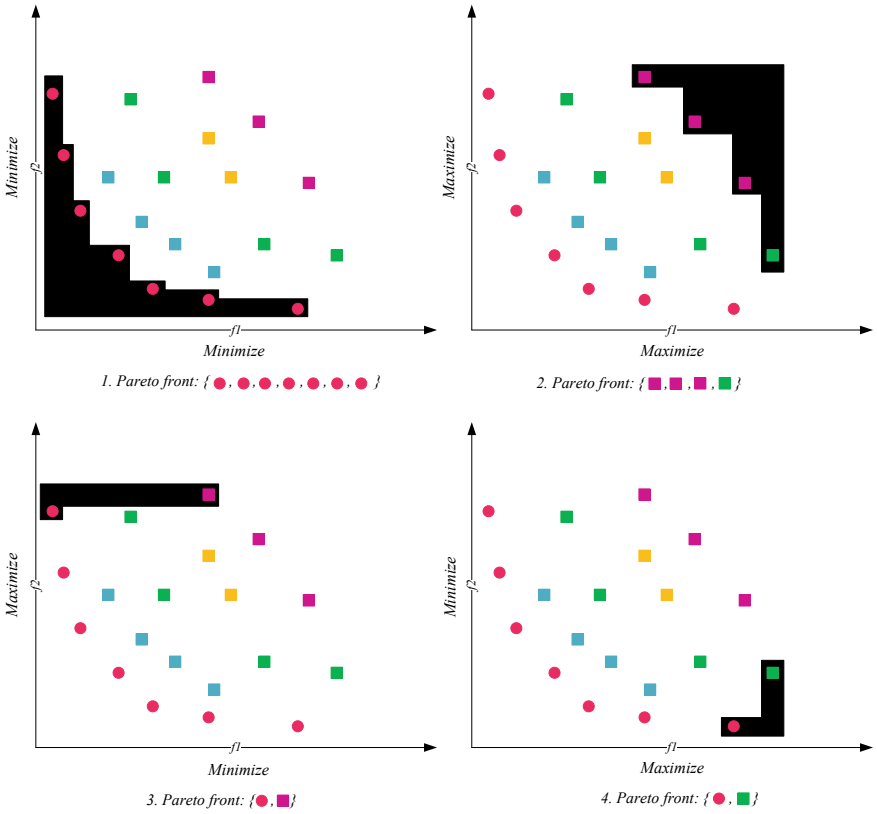

Fig. 2.1 Pareto optimal fronts for a bi-objective problem. Four possible cases in which the two objective can be minimized or maximized are considered. In each case the solutions highlighted in the black regions dominate other solutions in the white areas since they show equal objective value in both objectives and better in at least one of them

This classification is based on the time when a decision maker is involved. This is because from a decision maker perspective, one of the Pareto optimal solutions will be required for a given application. Decision making can be done before, during, or after a multi-objective optimization process. The three classes are: a priori, interactive, and a posterior methods.

In the first class [4], a decision maker is involved before starting the multi-objective optimization process. Therefore, we know how much of each objective is important to the decision maker. This will assist to narrow down the search process to the areas of the search space and objective space that the decision maker is interested in. Such methods are called a priori because we know the importance of objectives prior to the commencement of the optimization process.

In the second class [5], a decision maker is involved during the multi-objective optimization process. This type of optimization is often called human-in-the-loop

optimization. In such methods, the optimization process is periodically paused and a decision maker is required to choose the most desirable or promising solution(s) obtained so far. After feeding the algorithm with the preferred solutions, depending on the mechanism the algorithm with consider those solutions in the rest of the multi-objective optimization process.

In the last class [3], decision making is done after the optimization process. This means that the algorithm finds as many Pareto optimal solutions as possible for the problem. A decision maker the decides which solution is good for which applications. Due to the used of a decision maker after the optimization process, such methods are often called a posteriori multi-objective optimization algorithms.

These three types of multi-objective optimization are covered in the following sections.

2.4 A Priori Multi-objective Optimization

One of the most popular a priori methods is aggregation methods, in which multiple objectives are combined into a single objective using a set of weights. This kind of optimization can be formulated as a minimization problem as follows without the loss of generality:

$$Minimize: \quad f(\overrightarrow{x}) = \sum_{i=1}^{o} w_i f_i(\overrightarrow{x}) \tag{2.17}$$

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.18}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.19}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.20}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

It can be seen in the above equations that the multiple objectives are aggregated using a set of weights. These ways are defined prior to the optimization process, which is where the name a priori multi-objective optimization comes from. A decision maker shows how much each of the objectives are important with the set of weights. The main advantage of this method is its simplicity and low computational cost since a single-objective algorithm can optimize the aggregated objectives without the need to store and handle non-dominated solutions.

However, there are some drawbacks with a priori methods [6]. Firstly, the set of weight is required that might not be available if there is no decision maker. Secondly,

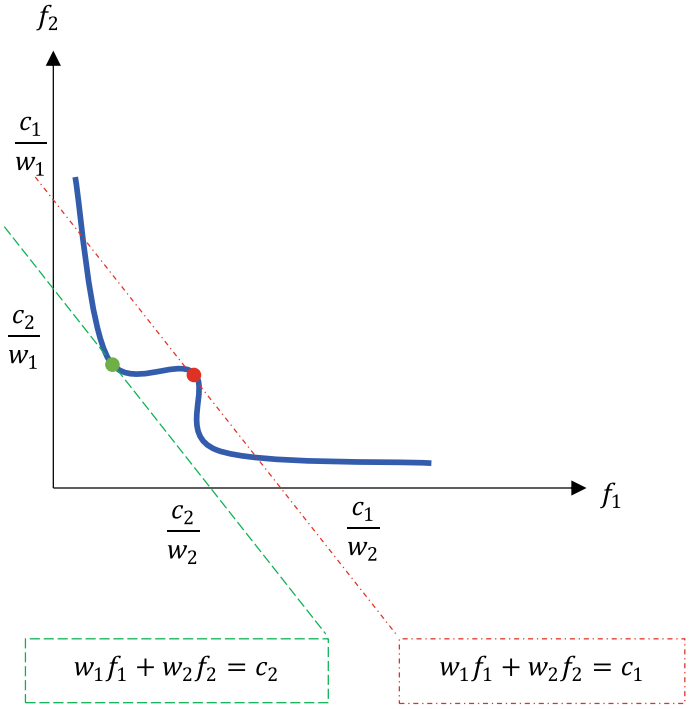

Fig. 2.2 We would like c to be as small as possible because it will lead to a smaller target value (aggregated objective). Figure 2.2 shows the intersection of the straight line and the $X(f_1)$ axis is $\frac{c}{w_1}$, and the intersection of the straight line and the $Y(f_2)$ axis is $\frac{c}{w_2}$. Therefore, smaller c values are obtained the straight line gets close to the origin. Therefore, there is always a point in the convex region that gives smaller c and considered to be a better solution in the non-convex regions. This is the main drawback of aggregation-based methods

the algorithm needs to be run multiple times to find multiple Pareto optimal solutions. Thirdly, to find the Pareto optimal solution set, the weights should be changed. In this case, even changing the weights uniformly does not guarantee finding uniformly distributed Pareto optimal solution. Finally, the non-convex regions of the Pareto optimal front cannot be obtained due to the use of positive weights. This can be seen in Fig. 2.2.

In this figure it is assumed that we want to minimize both objectives. In the aggregation-based method, we have to use the $w_1 * f_1 + w_2 * f_2$ equation to find the Pareto optimal solution. This means that we want to find the intersection of the straight line $w_1 * f_1 + w_2 * f_2 = c$ and the feasible domain.

Obviously, we would like c to be as small as possible because it will lead to a smaller target value (aggregated objective). Figure 2.2 shows the intersection of the

straight line and the $X(f_1)$ axis is $\frac{c}{w_1}$, and the intersection of the straight line and the $Y(f_2)$ axis is $\frac{c}{w_2}$. Therefore, smaller c values are obtained the straight line gets close to the origin. Therefore, there is always a point in the convex region that gives smaller c and considered to be a better solution in the non-convex regions. This is the main drawback of aggregation-based methods.

2.5 A Posteriori Multi-objective Optimization

In a posteriori multi-objective optimization algorithm, the multi-objective formulation of the problem is maintained and they are all optimized simultaneously. A multi-objective optimization when maintaining the multi-objective formulation can be formulated as a minimization problem as follows without the loss of generality.

A multi-objective optimization can be formulated as a maximization problem as follows:

$$Minimize: \quad \overrightarrow{F(\overrightarrow{x})} = \{f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_o(\overrightarrow{x})\} \tag{2.21}$$

$$Subject\ to: \quad g_i(\overrightarrow{x}) \geq 0, i = 1, 2, ..., m \tag{2.22}$$

$$h_i(\overrightarrow{x}) = 0, i = 1, 2, ..., p \tag{2.23}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, ..., n \tag{2.24}$$

where $\overrightarrow{x}$ is a vector that stores all the variables ($\overrightarrow{x} = \{x_1, x_2, x_3, ..., x_{n-1}, x_n\}$) for the problem, n is number of variables, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the *i-th* variable, and ub_i is the upper bound of the *i-th* variable.

It may be see in this formulation that we have a vector of objectives that all should be optimized. In a posteriori method Pareto optimal dominance is used to compare solutions. During the optimization process, therefore, a posteriori optimization algorithm need to store non-dominated solutions as the best solutions for the problem. There are two ultimate goals here. For one, we have to find an accurate approximation of the true Pareto optimal solutions, which is called convergence.

For another, the distribution of solutions across all objectives should be as uniform as possible, which is called coverage. The reason why coverage is important is because in a posteriori method, decision making is done after the optimization process. Therefore, a uniformly distributed Pareto optimal solutions give the decision maker a large number of different solutions to choose from for different applications and purposes.

2.6 Interactive Multi-objective Optimization

In the interactive multi-objective optimization, decision making is done during the optimization process [7]. This means that human's input is fetched to guide the search process to the regions that are of the interests of decision makers. This is why such methods are often called human-in-the-loop optimization.

In interactive optimization methods, users (including decision makers) interact with the optimization algorithm to improve the efficiency of the algorithm, enrich the optimization model, or evaluating the quality of the solution(s) obtained. In interactive multi-objective optimization, a set of random solution is first generated. The user then evaluations some or all of those solutions, provide his/her preferences. The optimization process is then continued with the preferences to find desirable solutions. This process is continued until the user confirms one or multiple of the solutions.

Interactive methods tend to be more efficient than a priori algorithms. This is because a priori method required the preferences to be defined before the optimization process, so there is no opportunity for the decision maker to adjust, adapt, or correct those preferences. In case of any changes in the weights, the whole optimization process should be re-started.

Interactive methods also have several advantages as compared to a posteriori multi-objective optimization algorithm. The set of Pareto optimal solutions used and improved in each iteration in interactive methods is smaller than that in a posteriori method. Therefore, the algorithm does not waste computational resources searching in non-promising regions of the search space.

2.7 Conclusion

This section provided preliminaries and essential definitions in the field of multi-objective optimization. The three classes of multi-objective optimization algorithms were covered as well including: a priori, a posteriori, and interactive methods.

References

1. Deb K (2014) Multi-objective optimization. In: Search methodologies. Springer, Boston, pp 403–449
2. Censor Y (1977) Pareto optimality in multiobjective problems. Appl Math Optim 4(1):41–59
3. Jaszkiewicz A, Branke J (2008) Interactive multiobjective evolutionary algorithms. In: Multi-objective optimization. Springer, Berlin, pp 179–193
4. Marler RT, Arora JS (2010) The weighted sum method for multi-objective optimization: new insights. Struct Multidiscip Optim 41(6):853–862
5. Coello CAC, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, vol 5. Springer, New York, pp 79–104

6. Jin Y, Olhofer M, Sendhoff B (2001) Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how? In: Proceedings of the 3rd annual conference on genetic and evolutionary computation. Morgan Kaufmann Publishers Inc, pp 1042–1049
7. Meignan D, Knust S, Frayret JM, Pesant G, Gaud N (2015) A review and taxonomy of interactive optimization methods in operations research. ACM Trans Inter Intell Syst (TiiS) 5(3):17