symmetry

Article

Using Shapley Values and Genetic Algorithms to Solve Multiobjective Optimization Problems

Hsien-Chung Wu

Department of Mathematics, National Kaohsiung Normal University, Kaohsiung 802, Taiwan; hcwu@nknu.edu.tw

Abstract: This paper proposes a new methodology to solve multiobjective optimization problems by invoking genetic algorithms and the concept of the Shapley values of cooperative games. It is well known that the Pareto-optimal solutions of multiobjective optimization problems can be obtained by solving the corresponding weighting problems that are formulated by assigning some suitable weights to the objective functions. In this paper, we formulated a cooperative game from the original multiobjective optimization problem by regarding the objective functions as the corresponding players. The payoff function of this formulated cooperative game involves the symmetric concept, which means that the payoff function only depends on the number of players in a coalition and is independent of the role of players in this coalition. In this case, we can reasonably set up the weights as the corresponding Shapley values of this formulated cooperative game. Under these settings, we can obtain the so-called Shapley–Pareto-optimal solution. In order to choose the best Shapley–Pareto-optimal solution, we used genetic algorithms by setting a reasonable fitness function.

Keywords: cooperative games; genetic algorithms; Pareto-optimal solutions; Shapley values; weighting problems

Citation: Wu, H.-C. Using Shapley Values and Genetic Algorithms to Solve Multiobjective Optimization Problems. *Symmetry* **2021**, *13*, 2021. https://doi.org/10.3390/sym13112021

Academic Editor: Raúl Baños Navarro

Received: 21 August 2021
Accepted: 1 October 2021
Published: 25 October 2021

1. Introduction

The purpose of an optimization problem is to search for a minimum or a maximum of a real-valued function that is also called an objective function. When the objective function is a vector-valued function instead of a real-valued function, the optimization problem turns into the so-called multiobjective optimization problem. The variables of objective functions are also called decision variables, which are usually assumed to be nonnegative variables, that is the values of the variables are assumed to be nonnegative real numbers. When the decision variables are assumed to be in a predefined search space, we have a particular kind of optimization called the constrained optimization problem.

As we mentioned above, multiobjective optimization problems consider the vector-valued objective functions, which can also be regarded as considering several conflicting objectives. The solution concepts of multiobjective optimization problems are usually based on partial orderings in which the Pareto-optimal solution is usually taken into account. In this case, the set of all Pareto-optimal solutions is frequently large in the sense that it is always an uncountable set. Solving multiobjective optimization problems usually requires the decision-makers to provide some preference relations among the set of all Pareto-optimal solutions. When several decision-makers participate, the aspect of negotiation and consensus striving among the decision-makers should be considered.

The monographs of Chankong and Haimes [1], Cohon [2], Hwang et al. [3–5], Miettinen [6], Sawaragi et al. [7], Steuer [8], and Yu [9] provided a penetrating overview of multiobjective optimization. Vincke [10] dealt with multiattribute decision analysis. Ringuest [11] considered the behavioral aspects of multiobjective optimization. Jahn [12] and Luc [13] presented the theoretical aspects of the multiobjective optimization problem. The other interesting monographs on the topic are those of Rietveld [14] and Zeleny [15,16].

The pioneering work by von Neumann and Morgenstern [17] initiated game theory in economics, which mainly deals with the behavior of players whose decisions affect each other. The topic of cooperative games is a kind of game considering coalitions. The cooperation means that players have complete freedom of communication and comprehensive information to form the different coalitions. Nash [18] studied a general two-person cooperative game. On the other hand, the monotonicity of cooperative games means that, when a game is changed such that the contributions of some player compared to all coalitions increases or stays the same, the allocation of those players should not decrease. Young [19] studied the monotonic solutions of cooperative games. The well-known Shapley value that is a unique symmetric solution is also monotonic. We may also refer to the monographs of Barron [20], Branzei et al. [21], Curiel [22], González-Díaz et al. [23], and Owen [24] for more details on the topic of game theory.

The fusion of multiobjective optimization problems and game theory has been studied by many researchers. These approaches formulated the original multiobjective optimization problems as a noncooperative or cooperative game in which the objectives were treated as the corresponding players, and the solutions of this formulated game were taken to be the solutions of the original multiobjective optimization problems. In this paper, we propose a different approach by transforming the original multiobjective optimization problems into a weighting problem in which the weights are taken to be the Shapley value of this formulated cooperative game, which can be the first attempt for solving multiobjective optimization problems.

Jing et al. [25] considered a bi-objective optimization problem in which the multibenefit allocation constraints were modeled and inspired by cooperative game theory. The ϵ-constraint approach was used to convert the bi-objective optimization problem into a single-objective optimization problem. Lokeshgupta and Sivasubramani [26] also considered a bi-objective optimization problem in which the two objective functions were treated as two players by incorporating the cooperative game. In order to generate the best compromise solution of the proposed bi-objective problem, the form of the so-called super-criterion was considered, and a mixed-integer nonlinear programming was applied to maximize the super-criterion.

Lee [27] considered a bi-objective optimization problem in which two objectives were treated as two players and suggested a noncooperative game corresponding to this bi-objective optimization problem. The Nash equilibrium was obtained from this two-player noncooperative game without using the heuristic algorithms. Li et al. [28] considered a three-objective optimization problem that was formulated as a three-player game. The best solution was obtained by using the genetic algorithm and Tabu search among the Nash equilibrium solutions. Chai et al. [29] considered a four-objective optimization problem, and Cao et al. [30] considered a bi-objective optimization problem that were solved by using the genetic algorithm. Since the selection process in the genetic algorithm is usually comparative and competitive, they adopted the noncooperative game theory to design the selection process and directly obtained the Pareto-optimal solutions without converting the four-objective optimization problem into a single-objective optimization.

Yu et al. [31] and Zhang et al. [32] considered a three-objective optimization problem in which three objectives were treated as three players and the Nash equilibrium among these three players were taken into account. Zhang et al. [32] considered the subgame perfect Nash equilibrium to be the solutions of the models, and Yu et al. [31] incorporated the genetic algorithm to obtain the solutions without converting the three-objective optimization problem into a single-objective optimization problem. Meng and Xie [33] considered a bi-objective optimization problem in which a competitive–cooperative game method was proposed to obtain the optimal preference solutions.

The approach proposed in this paper is completely different from the above approaches, where the original multiobjective optimization problem is converted into a single-objective optimization using the weighting approach in which the corresponding weights are inspired by the Shapely value of cooperative games. It is well-known that the Pareto-optimal solutions of multiobjective optimization problems can be obtained by solving the corresponding weighting problems that are formulated by assigning some suitable weights to the objective functions. There is no usual way to determine the weights for establishing the weighting problems. In this paper, we formulated a cooperative game from a multiobjective optimization problem in which the ith objective is treated as player i. The payoff function of this formulated cooperative game involves the symmetric concept, which means that the payoff function only depends on the number of players in a coalition and is independent of the role of the players in this coalition. According to the Shapley values of this formulated cooperative game, we can reasonably set up the weights for the corresponding weighting problems. Under these settings, we can obtain the so-called Shapley–Pareto-optimal solutions. Usually, the family of all Shapley–Pareto-optimal solution is large. In order to choose the best Shapley–Pareto-optimal solution, we used genetic algorithms by setting a reasonable fitness function.

Using genetic algorithms to solve the multiobjective optimization problems has been studied for a long time by referring to the monographs of Deb [34], Osyczka [35], Sakawa [36], and Tan et al. [37]. However, this paper did not intend to invoke genetic algorithms to directly solve the multiobjective optimization problems. Instead, we used genetic algorithms to obtain the best Shapley–Pareto-optimal solution from the family of all Shapley–Pareto-optimal solutions, where the so-called best Shapley–Pareto-optimal solution is based on a reasonable fitness function.

In Section 2, the concept of Shapley values and the basic properties of multiobjective optimization problems are presented. In Section 3, a multiobjective optimization is formulated as a cooperative game by considering objective functions as the corresponding players in which the payoff function involving the symmetric concept is taken into account. In Section 4, the Shapley values of the formulated cooperative game are taken to define the so-called Shapley–Pareto-optimal solution. In Section 5, a genetic algorithm is designed to find the best Shapley–Pareto-optimal solution. A numerical example is also provided to demonstrate the usefulness of the proposed methodology.

2. Formulation of the Cooperative Game

Consider the following multiobjective optimization problem:

$$\begin{array}{lll}\text{(MOP)} & \max & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_n(\mathbf{x})) \\ & \text{subject to} & \mathbf{x} \in F \subseteq \mathbb{R}^p\end{array}$$

where F is a feasible region of problem (MOP) and each f_i is a real-valued function defined on $\mathbb{R}^p$ for $i = 1, \cdots, n$. A decision vector $\mathbf{x}^* \in F$ is called a *Pareto-optimal solution* of the problem (MOP) when there does not exist another decision vector $\mathbf{x} \in F$ satisfying $f_i(\mathbf{x}) \geq f_i(\mathbf{x}^*)$ for all $i = 1, \cdots, p$ and $f_j(\mathbf{x}) > f_j(\mathbf{x}^*)$ for at least one index j.

The weighting method is usually used to obtain the Pareto-optimal solution. The weighting problem for a multiobjective optimization problem assigns weights to each objective in which the weights represent the importance of objective functions. Therefore, we can consider the following weighting problem:

$$\begin{array}{lll}\text{(WP)} & \max & \sum_{i=1}^{n} w_i \cdot f_i(\mathbf{x}) \\ & \text{subject to} & \mathbf{x} \in F \subseteq \mathbb{R}^p\end{array}$$

where $w_i \geq 0$ for all $i = 1, \cdots, n$ and:

$$w_1 + w_2 + \cdots + w_n = 1.$$

Then, we have the following well-known results.

- If $w_i > 0$ for all $i = 1, \cdots, n$, then the optimal solution of the WP is a Pareto-optimal solution of the MOP;

- The unique optimal solution of the WP is a Pareto-optimal solution of the MOP;
- Suppose that the problem (MOP) is convex. If $\mathbf{x}^*$ is a Pareto-optimal solution, then there exist nonnegative weights $w_i \geq 0$ for $i = 1, \cdots, n$ such that $\mathbf{x}^*$ is an optimal solution of the WP.

In order to obtain the Pareto-optimal solution, it suffices to solve the corresponding weighting problem. Therefore, we have many Pareto-optimal solutions according to different weighting problems.

The determination of the weights for establishing the weighting problem depends on the viewpoint of the decision-makers. This means that there is no usual way to set up the weighting problems. This paper used the Shapley value to determine the weights. The main reason is that the ith objective function f_i can be regarded as the payoff of player i. In this case, we can formulate a cooperative game such that its Shapley values are taken to be the weights for creating the corresponding weighting problem.

Consider the MOP with n objective functions f_i for $i = 1, \cdots, n$ and feasible set F. We first maximize each objective function f_i on the feasible set F and let:

$$z_i^* = \sup_{\mathbf{x} \in F} f_i(\mathbf{x}) \text{ for } i = 1, \cdots, n.$$

We may call the vector $\mathbf{z}^* = (z_1^*, z_2^*, \cdots, z_n^*)$ the *ideal objective value* of the MOP.

We associated the vector-valued objective function $\mathbf{f}$ with a cooperative game. The ith objective value f_i is regarded as the payoff of player i. The payoff function v can be predetermined by the decision-makers. Since z_i^* is the ideal payoff of player i, it follows that the payoff of player i must satisfy $v(\{i\}) \leq z_i^*$.

Let $N = \{1, \cdots, n\}$ be the set of all players, and let S be a subset of N, which is regarded as a coalition. Under this coalition, $S = \{i_1, i_2, \cdots, i_s\}$ with $s = |S|$, where $|S|$ denotes the number of players of coalition S, the payoff of this coalition S will be greater than the total payoffs of players in S. In other words, we must have:

$$v(S) \geq v(\{i_1\}) + v(\{i_2\}) + \cdots + v(\{i_s\}).$$

On the other hand, the payoff of coalition S cannot be greater than the total ideal payoffs on S. More precisely, we have:

$$v(\{i_1\}) + v(\{i_2\}) + \cdots + v(\{i_s\}) \leq v(S) \leq z_{i_1}^* + z_{i_2}^* \cdots + z_{i_s}^*. \tag{1}$$

We define a payoff function on the family of all subsets of N (i.e., on all coalitions) via the symmetric concept. Given any coalition S with $|S| \geq 2$, we define:

$$v(S) = \sum_{m=1}^{s} v(\{i_m\}) + b_s,$$

where b_s is an extra payoff that benefits from the coalition via the symmetric concept, which means that it only depends on the number of players in a coalition without taking into account the roles of the players in a coalition. For example, the extra benefit can be taken as:

$$b_s = \kappa_s \sum_{m=1}^{s} v(\{i_m\})$$

which is a discount of the original payoff $\sum_{m=1}^{s} v(\{i_m\})$ by a discount factor κ_s, where the discount factor κ_s can be regarded as a symmetric constant that is independent of the players in the coalition S with $|S| = s$ and is dependent on the number s of the players in the coalition S. In other words, the symmetric constant κ_s will be the same for $|S| = s$

regardless of the players in the coalition S. In this paper, the symmetric constant κ_s is taken as $\kappa_s = c_s/s$. Therefore, the payoff of coalition S is given by:

$$v(S) = \sum_{m=1}^{s} v(\{i_m\}) + \frac{c_s}{s}\sum_{m=1}^{s} v(\{i_m\}), \tag{2}$$

where c_s is a nonnegative constant and is independent of the players in the coalition S with $|S| = s$, which says that, under the coalition S, the extra payoff can be obtained by taking c_s multiplying the average of individual payoffs.

Since the upper bound of $v(S)$ is given in (1), the constant c_s should satisfy:

$$\sum_{m=1}^{s} v(\{i_m\}) + \frac{c_s}{s}\sum_{m=1}^{s} v(\{i_m\}) \leq \sum_{m=1}^{s} z^*_{i_m},$$

which says that

$$0 \leq c_s \leq \frac{s \cdot \sum_{m=1}^{s}\left(z^*_{i_m} - v(\{i_m\})\right)}{\sum_{m=1}^{s} v(\{i_m\})} = \frac{s \cdot \sum_{m=1}^{s} z^*_{i_m}}{\sum_{m=1}^{s} v(\{i_m\})} - s \equiv U_s(S)$$

for $s = 2, \cdots, n$, where $U_s(S)$ is dependent on coalition S with $|S| = s$. Now, we define:

$$U_s = \min\{U_s(S) : S \subseteq N \text{ with } |S| = s\}, \tag{3}$$

Then, we have:

$$0 \leq c_s \leq U_s \tag{4}$$

for $s = 2, \cdots, n$. For convenience, we also define $c_1 = 0$.

3. Shapley Values

Let $N = \{1, \cdots, n\}$ be a set of all players. Any nonempty subset $S \subseteq N$ is called a coalition. A cooperative game is an ordered pair (N, v), where the characteristic function v is a function from the family of all subsets of N into $\mathbb{R}$ satisfying $v(\emptyset) = 0$. The number $v(S)$ can be regarded as the worth of coalition S in the game (N, v).

The carrier of cooperative game (N, v) is a coalition T satisfying $v(S) = v(S \cap T)$ for any coalition $S \subseteq N$. This definition states that the player $i \notin T$ is a dummy player, that is to say, the player i has nothing to contribute to any coalitions.

Let π be a permutation of N, i.e., a one-to-one function $\pi : N \to N$. Given a coalition $S \subseteq N$ with $|S| = s$, i.e., $S = \{i_1, i_2, \cdots, i_s\}$, we write $\pi(S) = \{\pi(i_1), \pi(i_2), \cdots, \pi(i_s)\}$. Then, we can define a cooperative game (N, v^π) by $v^\pi(\pi(S)) = v(S)$, i.e., $v^\pi(S) = v(\pi^{-1}(S))$.

Given a cooperative game (N, v), we considered a corresponding vector $\boldsymbol{\phi}(v) = (\phi_1(v), \cdots, \phi_n(v))$ in which the ith component $\phi_i(v)$ is interpreted as the payoffs received by player i under an agreement. This correspondence $\boldsymbol{\phi}$ is taken to satisfy the following *Shapley axioms*:

- (S1) If S is any carrier of the game (N, v), then $\sum_{i \in S} \phi_i(v) = v(S)$;
- (S2) For any permutation π of N and $i \in N$, we have $\phi_{\pi(i)}(v^\pi) = \phi_i(v)$;
- (S3) If (N, v_1) and (N, v_2) are any cooperative games, then we have $\phi_i(u + v) = \phi_i(u) + \phi_i(v)$ for all $i \in N$.

The function $\boldsymbol{\phi}$ from the family of all cooperative games into the n-dimensional Euclidean space $\mathbb{R}^n$ defines a vector $\boldsymbol{\phi}(v)$, which is called the *Shapley value* of cooperative game (N, v).

It is well known that there exists a unique function $\boldsymbol{\phi}$ defined on the family of all cooperative games and satisfying Axioms (S1), (S2), and (S3). More precisely, for $i \in N$, we have:

$$\phi_i(v) = \sum_{\{S: i \in S \subseteq N\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{i\})) \tag{5}$$

Let (N, v) be a cooperative game formulated from the MOP with the characteristic function given in (2). In order to use the weighting method to solve the problem (MOP), the weights are determined by the vector $\mathbf{w}(v) = (w_1(v), \cdots, w_n(v))$ in which the ith component $w_i(v)$ is interpreted as the fair payoffs received by player i under an agreement. We see that the weights $\mathbf{w}$ depend on the cooperative game (N, v), where the weights $\mathbf{w}$ satisfy the following agreement that is set by the Shapley axioms:

- If S is any carrier of the game (N, v), then $\sum_{i \in S} w_i(v) = v(S)$;
- For any permutation π of N and $i \in N$, we have $w_{\pi(i)}(v^\pi) = w_i(v)$;
- If (N, v_1) and (N, v_2) are any cooperative games, then we have $w_i(v_1 + v_2) = w_i(v_1) + w_i(v_2)$ for all $i \in N$.

The function $\mathbf{w}$ by $(N, v) \mapsto \mathbf{w}(v)$ defines a vector $\mathbf{w}(v)$, which is the Shapley value of cooperative game (N, v). Therefore, given a corresponding cooperative game (N, v), we can solve the WP by using the Shapley value $\mathbf{w}(v) = (w_1(v), \cdots, w_n(v))$ as the weights. More precisely, the weights are given by the following formula:

$$w_i(v) = \sum_{\{S: i \in S \subseteq N\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{i\})). \tag{6}$$

From (2), we see that:

$$v(S) - v(S \setminus \{i\}) = v(\{i\}) + \left(\frac{c_s}{s} - \frac{c_{s-1}}{s-1}\right) \cdot \sum_{j \in S \setminus \{i\}} v(\{j\}) + \frac{c_s}{s} \cdot v(\{i\}), \tag{7}$$

where $s = |S|$.

For the subsequent discussion, we also assumed that the following conditions are satisfied:

- For all $i = 1, \cdots, n$, we assumed $z_i^* > 0$ and $0 \leq v(\{i\}) < z_i^*$. In particular, we can define $v(\{i\}) = k_i \cdot z_i^*$ for some constant $0 < k_i < 1$. Under this assumption, from (4), we see that $c_s > 0$ for all $s = 2, \cdots, n$, which also says that $v(S) \geq 0$ for all $S \subseteq N$;
- Recall that $c_1 = 0$ for convenience. For $s = 2, \cdots, n$, we assumed:

$$\frac{c_s}{s} \geq \frac{c_{s-1}}{s-1}. \tag{8}$$

Under this assumption, it is clear to see $w_i(v) \geq 0$ for all $i = 1, \cdots, n$ by referring to (6) and (7).

Now we consider the normalized weights as follows

$$\bar{w}_i(v) = \frac{w_i(v)}{\sum_{k=1}^{n} w_k(v)} \text{ for } i = 1, \cdots, n, \tag{9}$$

which says that $0 < \bar{w}_i(v) < 1$ for all $i = 1, \cdots, n$.

The weighting problem for a multiobjective programming problem assigns weights to each objective in which the weights represent the importance of the objective functions. Since we treated the multiobjective programming problem as a cooperative game, the payoffs of players correspond to the importance of the objective functions. In other words, the reasonable importance can be taken as the fair payoffs of the players. Here, we take the Shapley value $\mathbf{w}(v)$ to represent the importance.

Since the optimal solutions of the WP are Pareto-optimal solutions of the problem (MOP), when the weights are taken to be normalized Shapley values given in (9), the optimal solution of the weighting problem is then called a *Shapley–Pareto-optimal solution*.

From (2), we see that the cooperative game (N, v) depends on the nonnegative constants c_i for $i = 1, \cdots, n$, that is the payoff function v depends on the vector $\mathbf{c} = (c_1, \cdots, c_n)$. From (6) and (7), we also see that the weights and normalized weights depend on $\mathbf{c}$. Therefore, we write $w_i(v) \equiv w_i(\mathbf{c})$ and $\bar{w}_i(v) \equiv \bar{w}_i(\mathbf{c})$ for $i = 1, \cdots, n$. The purpose is to obtain the Shapley–Pareto-optimal solution by solving the following weighting problem:

$$\begin{array}{lll} \text{(WP)} & \max & \displaystyle\sum_{i=1}^{n} \bar{w}_i(\mathbf{c}) \cdot f_i(\mathbf{x}) \\ & \text{subject to} & \mathbf{x} \in F \subseteq \mathbb{R}^p \end{array}$$

The Shapley–Pareto-optimal solution is denoted by $\mathbf{x}^*(\mathbf{c})$, which depends on the vector $\mathbf{c}$. Let $\mathcal{X}$ be the set of all Shapley–Pareto-optimal solutions, i.e.,

$$\mathcal{X} = \{\mathbf{x}^*(\mathbf{c}) : 0 \leq c_s \leq U_s \text{ for } s = 1, \cdots, n\},$$

where U_s can refer to (3). In the sequel, we use the genetic algorithm to obtain the best Shapley–Pareto-optimal solution in $\mathcal{X}$ by evolving vectors $\mathbf{c}$.

4. Designing the Genetic Algorithm

The main purpose of this paper was to find a best Shapley–Pareto-optimal solution from $\mathcal{X}$ by maximizing the following objective function:

$$\eta(\mathbf{c}) = \sum_{i=1}^{n} \bar{w}_i(\mathbf{c}) \cdot f_i(\mathbf{x}^*(\mathbf{c})). \tag{10}$$

The Shapley–Pareto-optimal solution $\mathbf{x}^*(\mathbf{c})$ depends on the vector $\mathbf{c}$. We say that $\mathbf{x}^*(\mathbf{c}^*)$ is the best Shapley–Pareto-optimal solution when $\eta(\mathbf{c}^*) \geq \eta(\mathbf{c})$. Obtaining the best Shapley–Pareto-optimal solution is a hard problem. Therefore, we used the genetic algorithm to obtain the approximated best Shapley–Pareto-optimal solution. The chromosome was taken to be the vector $\mathbf{c}$ of real codes in $\mathbb{R}^n$. The purpose was to find the best chromosome according to the fitness function given in (10).

In the sequel, we present a recursive procedure to generate the nonnegative constants c_s for $s = 2, \cdots, n$ such that the inequalities (4) and (8) are satisfied, that is the nonnegative constants c_s must satisfy $c_1 = 0$,

$$0 \leq c_s \leq U_s \text{ and } \frac{c_s}{s} \geq \frac{c_{s-1}}{s-1} \tag{11}$$

for $s = 2, \cdots, n$. In other words, the chromosome $\mathbf{c}$ has the form of $\mathbf{c} = (0, c_2, \cdots, c_n)$, where c_s satisfies the inequalities (11) for $s = 2, \cdots, n$.

Proposition 1. *Suppose that each c_n is initially generated as a random number in the closed interval $[0, U_n]$, that is c_n is a uniform distribution ranging over $[0, U_n]$, where U_n is given in (3). Let:*

$$V_s = \min\left\{U_s, \left(1 - \frac{1}{s+1}\right) \cdot c_{s+1}\right\} \textit{ for } s = 2, \cdots, n-1.$$

For $s = n-1, n-2, \cdots, 2$, we generated each c_s as a random number in the closed interval $[0, V_s]$, that is c_s is a uniform distribution ranging over $[0, V_s]$ for $s = 2, \cdots, n-1$. Then, we have:

$$0 \leq c_s \leq U_s \textit{ and } \frac{c_s}{s} \geq \frac{c_{s-1}}{s-1}$$

for $s = 2, \cdots, n$.

Proof. Since $0 \leq c_s \leq V_s$ for $s = 2, \cdots, n-1$, we have $c_s \leq U_s$ for $s = 2, \cdots, n$ and:

$$c_{s-1} \leq \left(1 - \frac{1}{s}\right) \cdot c_s \text{ for } s = 3, \cdots, n,$$

which also implies:

$$\frac{c_s}{s} \geq \frac{c_{s-1}}{s-1} \text{ for } s = 3, \cdots, n. \tag{12}$$

Since $c_1 = 0$, the proof is complete. □

Remark 1. *According to Proposition 1, the nonnegative constants c_s can be randomly and sequentially generated as follows for $s = 2, \cdots, n$. We first generated c_n as a random number in the closed interval $[0, U_n]$. Then, we generated c_{n-1} as a random number in the closed interval:*

$$[0, V_{n-1}] = \left[0, \min\left\{U_{n-1}, \left(1 - \frac{1}{n}\right) \cdot c_n\right\}\right].$$

In this case, we have:

$$0 \leq c_{n-1} \leq U_{n-1} \textit{ and } \frac{c_n}{n} \geq \frac{c_{n-1}}{n-1}.$$

Similarly, we generated c_{n-2} as a random number in the closed interval:

$$[0, V_{n-2}] = \left[0, \min\left\{U_{n-2}, \left(1 - \frac{1}{n-1}\right) \cdot c_{n-1}\right\}\right].$$

In this case, we have:

$$0 \leq c_{n-2} \leq U_{n-2} \textit{ and } \frac{c_{n-1}}{n-1} \geq \frac{c_{n-2}}{n-2}.$$

Recursively, we can generate the nonnegative constants c_s satisfying:

$$0 \leq c_s \leq U_s \textit{ and } \frac{c_s}{s} \geq \frac{c_{s-1}}{s-1}$$

for $s = 2, \cdots, n$.

The next result is used for the crossover operation.

Proposition 2. *Suppose that $\widehat{\mathbf{c}} = (0, \widehat{c}_2, \cdots, \widehat{c}_n)$ and $\bar{\mathbf{c}} = (0, \bar{c}_2, \cdots, \bar{c}_n)$ are two vectors satisfying the inequalities (11). Given any $\lambda \in (0, 1)$, we considered the crossover operation $\mathbf{c} = \lambda\widehat{\mathbf{c}} + (1-\lambda)\bar{\mathbf{c}}$ with the components given by $c_1 = 0$ and $c_s = \lambda\widehat{c}_s + (1-\lambda)\bar{c}_s$ for $s = 2, \cdots, n$. Then, $\mathbf{c}$ also satisfies the inequalities (11).*

Proof. It is clear to see that $0 \leq c_s \leq U_s$ for $s = 2, \cdots, n$. Now, for $s = 2, \cdots, n$, we have:

$$\begin{aligned}
\frac{c_s}{s} &= \frac{1}{s}(\lambda\widehat{c}_s + (1-\lambda)\bar{c}_s) \\
&= \lambda \cdot \frac{\widehat{c}_s}{s} + (1-\lambda) \cdot \frac{\bar{c}_s}{s} \\
&\geq \lambda \cdot \frac{\widehat{c}_{s-1}}{s-1} + (1-\lambda) \cdot \frac{\bar{c}_{s-1}}{s-1} \\
&= \frac{1}{s-1}(\lambda\widehat{c}_{s-1} + (1-\lambda)\bar{c}_{s-1}) \\
&= \frac{c_{s-1}}{s-1}
\end{aligned}$$

This completes the proof. □

Proposition 2 says that the crossover operation regarding the chromosomes $\widehat{\mathbf{c}} = (0, \widehat{c}_2, \cdots, \widehat{c}_n)$ and $\bar{\mathbf{c}} = (0, \bar{c}_2, \cdots, \bar{c}_n)$ is taken to be the convex combination of $\widehat{\mathbf{c}}$ and $\bar{\mathbf{c}}$, where $\widehat{c}_s$ and $\bar{c}_s$ are real numbers satisfying the inequalities (11) for $s = 2, \cdots, n$. The convex combination of $\widehat{c}_s$ and $\bar{c}_s$ is given by $c_s = \lambda \widehat{c}_s + (1-\lambda)\bar{c}_s$ for some $\lambda \in (0,1)$. In this paper, the parameter λ was taken to be a random number in $(0,1)$. Proposition 2 shows that this new chromosome $\mathbf{c} = (0, c_2, \cdots, c_n)$ also satisfies the inequalities (11) for $s = 2, \cdots, n$. More precisely, the new chromosome is given by:

$$\mathbf{c} = (0, \lambda\widehat{c}_2 + (1-\lambda)\bar{c}_2, \cdots, \lambda\widehat{c}_n + (1-\lambda)\bar{c}_n). \tag{13}$$

For example, when the random number λ is 0.43215, the new chromosome is given by:

$$\mathbf{c} = (0, (0.43215)\widehat{c}_2 + (0.56785)\bar{c}_2, \cdots, (0.43215)\widehat{c}_n + (0.56785)\bar{c}_n).$$

Given a vector $\bar{\mathbf{c}} = (0, \bar{c}_2, \cdots, \bar{c}_n)$, we considered the mutation $\mathbf{c}$ of $\bar{\mathbf{c}}$ with the components given by $c_1 = 0$ and c_s that are obtained as follows for $s = 2, \cdots, n$:

- Generate a random Gaussian number with mean zero and standard deviation σ_n (i.e., a normal distribution $N(0, \sigma_n^2)$), and assign:

$$\widehat{c}_n = \bar{c}_n + N(0, \sigma_n^2) = \bar{c}_n + \sigma_n \cdot N(0,1).$$

  In this paper, the standard deviation σ_n was taken to be the following form:

$$\sigma_n = \beta \cdot \eta(\bar{\mathbf{c}}) + z_n,$$

  where β is a constant of proportionality to scale $\eta(\bar{\mathbf{c}})$ and z_n represents an offset. The new mutated individual c_n is defined by:

$$c_n = \begin{cases} \widehat{c}_n & \text{if } \widehat{c}_n \in [0, U_n] \\ U_n & \text{if } \widehat{c}_n > U_n \\ 0 & \text{if } \widehat{c}_n < 0 \end{cases}$$

  where U_n is given in (3). Then, $c_n \in [0, U_n]$;
- Let:

$$V_{n-1} = \min\left\{ U_{n-1}, \left(1 - \frac{1}{n}\right) \cdot c_n \right\}.$$

  Generate a random Gaussian number with mean zero and standard deviation σ_{n-1}, and assign:

$$\widehat{c}_{n-1} = \bar{c}_{n-1} + N(0, \sigma_{n-1}^2) = \bar{c}_{n-1} + \sigma_{n-1} \cdot N(0,1),$$

  where the standard deviation σ_{n-1} is taken to be the following form:

$$\sigma_{n-1} = \beta \cdot \eta(\bar{\mathbf{c}}) + z_{n-1}.$$

  The new mutated individual c_{n-1} is defined by:

$$c_{n-1} = \begin{cases} \widehat{c}_{n-1} & \text{if } \widehat{c}_{n-1} \in [0, V_{n-1}] \\ V_{n-1} & \text{if } \widehat{c}_{n-1} > V_{n-1} \\ 0 & \text{if } \widehat{c}_{n-1} < 0. \end{cases}$$

  Then, $c_{n-1} \in [0, V_{n-1}]$;
- Let

$$V_{n-2} = \min\left\{ U_{n-2}, \left(1 - \frac{1}{n-1}\right) \cdot c_{n-1} \right\}.$$

Generate a random Gaussian number with mean zero and standard deviation σ_{n-2}, and assign:

$$\widehat{c}_{n-2} = \bar{c}_{n-2} + N(0, \sigma_{n-2}^2) = \bar{c}_{n-2} + \sigma_{n-2} \cdot N(0,1),$$

where the standard deviation σ_{n-2} is taken to be the following form:

$$\sigma_{n-2} = \beta \cdot \eta(\bar{\mathbf{c}}) + z_{n-2}.$$

The new mutated individual c_{n-2} is defined by:

$$c_{n-2} = \begin{cases} \widehat{c}_{n-2} & \text{if } \widehat{c}_{n-2} \in [0, V_{n-2}] \\ V_{n-2} & \text{if } \widehat{c}_{n-2} > V_{n-2} \\ 0 & \text{if } \widehat{c}_{n-2} < 0. \end{cases}$$

Then, $c_{n-1} \in [0, V_{n-1}]$;

- Recursively, for $s = n-3, n-4, \cdots, 2$, we can define:

$$V_s = \min\left\{ U_s, \left(1 - \frac{1}{s+1}\right) \cdot c_{s+1} \right\}.$$

Generate a random Gaussian number with mean zero and standard deviation σ_s, and assign:

$$\widehat{c}_s = \bar{c}_s + N(0, \sigma_s^2) = \bar{c}_s + \sigma_s \cdot N(0,1),$$

where the standard deviation σ_s is taken to be the following form

$$\sigma_s = \beta \cdot \eta(\bar{\mathbf{c}}) + z_s.$$

The new mutated individual c_s is defined by:

$$c_s = \begin{cases} \widehat{c}_s & \text{if } \widehat{c}_s \in [0, V_s] \\ V_s & \text{if } \widehat{c}_s > V_s \\ 0 & \text{if } \widehat{c}_s < 0. \end{cases} \tag{14}$$

Then, $c_s \in [0, V_s]$.

Proposition 3. *Given a vector* $\bar{\mathbf{c}} = (0, \bar{c}_2, \cdots, \bar{c}_n)$ *satisfying the inequalities* (11), *we considered the mutation* $\mathbf{c}$ *of* $\bar{\mathbf{c}}$ *obtained in the way of* (14). *Then,* $\mathbf{c}$ *satisfies the inequalities* (11).

Proof. Since $c_n \in [0, U_n]$ and $c_s \in [0, V_s]$ for $s = 2, \cdots, n-1$, the argument in the proof of Proposition 1 is still valid. This completes the proof. □

Proposition 3 says that if the chromosome $\bar{\mathbf{c}} = (0, \bar{c}_2, \cdots, \bar{c}_n)$ satisfies the inequalities (11), then its mutation $\mathbf{c} = (0, c_2, \cdots, c_n)$ satisfies the inequalities (11) to keep the feasibility, where c_s is given by:

$$c_s = \begin{cases} \bar{c}_s + (\beta \cdot \eta(\bar{\mathbf{c}}) + z_s) \cdot N(0,1) & \text{if } 0 \le \bar{c}_s + (\beta \cdot \eta(\bar{\mathbf{c}}) + z_s) \cdot N(0,1) \le V_s \\ V_s & \text{if } \bar{c}_s + (\beta \cdot \eta(\bar{\mathbf{c}}) + z_s) \cdot N(0,1) > V_s \\ 0 & \text{if } \bar{c}_s + (\beta \cdot \eta(\bar{\mathbf{c}}) + z_s) \cdot N(0,1) < 0 \end{cases} \tag{15}$$

for $s = 2, \cdots, n$. Therefore, the mutation $\mathbf{c}$ of $\bar{\mathbf{c}}$ can be obtained by randomly generating a standard normal distribution $N(0,1)$.

4.1. Computational Procedure

Initially, the population size was assumed to be m. Therefore, we have m chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$. Each chromosome $\mathbf{c}^{(j)}$ is generated according to Proposition 1 and

Remark 1. More precisely, each chromosome $\mathbf{c}^{(j)}$ has the form of $\mathbf{c}^{(j)} = (0, c_2^{(j)}, \cdots, c_n^{(j)})$ for $j = 1 \cdots, m$, where $c_s^{(j)}$ satisfies the inequalities (11) given by:

$$0 \leq c_s^{(j)} \leq U_s \text{ and } \frac{c_s^{(j)}}{s} \geq \frac{c_{s-1}^{(j)}}{s-1}$$

for $s = 2, \cdots, n$. Each chromosome $\mathbf{c}^{(j)}$ is assigned a fitness value $\eta(\mathbf{c}^{(j)})$ for $j = 1, \cdots, m$. The m chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$ are ranked in descending order of their corresponding fitness values $\eta(\mathbf{c}^{(j)})$ for $j = 1, \cdots, m$, where the first one is saved to be the (initial) best fitness values, named $\bar{\eta}_0$.

The mutation is based on Proposition 3. Each chromosome $\mathbf{c}^{(j)} = (0, c_2^{(j)}, \cdots, c_n^{(j)})$ is mutated and assigned to $\mathbf{c}^{(j+m)} = (0, c_2^{(j+m)}, \cdots, c_n^{(j+m)})$ in the way of (15) for $j = 1, \cdots, m$. After this mutation step, we can obtain $2m$ chromosomes. Now, we generated a standard normal distribution $N(0,1)$ and set:

$$\sigma_s = \beta \cdot \eta(\mathbf{c}^{(j)}) + z_s.$$

From (14) or (15), the mutated chromosome is given by:

$$c_s^{(j+m)} = \begin{cases} c_s^{(j)} + \sigma_s \cdot N(0,1) & \text{if } 0 \leq c_s^{(j)} + \sigma_s \cdot N(0,1) \leq V_s \\ V_s & \text{if } c_s^{(j)} + \sigma_s \cdot N(0,1) > V_s \\ 0 & \text{if } c_s^{(j)} + \sigma_s \cdot N(0,1) < 0 \end{cases}$$

for $s = 2, \cdots, n$ and $j = 1, \cdots, m$.

After the mutation step, we have $2m$ chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m$. Then, the crossover operation is based on Proposition 2. Randomly select two chromosomes $\mathbf{c}^{(j)}$ and $\mathbf{c}^{(k)}$ for $j, k \in \{1, \cdots, 2m\}$ with $j \neq k$. In order to calculate their convex combination according to (13), we generated a random number $\lambda \in (0,1)$; the new chromosome is given by $\mathbf{c}^{(2m+1)} = \lambda \mathbf{c}^{(j)} + (1-\lambda)\mathbf{c}^{(k)}$. More precisely, we have:

$$\mathbf{c}^{(2m+1)} = \left(0, c_2^{(2m+1)}, \cdots, c_n^{(2m+1)}\right) = \left(0, \lambda c_2^{(j)} + (1-\lambda)c_2^{(k)}, \cdots, \lambda c_n^{(j)} + (1-\lambda)c_n^{(k)}\right),$$

i.e., $c_s^{(2m+1)} = \lambda c_s^{(j)} + (1-\lambda)c_s^{(k)}$ for $s = 2, \cdots, n$. After this crossover step, we have $2m+1$ chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m+1$.

Now, we have m old chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$ and $m+1$ new chromosomes $\mathbf{c}^{(m+j)}$ for $j = 1, \cdots, m+1$ that were generated from the mutation and crossover steps. Therefore, we can calculate the $m+1$ new fitness values $\eta(\mathbf{c}^{(j+m)})$ for $j = 1, \cdots, m+1$. In this case, we have in total $2m+1$ fitness values that can be used to select the m best chromosomes to be the next generation. The $2m+1$ chromosomes are ranked in descending order of their corresponding fitness values, and the first m chromosomes are treated as the m best chromosomes and saved to be the next generation. The computational procedure is presented below:

- Step 1 (initialization). The population size is assumed to be m. The initial population is determined by setting $\mathbf{c}^{(j)} = (c_1^{(j)}, \cdots, c_n^{(j)})$ such that $c_1^{(j)} = 0$, $c_n^{(j)}$ is a random number in $[0, U_n]$ for all $j = 1, \cdots, m$, and $c_s^{(j)}$ are random numbers in $[0, V_s]$ for all $s = 2, \cdots, n-1$ and $j = 1, \cdots, m$, where V_s is given in Remark 1. Then, $\mathbf{c}^{(j)}$ satisfies the inequalities (11) for $j = 1, \cdots, m$. Given each $\mathbf{c}^{(j)}$, calculate the normalized Shapley value $\bar{w}_i(\mathbf{c}^{(j)})$ according to (6) and (9) for $i = 1, \cdots, n$ and $j = 1, \cdots, m$. Each $\mathbf{c}^{(j)}$ is assigned a fitness value given by:

$$\eta(\mathbf{c}^{(j)}) = \sum_{i=1}^{n} \bar{w}_i(\mathbf{c}^{(j)}) \cdot f_i(\mathbf{x}^*(\mathbf{c}^{(j)}))$$

for $j = 1, \cdots, m$. The m chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$ are ranked in descending order of their corresponding fitness values $\eta(\mathbf{c}^{(j)})$ for $j = 1, \cdots, m$, where the first one is saved to be the (initial) best fitness values, named $\bar{\eta}_0$. Save $\mathbf{c}^{(j)}$ as an old elite $\mathbf{c}^{(*j)}$ given by $c_s^{(*j)} \leftarrow c_s^{(j)}$ for $s = 1, \cdots, n$ and $j = 1, \cdots, m$. Regarding the stopping criterion, set the tolerance ϵ, and set the maximum times of iterations as m^* for satisfying the tolerance ϵ. Set $k = 0$, which means the initial generation, and $k^* = 1$, which means the first time for satisfying the tolerance ϵ;

- Step 2 (mutation). Set $k \leftarrow k + 1$, which means the kth generation. Each $\mathbf{c}^{(j)}$ is mutated and assigned to $\mathbf{c}^{(j+m)}$ in the way of (14). Generate a random Gaussian number with mean zero and standard deviation σ_s. In this paper, the standard deviation σ_s was taken to be the following form:

$$\sigma_s = \beta \cdot \eta(\mathbf{c}^{(j)}) + z_s,$$

  where β is a constant of proportionality to scale $\eta(\mathbf{c}^{(j)})$ and z_s represents an offset. Therefore, we obtain the mutated chromosome $c_1^{(j+m)} = 0$ and $c_s^{(j+m)} \in [0, V_s]$ for $s = 2, \cdots, n$. After this step, we have $2m$ chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m$;
- Step 3 (crossover). Randomly select $\mathbf{c}^{(j)}$ and $\mathbf{c}^{(k)}$ for $j, k \in \{1, \cdots, 2m\}$ with $j \neq k$. Generate a random number $\lambda \in (0, 1)$; the new chromosome is given by $\mathbf{c}^{(2m+1)} = \lambda \mathbf{c}^{(j)} + (1 - \lambda)\mathbf{c}^{(k)}$ with components $c_s^{(2m+1)} = \lambda c_s^{(j)} + (1 - \lambda) c_s^{(k)} \in [0, U_s]$ for $s = 1, \cdots, n$. After this step, we have $2m + 1$ chromosomes $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m + 1$. Proposition 2 says that $\mathbf{c}^{(2m+1)}$ satisfies the inequalities (11);
- Step 4 (calculate new fitness). For each $\mathbf{c}^{(j+m)}$, calculate the normalized Shapley value $\bar{w}_i(\mathbf{c}^{(j+m)})$ according to (6) and (9) for $i = 1, \cdots, n$ and $j = 1, \cdots, m + 1$. Each $\mathbf{c}^{(j+m)}$ is assigned a fitness value given by:

$$\eta(\mathbf{c}^{(j+m)}) = \sum_{i=1}^{n} \bar{w}_i(\mathbf{c}^{(j+m)}) \cdot f_i(\mathbf{x}^*(\mathbf{c}^{(j+m)}))$$

  for $j = 1, \cdots, m + 1$;
- Step 5 (selection). The $m + 1$ new chromosomes $\mathbf{c}^{(j+m)}$ for $j = 1, \cdots, m + 1$ obtained from Steps 2 and 3 and m old elites $\mathbf{c}^{(*j)}$ for $j = 1, \cdots, m$ are ranked in descending order of their corresponding fitness values $\eta(\mathbf{c}^{(j+m)})$ and $\eta(\mathbf{c}^{(*j)})$. The first m chromosomes are saved to be the new elites $\mathbf{c}^{(*j)}$ for $j = 1, \cdots, m$, and the first one is saved to be the best fitness value named as $\bar{\eta}_k$ for the kth generation;
- Step 6 (stopping criterion). It may happen that $\bar{\eta}_k = \bar{\eta}_{k-1}$. In order not to be trapped in the local optimum, we proceeded with more iterations for m^* times even though $\bar{\eta}_k - \bar{\eta}_{k-1} < \epsilon$. If $0 \leq \bar{\eta}_k - \bar{\eta}_{k-1} < \epsilon$ and reach the times of iterations m^*, then the algorithm is halted and returns the Shapley–Pareto-optimal solution. Otherwise, the new elites $\mathbf{c}^{(*j)}$ for $j = 1, \cdots, m$ are copied to be the next generation $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$. Set $k^* \leftarrow k^* + 1$, and the algorithm proceeds to Step 2.

The genetic algorithm is a randomized algorithm. The final results for different runs may be different. However, in the long run, the final results will converge to a desired result. On the other hand, it can happen that the genetic algorithm can be trapped in the local optimum. In order to avoid this difficulty, we may try many runs to make sure the results are almost the same. The above Step 6 also provides a criterion to avoid being trapped in the local optimum.

4.2. Numerical Example

Consider the following multiobjective linear programming problem:

$$\begin{array}{rl}\max & 11x_2+11x_3+12x_4+9x_5+9x_6-9x_7\\ \max & 11x_1+11x_3+9x_4+12x_5+9x_6-9x_7\\ \max & 11x_1+11x_2+9x_4+9x_5+12x_6+12x_7\\ \text{subject to} & x_1+x_2+x_3+x_4+x_5+x_6+x_7=1\\ & x_i\geq 0 \text{ for } i=1,\cdots,7.\end{array}$$

Three objective functions are given by:

$$\begin{aligned}f_1(\mathbf{x})&=11x_2+11x_3+12x_4+9x_5+9x_6-9x_7\\ f_2(\mathbf{x})&=11x_1+11x_3+9x_4+12x_5+9x_6-9x_7\\ f_3(\mathbf{x})&=11x_1+11x_2+9x_4+9x_5+12x_6+12x_7\end{aligned}$$

with the feasible set

$$F=\{\mathbf{x}:x_1+x_2+x_3+x_4+x_5+x_6+x_7=1 \text{ and } x_i\geq 0 \text{ for } i=1,\cdots,7\}.$$

We first calculated the ideal objective value $\mathbf{z}^*=(z_1^*,z_2^*,z_3^*)$ given by:

$$z_1^*=\sup_{\mathbf{x}\in F} f_1(\mathbf{x})=z_2^*=\sup_{\mathbf{x}\in F} f_2(\mathbf{x})=z_3^*=\sup_{\mathbf{x}\in F} f_3(\mathbf{x})=12$$

with the corresponding optimal solutions given by:

$$\mathbf{x}^{(*1)}=(0,0,0,1,0,0,0,0),\quad \mathbf{x}^{(*2)}=(0,0,0,0,0,1,0,0) \text{ and } \mathbf{x}^{(*3)}=(0,0,0,0,0,0,0,1).$$

According to the above settings, we have $N=\{1,2,3\}$ representing three players. Now, we take:

$$v(\{1\})=0.5\cdot z_1^*,\quad v(\{2\})=0.6\cdot z_2^* \text{ and } v(\{3\})=0.7\cdot z_3^*.$$

For $s=|S|\geq 2$, according to (2), we have:

$$v(S)=\sum_{m=1}^{s} v(\{i_m\})+\frac{c_s}{s}\sum_{m=1}^{s} v(\{i_m\}).$$

More precisely, we obtain:

$$\begin{aligned}v(\{1,2\})&=\left(1+\frac{c_2}{2}\right)(v(\{1\})+v(\{2\}))=\left(1+\frac{c_2}{2}\right)(0.5\cdot z_1^*+0.6\cdot z_2^*)\\ v(\{1,3\})&=\left(1+\frac{c_2}{2}\right)(v(\{1\})+v(\{3\}))=\left(1+\frac{c_2}{2}\right)(0.5\cdot z_1^*+0.7\cdot z_3^*)\\ v(\{2,3\})&=\left(1+\frac{c_2}{2}\right)(v(\{2\})+v(\{3\}))=\left(1+\frac{c_2}{2}\right)(0.6\cdot z_2^*+0.7\cdot z_3^*)\\ v(\{1,2,3\})&=\left(1+\frac{c_3}{3}\right)(v(\{1\})+v(\{2\})++v(\{3\}))=\left(1+\frac{c_3}{3}\right)(0.5\cdot z_1^*+0.6\cdot z_2^*+0.7\cdot z_3^*).\end{aligned}$$

By referring to (3), we have:

$$U_2(\{1,2\}) = \frac{2 \cdot \left(z_1^* + z_2^*\right)}{v(\{1\}) + v(\{2\})} - 2 = \frac{2 \cdot \left(z_1^* + z_2^*\right)}{0.5 \cdot z_1^* + 0.6 \cdot z_2^*} - 2 = \frac{4}{1.1} - 2$$
$$U_2(\{1,3\}) = \frac{2 \cdot \left(z_1^* + z_3^*\right)}{v(\{1\}) + v(\{3\})} - 2 = \frac{2 \cdot \left(z_1^* + z_3^*\right)}{0.5 \cdot z_1^* + 0.7 \cdot z_3^*} - 2 = \frac{4}{1.2} - 2$$
$$U_2(\{2,3\}) = \frac{2 \cdot \left(z_2^* + z_3^*\right)}{v(\{2\}) + v(\{3\})} - 2 = \frac{2 \cdot \left(z_2^* + z_3^*\right)}{0.6 \cdot z_2^* + 0.7 \cdot z_3^*} - 2 = \frac{4}{1.3} - 2.$$

Therefore, we obtain:

$$U_2 = \min\{U_2(\{1,2\}), U_2(\{1,3\}), U_2(\{2,3\})\} = \frac{4}{1.3} - 2 = \frac{14}{13}.$$

We also have:

$$\begin{aligned} U_3 = U_3(\{1,2,3\}) &= \frac{3 \cdot \left(z_1^* + z_2^* + z_3^*\right)}{v(\{1\}) + v(\{2\}) + v(\{3\})} - 3 \\ &= \frac{3 \cdot \left(z_1^* + z_2^* + z_3^*\right)}{0.5 \cdot z_1^* + 0.6 \cdot z_2^* + 0.7 \cdot z_3^*} - 3 = \frac{9}{1.8} - 3 = 2. \end{aligned}$$

The detailed computational procedure is presented below:

- Step 1 (initialization). The population size is assumed to be $m = 20$. The initial population is determined by setting $\mathbf{c}^{(j)} = (c_1^{(j)}, c_2^{(j)}, c_3^{(j)})$ such that $c_1^{(j)} = 0$, $c_3^{(j)}$ is a random number in $[0, U_3] = [0, 2]$ for all $j = 1, \cdots, 20$, and $c_2^{(j)}$ are random numbers in $[0, V_2]$ for all $j = 1, \cdots, 20$, where V_2 refers to Remark 1 and is given by:

$$V_2 = \min\left\{U_2, \left(1 - \frac{1}{3}\right) \cdot c_3^{(j)}\right\} = \min\left\{\frac{14}{13}, \frac{2 \cdot c_3^{(j)}}{3}\right\}.$$

Then, $\mathbf{c}^{(j)}$ satisfies the inequalities (11) for $j = 1, \cdots, m$. Given each $\mathbf{c}^{(j)}$, according to (6), we calculate:

$$w_i(\mathbf{c}^{(j)}) \equiv w_i(v) = \sum_{\{S: i \in S \subseteq N\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{i\})).$$

More precisely, we have:

$$\begin{aligned} w_1(\mathbf{c}^{(j)}) &= \sum_{\{S: S=\{1\},\{1,2\},\{1,3\},\{1,2,3\}\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{1\})) \\ &= \frac{0!2!}{3!} \cdot (v(\{1\}) - v(\emptyset)) + \frac{1!1!}{3!} \cdot (v(\{1,2\}) - v(\{2\})) \\ &\quad + \frac{1!1!}{3!} \cdot (v(\{1,3\}) - v(\{3\})) + \frac{2!0!}{3!} \cdot (v(\{1,2,3\}) - v(\{2,3\})) \end{aligned}$$

$$\begin{aligned} w_2(\mathbf{c}^{(j)}) &= \sum_{\{S: S=\{2\},\{1,2\},\{2,3\},\{1,2,3\}\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{2\})) \\ &= \frac{0!2!}{3!} \cdot (v(\{2\}) - v(\emptyset)) + \frac{1!1!}{3!} \cdot (v(\{1,2\}) - v(\{1\})) \\ &\quad + \frac{1!1!}{3!} \cdot (v(\{2,3\}) - v(\{3\})) + \frac{2!0!}{3!} \cdot (v(\{1,2,3\}) - v(\{1,3\})) \end{aligned}$$

$$
\begin{aligned}
w_3(\mathbf{c}^{(j)}) &= \sum_{\{S:S=\{3\},\{1,3\},\{2,3\},\{1,2,3\}\}} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \cdot (v(S) - v(S \setminus \{3\})) \\
&= \frac{0!2!}{3!} \cdot (v(\{3\}) - v(\emptyset)) + \frac{1!1!}{3!} \cdot (v(\{1,3\}) - v(\{1\})) \\
&\quad + \frac{1!1!}{3!} \cdot (v(\{2,3\}) - v(\{2\})) + \frac{2!0!}{3!} \cdot (v(\{1,2,3\}) - v(\{1,2\}))
\end{aligned}
$$

Then, according to (9), we also calculate the normalized Shapley value:

$$\bar{w}_i(\mathbf{c}^{(j)}) = \frac{w_i(\mathbf{c}^{(j)})}{w_1(\mathbf{c}^{(j)}) + w_2(\mathbf{c}^{(j)}) + w_3(\mathbf{c}^{(j)})}$$

for $i = 1,2,3$ and $j = 1, \cdots, m$. Each $\mathbf{c}^{(j)}$ is assigned a fitness value given by:

$$\eta(\mathbf{c}^{(j)}) = \bar{w}_1(\mathbf{c}^{(j)}) \cdot f_1(\mathbf{x}^*(\mathbf{c}^{(j)})) + \bar{w}_2(\mathbf{c}^{(j)}) \cdot f_2(\mathbf{x}^*(\mathbf{c}^{(j)})) + \bar{w}_3(\mathbf{c}^{(j)}) \cdot f_3(\mathbf{x}^*(\mathbf{c}^{(j)}))$$

for $j = 1, \cdots, m$. The m individuals $\mathbf{c}^{(j)}$ for $j = 1, \cdots, m$ are ranked in descending order of their corresponding fitness values $\eta(\mathbf{c}^{(j)})$ for $j = 1, \cdots, m$, where the first one is saved to be the (initial) best fitness values, named $\bar{\eta}_0$. Set $k = 0$, $m^* = 20$, $k^* = 1$, and the tolerance $\epsilon = 10^{-6}$. Save $\mathbf{c}^{(j)}$ as an elite $\mathbf{c}^{(*j)}$ given by $c_s^{(*j)} \leftarrow c_s^{(j)}$ for $s = 1, \cdots, n$ and $j = 1, \cdots, m$;

- Step 2 (mutation). Set $k \leftarrow k+1$, which means the kth generation. Each $\mathbf{c}^{(j)} = (0, c_2^{(j)}, c_3^{(j)})$ is mutated and assigned to $\mathbf{c}^{(j+m)} = (0, c_2^{(j+m)}, c_3^{(j+m)})$ in the way of (14). Generate a random Gaussian number with mean zero and standard deviation σ_3, and assign:

$$\widehat{c}_3^{(j+m)} = c_3^{(j+m)} + N(0, \sigma_3^2) = c_3^{(j+m)} + \sigma_3 \cdot N(0,1).$$

The new mutated individual $c_3^{(j+m)}$ is defined by:

$$c_3^{(j+m)} = \begin{cases} \widehat{c}_3^{(j+m)} & \text{if } \widehat{c}_3^{(j+m)} \in [0, U_3] = [0,2] \\ U_3 = 2 & \text{if } \widehat{c}_3^{(j+m)} > U_3 = 2 \\ 0 & \text{if } \widehat{c}_3^{(j+m)} < 0. \end{cases}$$

Then, $c_3^{(j+m)} \in [0, U_3] = [0,2]$. Let:

$$V_2 = \min\left\{U_2, \left(1 - \frac{1}{3}\right) \cdot c_3^{(j+m)}\right\}.$$

Generate a random Gaussian number with mean zero and standard deviation σ_2, and assign:

$$\widehat{c}_2^{(j+m)} = c_2^{(j+m)} + N(0, \sigma_2^2) = c_2^{(j+m)} + \sigma_2 \cdot N(0,1).$$

The new mutated individual $c_2^{(j+m)}$ is defined by:

$$c_2^{(j+m)} = \begin{cases} \widehat{c}_2^{(j+m)} & \text{if } \widehat{c}_2^{(j+m)} \in [0, V_2] \\ V_2 & \text{if } \widehat{c}_2^{(j+m)} > V_2 \\ 0 & \text{if } \widehat{c}_2^{(j+m)} < 0. \end{cases}$$

Then, $c_2^{(j+m)} \in [0, V_2]$. For $s = 2,3$, the standard deviation σ_s is taken to be the following form:

$$\sigma_s = \beta \cdot \eta(\mathbf{c}^{(j)}) + z_s,$$

where β is a constant of proportionality to scale $\eta(\mathbf{c}^{(j)})$ and z_s represents an offset. After this step, we have $2m$ individuals $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m$;

- Step 3 (crossover). Randomly select:

$$\mathbf{c}^{(j)} = (0, c_2^{(j)}, c_3^{(j)}) \text{ and } \mathbf{c}^{(k)} = (0, c_2^{(k)}, c_3^{(k)})$$

  for $j, k \in \{1, \cdots, 2m\}$ with $j \neq k$. Generate a random number $\lambda \in (0,1)$; the new individual is given by $\mathbf{c}^{(2m+1)} = \lambda \mathbf{c}^{(j)} + (1-\lambda)\mathbf{c}^{(k)}$ with components $c_s^{(2m+1)} = \lambda c_s^{(j)} + (1-\lambda) c_s^{(k)}$ for $s = 2, 3$. After this step, we have $2m+1$ individuals $\mathbf{c}^{(j)}$ for $j = 1, \cdots, 2m+1$;
- Step 4 (calculate new fitness). For each $\mathbf{c}^{(j+m)} = (0, c_2^{(j+m)}, c_3^{(j+m)})$, we calculate the normalized Shapley value $\bar{w}_i(\mathbf{c}^{(j+m)})$ according to (6) and (9) for $i = 1, 2, 3$ and $j = 1, \cdots, m+1$. Each $\mathbf{c}^{(j+m)}$ is assigned a fitness value given by:

$$\begin{aligned}\eta(\mathbf{c}^{(j+m)}) = {} & \bar{w}_1(\mathbf{c}^{(j+m)}) \cdot f_1(\mathbf{x}^*(\mathbf{c}^{(j+m)})) + \bar{w}_2(\mathbf{c}^{(j+m)}) \cdot f_2(\mathbf{x}^*(\mathbf{c}^{(j+m)})) \\ & + \bar{w}_3(\mathbf{c}^{(j+m)}) \cdot f_3(\mathbf{x}^*(\mathbf{c}^{(j+m)}))\end{aligned}$$

  for $j = 1, \cdots, m+1$;
- Step 5 (selection). The $m+1$ new individuals $\mathbf{c}^{(j+m)} = (0, c_2^{(j+m)}, c_3^{(j+m)})$ for $j = 1, \cdots, m+1$ obtained from Steps 2 and 3 and m old elites $\mathbf{c}^{(*j)} = (0, c_2^{(*j)}, c_3^{(*j)})$ for $j = 1, \cdots, m$ are ranked in descending order of their corresponding fitness values $\eta(\mathbf{c}^{(j+m)})$ and $\eta(\mathbf{c}^{(*j)})$. The first m individuals are saved to be the new elites $\mathbf{c}^{(*j)} = (0, c_2^{(*j)}, c_3^{(*j)})$ for $j = 1, \cdots, m$, and the first one is saved to be the best fitness value named as $\bar{\eta}_k$ for the kth generation;
- Step 6 (stopping criterion). If $0 \leq \bar{\eta}_k - \bar{\eta}_{k-1} < \epsilon$ and reach the times of iterations m^*, then the algorithm is halted and returns the Shapley–Pareto-optimal solution. Otherwise, the new elites $\mathbf{c}^{(*j)} = (0, c_2^{(*j)}, c_3^{(*j)})$ for $j = 1, \cdots, m$ are copied to be the next generation $\mathbf{c}^{(j)} = (0, c_2^{(j)}, c_3^{(j)})$ for $j = 1, \cdots, m$. Set $k^* \leftarrow k^* + 1$, and the algorithm proceeds to Step 2.

The computer code was implemented using Microsoft Excel VBA in which a built-in optimization tool can be used. Since this is a randomized algorithm, we tried many runs in order to obtain the convergent results. Therefore, after many runs, the best fitness value was around 10.16666411, and the best Shapley–Pareto-optimal solution was:

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 0, 0, 0, 0, 1, 0).$$

We also remark that this simple numerical example can also be solved by using the other heuristic algorithms such as ant colony optimization, artificial immune systems, particle swarm optimization, simulated annealing, Tabu search etc. The purpose of this paper was not to provide a new genetic algorithm to compare the efficiency among the different heuristic algorithms. The genetic algorithm adopted in this paper was the standard one. Therefore, its efficiency can be realized from the existing article. The main purpose of this paper was to propose a new methodology to solve the multiobjective optimization problems by incorporating the Shapley value of a formulated cooperative game and solving its corresponding single-objective weighting problem by using the well-known numerical techniques of optimization for the single-objective function to collect a family of so-called Shapley–Pareto-optimal solutions. The intention of the genetic algorithms adopted in this paper was to obtain the best Shapley–Pareto-optimal solution from the large set of Shapley–Pareto-optimal solutions. In other words, other kinds of heuristic algorithms can also be used to obtain the best Shapley–Pareto-optimal solution.

5. Conclusions

A new methodology by applying the Shapley values of cooperative games was proposed to solve the multiobjective optimization problems. There are many ways that have been proposed to solve the multiobjective optimization problems in the literature. One efficient way is to convert a multiobjective optimization problem into a single-objective optimization problem by summing all the objective functions as a single objective with some suitable weights. Usually, those weights are determined by decision-makers by intuition. In this paper, we adopted the Shapley value of a formulated cooperative game to be the weights of this weighting problem, which can avoid the biased assignment of weights directly determined by the decision-makers.

We can solve the single-objective weighting problem to obtain the so-called Shapley–Pareto-optimal solution by using the well-known numerical techniques of optimization for the single-objective function. For example, in the linear case, this single-objective weight problem will be a linear programming problem that can be solved by using the simplex method. Since the payoff function of this formulated cooperative game depends on the symmetric constant $\kappa_s = c_s/s$ for $s = 1, \cdots, n$, as shown in (2), this means that the different symmetric constants will determine the different payoff functions, which also obtain the different Shapley values. In other words, the Shapley–Pareto-optimal solution will depend on the vector $\mathbf{c} = (c_1, \cdots, c_n)$ of nonnegative constants, which also means that we may obtain a large set of Shapley–Pareto-optimal solutions. In order to obtain the best Shapley–Pareto-optimal solution from the set of Shapley–Pareto-optimal solutions, a genetic algorithm was adopted in this paper by evolving the nonnegative vector $\mathbf{c}$ of constants.

It is well known that genetic algorithms can converge to the desired solution with probability one. In other words, we can obtain the approximated optimal solutions in the long run. The numerical example presented in this paper considered a three-objective linear programming problem with one constraint and seven decision variables. Although only one constraint was considered, this does not mean that this numerical example is too simple to demonstrate the methodology proposed in this paper. The reason is that seven decision variables were taken into account in this numerical example. The difficulty of optimization problems always depends on the number of decision variables rather than the number of constraints.

Although the genetic algorithm was adopted in this paper to obtain the best Shapley–Pareto-optimal solution from the set of Shapley–Pareto-optimal solutions, the other heuristic algorithms such as ant colony optimization, artificial immune systems, particle swarm optimization, simulated annealing, tabu search, etc., can also be used to obtain the best Shapley–Pareto-optimal solutions. The purpose of this paper was not to provide a new genetic algorithm to compare the efficiency among the different heuristic algorithms. The main purpose of this paper was to propose a new methodology to solve the multiobjective optimization problems by incorporating the Shapley value of a formulated cooperative game to obtain the best Shapley–Pareto-optimal solution using the genetic algorithm. The main issue of the proposed genetic algorithm in this paper was to provide a recursive procedure to generate the vector $\mathbf{c} = (c_1, \cdots, c_n)$ of nonnegative constants, as shown in Proposition 1, such that they were feasible to be used in the proposed genetic algorithms. The genetic algorithm adopted in this paper was the standard one. Its efficiency compared with the existing heuristic algorithms can be realized from the literature. In the future research, we may design a new genetic algorithm and compare its efficiency with the existing heuristic algorithms using statistical analysis.

This paper considered the Shapley values of cooperative games. There are many other solution concepts of cooperative games that can also be used to set up the weights of the single-objective weighting problem, which can be the future research. On the other hand, the theory of noncooperative games is another research topic of game theory, which can also be used to set up the weighting problems in the future research.

Funding: This research received no external funding

Conflicts of Interest: The author declares no conflict of interest.

References

1. Chankong, V.; Haimes, Y.Y. *Multiobjective Decision Making Theory and Methodology*; Elsevier Science Publishing Co.: New York, NY, USA, 1983.
2. Cohon, J.L. *Multiobjective Programming and Planning*; Academic Press: New York, NY, USA, 1978.
3. Hwang, C.-L.; Lin, M.-J. *Group Decision Making under Multiple Criteria: Methods and Applications*; Lecture Notes in Economics and Mathematical Systems 281; Springer: New York, NY, USA, 1987.
4. Hwang, C.-L.; Masud, A.S.M. *Multiple Objective Decision Making-Methods and Applications: A State-of-the-Art Survey*; Lecture Notes in Economics and Mathematical Systems 164; Springer: Berlin/Heidelberg, Germany, 1979.
5. Hwang, C.L.; Yoon, K. *Multiple Attribute Decision Making Methods and Applications: A State-of-the-Art Survey*; Lecture Notes in Economics and Mathematical Systems 186; Springer: Berlin/Heidelberg, Germany, 1981.
6. Miettinen, K.M. *Nonlinear Multiobjective Optimization*; Kluwer Academic Publishers: New York, NY, USA, 1998.
7. Sawaragi, Y.; Nakayama, H.; Tanino, T. *Theory of Multiobjective Optimization*; Academic Press, Inc.: Orlando, FL, USA, 1985.
8. Steuer, R.E. *Multiple Criteria Optimization: Theory, Computation, and Applications*; John Wiley & Sons: New York, NY, USA, 1986.
9. Yu, P.L. *Multiple-Criteria Decision Making Concepts, Techniques, and Extensions*; Plenum Press: New York, NY, USA, 1985.
10. Vincke, P. *Multicriteria Decision-Aid*; John Wiley & Sons: New York, NY, USA, 1992.
11. Ringuest, J.L. *Multiohjective Optimization: Behavioral and Computational Considerations*; Kluwer Academic Publisher: New York, NY, USA, 1992.
12. Jahn, J. *Mathematical Vector Optimization in Partially Ordered Linear Spaces*; Methoden und Verfahren der Mathematischen Physik, Band 31; Verlag Peter Lang GmbH: Frankfurt am Main, Germany, 1986.
13. Luc, D.T. *Theory of Vector Optimization*; Lecture Notes in Economics and Mathematical Systems 319; Springer: Berlin/Heidelberg, Germany, 1989.
14. Rietveld, P. *Multiple Objective Decision Methods and Regional Planning*; North-Holland Publishing Company: New York, NY, USA, 1980.
15. Zeleny, M. *Linear Multiobjective Programming*; Lecture Notes in Economics and Mathematical Systems 95; Springer: Berlin/Heidelberg, Germany, 1974.
16. Zeleny M. *Multiple Criteria Decision Making*; McGraw-Hill: New York, NY, USA, 1982.
17. von Neumann, J.; Morgenstern, O. *Theory of Games and Economic Behavior*; Princeton University Press: Princeton, NJ, USA, 1944.
18. Nash, J.F. Two-Person Cooperative Games. *Econometrica* **1953**, *21*, 128–140.
19. Young, H.P. Monotonic Solutions of Cooperative Games. *Int. J. Game Theory* **1985**, *14*, 65–72.
20. Barron, E.N. *Game Theory: An Introduction*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
21. Branzei, R.; Dimitrov, D.; Tijs, S. *Models in Cooperative Game Theory*; Springer: Berlin/Heidelberg, Germany, 2008.
22. Curiel, I. *Cooperative Game Theory and Applications: Cooperative Games Arising from Combinatorial Optimization Problems*; Kluwer Academic Publishers: New York, NY, USA, 1997.
23. González-Díaz, J.; García-Jurado, I.; Fiestras-Janeiro, M.G. *An Introductory Course on Mathematical Game Theory*; American Mathematical Society: Providence, RI, USA, 2010.
24. Owen, G. *Game Theory*, 3rd ed.; Academic Press: New York, NY, USA, 1995.
25. Jing, R.; Wang, M.; Liang, H.; Wang, X.; Li, N.; Shah, N.; Zhao, Y. Multi-objective optimization of a neighborhood-level urban energy network: Considering Game-theory inspired multi-benefit allocation constraints. *Appl. Energy* **2018**, *231*, 534–548.
26. Lokeshgupta, B.; Sivasubramani, S. Cooperative game theory approach for multiobjective home energy management with renewable energy integration. *IET Smart Grid* **2019**, *2*, 34–41.
27. Lee, C.-S. Multi-objective game-theory models for conflict analysis in reservoir watershed management. *Chemosphere* **2012**, *87*, 608–613.
28. Li, X.; Gao, L.; Li, W. Application of game theory based hybrid algorithm for multiobjective integrated process planning and scheduling. *Expert Syst. Appl.* **2012**, *39*, 288–297.
29. Chai, R.; Savvarisa, A.; Tsourdosa, A.; Chai, S. Multi-objective trajectory optimization of Space Manoeuvre Vehicle using adaptive differential evolution and modified game theory. *Acta Astronaut.* **2017**, *136*, 273–280.
30. Cao, R.; Coit, D.W.; Hou, W.; Yang, Y. Game theory based solution selection for multiobjective redundancy allocation in interval-valued problem parameters. *Reliab. Eng. Syst. Saf.* **2020**, *199*, 106932.
31. Yu, Y.; Zhao, R.; Zhang, J.; Yang, D.; Zhou, T. Multi-objective game theory optimization for balancing economic, social and ecological benefits in the Three Gorges Reservoir operation. *Environ. Res. Lett.* **2021**, *16*, 085007
32. Zhang, Y.; Wang, J.; Liu, Y. Game theory based real-time multiobjective flexible job shop scheduling considering environmental impact. *J. Clean. Prod.* **2017**, *167*, 665–679.
33. Meng, R.; Xie, N.-G. A Competitive-Cooperative Game Method for Multi-Objective Optimization Design of a Horizontal Axis Wind Turbine Blade. *IEEE Access* **2019**, *7*, 155748–155758.
34. Deb, K. *Multiobjective Optimization Using Evolutionary Algorithms*; John Wiley & Sons: New York, NY, USA, 2001.
35. Osyczka, A. *Evolutionary Algorithms for Single and Multicriteria Design Optimization*; Studies in Fuzziness and Soft Computing 100; Physica-Verlag: New York, NY, USA, 2002.
36. Sakawa, M. *Genetic Algorithms and Fuzzy Multiobjective Optimization*; Kluwer Academic Publishers: New York, NY, USA, 2002.
37. Tan, K.C.; Khor, E.F.; Lee, T.H. *Multiobjective Evolutionary Algorithms and Applications*; Springer: New York, NY, USA, 2005.