Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1203248, 13 pages
https://doi.org/10.1155/2020/1203248

Research Article

A New Genetic Algorithm Encoding for Coalition Structure Generation Problems

Juan Pablo Contreras,[1] Paul Bosch ,[2] Mauricio Varas,[2] and Franco Basso[3,4]

[1]*Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile*
[2]*Facultad de Ingeniería, Universidad del Desarrollo, Santiago, Chile*
[3]*School of Industrial Engineering, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile*
[4]*Instituto Sistemas Complejos de Ingeniería (ISCI), Santiago, Chile*

Correspondence should be addressed to Paul Bosch; pbosch@udd.cl

Received 3 December 2019; Revised 3 March 2020; Accepted 14 March 2020; Published 13 April 2020

Academic Editor: Mauro Gaggero

Genetic algorithms have proved to be a useful improvement heuristic for tackling several combinatorial problems, including the coalition structure generation problem. In this case, the focus lies on selecting the best partition from a discrete set. A relevant issue when designing a Genetic algorithm for coalition structure generation problems is to choose a proper genetic encoding that enables an efficient computational implementation. In this paper, we present a novel hybrid encoding, and we compare its performance against several genetic encoding proposed in the literature. We show that even in difficult instances of the coalition structure generation problem, the proposed approach is a competitive alternative to obtaining good quality solutions in reasonable computing times. Furthermore, we also show that the encoding relevance increases as the number of players increases.

1. Introduction

Let $N = \{a_1, \ldots, a_n\}$ be a finite set, P a nonempty subset of N, and $\mathscr{P}_n$ the power set of N. A partition $\Pi \subset \mathscr{P}_n$ is a set of pairwise disjoint subsets of N whose union is all of N [1]. Partitions arise as solutions for problems in many fields, including Game Theory [2], Multiagent Systems [3], Logistics [4], and Mathematical Programming [5].

In this paper, we focus on selecting the best partition from a discrete set. This problem relates to the complete set partitioning problem [6] in the Operations Research and Management Science (OR/MS) literature and is commonly formulated as a coalition structure generation problem [2] in cooperative game theory. Although there is a vast body of game theory literature dealing with coalition structure generation problems issues, the OR/MS literature is more sparse in this stream [4, 7, 8].

For the sake of exposition, we state our problem using concepts of cooperative game theory. In this context, N represents a set of players, P is called a coalition, and Π refers to a coalition structure. We denote $\mathscr{H}_n$ as the set of all the partitions of N. Let $\Pi \in \mathscr{H}_n$ be a given partition of N. Each partition Π represents a possible coalitional structure.

Our motivation in coalition structure generation problems comes from the field of horizontal collaboration in logistics, where stakeholders located at the same level of the supply chain cooperate to diminish logistics costs [4, 7, 9, 10]. Consequently, we formulate the problem as cost minimization instead of profit maximization.

Thus, let v: $\mathscr{H}_n \longrightarrow \mathbb{R}_+$ be a function that computes the cost of each coalition structure Π, which is usually defined as the sum of the costs of each coalition that belongs to the partition. Specifically, let c: $\mathscr{P}_n \times \mathscr{H}_n \longrightarrow \mathbb{R}_+$ be a function that computes the cost of each coalition P that belongs to the coalition structure Π, which is called *characteristic function*. The coalition structure generation problem, which is the problem we focus on, is given by

$$\min_{\Pi \in \mathscr{H}_n} v(\Pi) = \min_{\Pi \in \mathscr{H}_n} \sum_{P \in \Pi} c(P, \Pi). \tag{1}$$

The number of partitions of a set $|\mathcal{H}_n|$ is given by the Bell numbers [11, 12] that can be computed using the recurrence formula (2), with $B_0 = B_1 = 1$:

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_n. \quad (2)$$

Note that Bell numbers increase quite fast with respect to the number of players n (e.g., $B_{15} = 1,382,958,545$) which makes the coalition structure generation problem a difficult one to solve. In fact, this problem is NP-complete [13]. Although exact algorithms have been successfully implemented for tackling this problem [14–16], methods based in a specialized enumeration of the search space are impractical even for small-sized instances. For tackling these problems, the general practice is to develop an appropriate heuristic solution method rather than to rely on optimal solution approaches [17].

Genetic algorithms (GAs) [18] have proved to be a useful improvement procedure for several combinatorial problems, including the coalition structure generation problem [19, 20]. These algorithms consist of four main aspects: an encoding, a fitness function, a replacement policy, and genetic operators. An important issue when designing a Genetic algorithm is to choose a proper genetic encoding that enables an efficient computational implementation. The choice of a given encoding is a crucial step in applying GAs [21], which also affects the genetic operators that act on each encoding [22].

In this paper, we review several genetic encoding proposed in the literature. The objective is to identify which of these representations provides the best performance when exploring the search space of coalition structure generation problems. The main contribution of this paper is the development of a new hybrid encoding for tackling this combinatorial problem. We show that the proposed encoding outperforms other encodings when tested on random instances gathered from common distributions.

The rest of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 reviews several encoding strategies and presents a new hybrid encoding for coalition structure generation problems. Section 4 shows numerical experiments and results. Finally, concluding remarks are provided in Section 5.

2. Related Work

Coalition structure generation problems can be solved for two different games: characteristic function games (CFGs) and partition function games (PFGs) [3]. In CFGs, the cost of each coalition is given by the *characteristic function* of the game [23]. In these problems, the objective is to minimize the sum of the cost of each formed coalition [13]. In the OR/MS literature, characteristic function games arise in the form of set partitioning problems (SPP) and complete set partitioning problems (CSPP). Real-world applications of SPP and CSPP include, respectively, the air-crew scheduling problem [24, 25] and the optimal aggregation of corporate subsidiaries in corporate tax structuring [26]. In PFGs, on the contrary, the cost of a coalition may differ depending on how nonmembers are partitioned [3]. In these games, the cost of each partition is given by the *partition function*, which lacks a particular structure [3, 13]. Applications of PFGs include, among others, oligopolies [27], communication networks [28], and collaborative logistics [29].

Several exact algorithms have been proposed for tackling coalition structure generation problems for CFGs. Particularly, Yeh [14] proposes the so-called DP algorithm, which is based on the dynamic programming approach of Bellman [30]. The DP algorithm evaluates, for every coalition, whether it is beneficial to split it, and if so, what the best such split is. Björklund et al. [31] propose a dynamic programming algorithm based on the principle of inclusion-exclusion of combinatorics and the zeta transform [12]. Michalak et al. [16] develop the so-called ODP, an optimal version of the algorithm developed by Yeh [14], which avoids its redundant operations. On the contrary, Sandholm et al. [13] propose an anytime algorithm that focuses on establishing a worst-case bound on the quality of the coalition structure while only searching for a small fraction of the coalition structures. Rahwan et al. [15] propose a tree-search algorithm called IP. This anytime algorithm partitions the solution space into subspaces and prunes those that have no potential of containing the optimal solution and searches through the remaining subspaces using a branch-and-bound technique. More recently, Michalak et al. [16] develop the so-called ODP-IP, a hybrid algorithm that combines techniques from the dynamic programming approach and the anytime approach. According to the authors, this is the fastest exact algorithm for coalition structure generation problems in CFGs in practice. To develop exact algorithms for coalition structure generation problems in PFGs, i.e., games with externalities, some authors have considered additional assumptions such as nonpositive or nonnegative externalities to avoid examining every single coalition structure [3]. Rahwan et al. [32] develop the first anytime algorithm for coalition structure generation problems in PFGs with either positive or negative externalities. In their work, the authors identify the minimum search that is required to establish a bound on the quality of the best coalition structure found. Besides, they develop an anytime algorithm that improves this bound with a further search. Banerjee and Kraemer [33] extend the Rahwan et al. [32] algorithm considering that entities are grouped into types, which are used to define the externality imposed upon a coalition by other coalitions merging. Finally, Rahwan et al. [34] present an extended version of the Rahwan et al. [32] algorithm.

More recently, the research has focused on developing new methods for concisely representing CFGs. These compact representations capture the interaction between agents in order to reduce the representation size [35]. Examples of compact representations include Marginal Contribution Nets (MC-net) [36], Logic-based representation [37], and Partition Decision Trees (PDT) [38], among others. Ideally, a compact representation should be able to represent any coalitional game using small data structures that could be used by efficient algorithms. Examples of algorithms that work with MC-nets and develop mixed-integer linear

programming formulations can be found in Ueda et al. [35] and Liao et al. [39], where an improved version of Weighted Partial MaxSAT (WPM) encoding alongside an off-the-shelf WPM solver were developed. A similar WPM-based technique was proposed in Zha et al. [40] but in the context of PDT compact representation. The basic idea behind compact representations is to define *rules* which are used to compute the value of each coalition. Keeping in memory a reduced number of rules could be much more efficient than keeping all the 2^n values, i.e, one for each possible coalition. Even though many applications fit this framework, in general, either the problem cannot be described using a reduced number of rules or to find such a representation can be a challenging problem itself [36]. In this work, we focus on the general case where the characteristic function is considered as a black box.

Coalition structure generation problems are NP-complete [13]. Therefore, exact algorithms are useful only for solving small-sized instances. For tackling large-sized instances, nonexact algorithms, or heuristics, become a suitable alternative to obtain *good* quality solutions at reasonable computing times. In a nutshell, these algorithms return solutions relatively quickly and scale up well when the numbers of players increase [34]. However, they provide no guarantee of the quality of their solutions. Several nonexact algorithms have been proposed for coalition structure generation problems. Shehory and Kraus [41] propose a greedy-distributed algorithm with low ratio bounds. Sen and Dutta [42] develop an order-based genetic algorithm for exploring the search space. Keinänen [43] proposes a simulated-annealing approach for tackling coalition structure generation problems in CFGs. Di Mauro et al. [44] propose a GRASP approach, which is then compared with exact methods considering randomly generated characteristic functions. From a different perspective, Dos Santos and Bazzan [45] propose the so-called *bee clustering* algorithm, a distributed clustering algorithm inspired by swarm intelligence techniques. Later, Farinelli et al. [46] propose a hierarchical agglomerative clustering approach.

In this paper, we focus on a particular type of nonexact algorithm, the Genetic algorithm (GA), which can be used for tackling coalition structure generation problems in both CFGs and PFGs [42]. Developed by Holland [18], GAs are stochastic search algorithms that emulate biological evolution based on Charles Darwin's theory of natural selection [47]. For surveys on GAs, see Whitley and Sutton [48] and Srinivas and Patnaik [49]. Moreover, a comprehensive overview can be found in Beasley et al. [50, 51]. GAs are also included in the Evolutionary Computation field [52]. A review of bioinspired computing algorithms can be found in Kar [53]. GAs have been effective in tackling several NP-complete combinatorial optimization problems, including the coalition structure generation problem. Particularly, Levine [19] proposes a parallel genetic algorithm with a penalization function for solving the set partitioning problem. Chu and Beasley [20] present a steady-state GA in conjunction with a specialized heuristic improvement operator for solving the same problem. Other GAs applications include Venugopal and Narendran [54] for machine-component grouping, Gonçalves and Resende [55] for manufacturing cell formation, and Boulif [56] for graph partitioning.

As stated in Section 1, one of the main decisions when designing a GA is to choose a genetic encoding. The selection of a proper encoding is a crucial step when implementing GAs [21], which also affects the genetic operators that act on each encoding [22]. Particularly, the encoding of solutions should implicitly give a semantic description of what a good solution is [57]. Some encodings have been proposed for tackling coalition structure generation problems considering a GA framework. For example, Sen and Dutta [42] propose an *order-based* encoding; Chu and Beasley [58] and Levine [19] propose a *column-based* encoding; and Gonçalves and Resende [55] propose a fractional encoding. As shown in Section 3, however, several other encodings can be applied for tackling this problem.

3. GAs for Coalition Structure Generation Problems

In this section, we detail GAs for the specific setting of coalition structure generation problems. Particularly, in Section 3.1, we provide an overview of GAs; in Section 3.2, we review several encoding schemes for coalition structure generation problems; in Section 3.3, we discuss the new encoding scheme; and in Section 3.4, we summarize the features of the encodings analyzed.

3.1. Overview of GAs. GAs are stochastic search algorithms that mimic the genetic evolution of species. In a GA framework, the main components are an encoding scheme, a fitness function, a replacement policy, and genetic operators. GAs start with an initial population that evolves, yielding a new population of the same size. The main operators used to create a new generation are selection, recombination, and mutation. A peculiarity of this approach is that genetic operators do not work directly on the solution space or phenotype; solutions have to be coded as strings over a particular alphabet. In this case, an encoding scheme is used to represent the solutions as chromosomes or individuals that belong to the genotype space. Algorithm 1 shows a pseudocode for a basic GA version [20]. We refer the reader to Beasley et al. [50, 51]; Pirlot [57]; and Boussaïd et al. [59] for comprehensive overviews (Algorithm 1).

The performance of GAs depends on the relationship between phenotype and genotype. There are three types of relations: one-to-one, one-to-many, and one-to-none. The one-to-one relation seems to be the ideal case. Unfortunately, in many instances, it is quite hard to find a simple encoding with this property. The one-to-many relation occurs when it is possible to represent a solution through many different chromosomes. In this case, the GA is said to face *redundancies*. Redundancies detriment the performance of GAs since the genotype space expands, and it is needed to explore a larger space. Finally, in the one-to-none relation, some solutions cannot be represented as chromosomes. In this case, the GA is said to face *blindness* since promising

(1) Generate an initial population;
(2) Evaluate fitness of individuals in population;
(3) **repeat**
(4) Select parents from the population;
(5) Apply a crossover operation to get new individuals;
(6) Mutate a random proportion of these new individuals;
(7) Evaluate the current population by its fitness;
(8) Create a new population using a replacement strategy;
(9) **until** A termination condition is satisfied
(10) Return the current population.

Algorithm 1: A basic GA.

sectors of the search space may not be analyzed. For further insight into both definitions, we refer the readers to Boulif [56].

3.2. Encoding Schemes for Coalition Structure Generation Problems. We review several encoding schemes and discuss some genetics operators and their variants. For the sake of clarity, we focus solely on encoding schemes that do not face blindness.

3.2.1. Column-Based Encoding. Column-based encoding was first proposed for tackling set partitioning, set covering, and graph partitioning problems [19, 58, 60, 61]. In this scheme, the genes are 0–1 bits. Bits correspond to coalitions, and the values 1 or 0 indicate whether the coalition is in the coalition structure or not. An example of this encoding/decoding is depicted in Figure 1 Gen 1.

Column-based encoding does not face redundancies. However, the size of the genotype space is much larger than the size of the phenotype space (i.e., $2^m \gg |\mathscr{H}_n|$, where $m = 2^n - 1$ is the number of coalitions). The excess of individuals in the genotype is due to the existence of chromosomes that cannot be decoded into valid partitions (see Figure 1, Gen 2). This phenomenon is a major drawback of column-based encoding since many iterations are required to get feasible individuals. To deal with this issue, some authors propose the use of penalty functions [19, 60] and feasibility repair operators [20, 62].

3.2.2. Row-Based Encoding. A row-based encoding works by creating empty *clusters*. Then, the clusters are filled up with players, and the formed coalitions correspond to the nonempty clusters. Since coalition structures are formed by at most n coalitions, exactly n clusters are enough to represent any possible solution. We next review two standard row-based encoding: integer and fractional.

Integer row-based encoding was proposed by Venugopal and Narendran [54] in the context of Machine-Component Grouping. In this scheme, solutions are represented using an array of n components with values in $\{1, \ldots, n\}$. Each component represents a player, and its value indicates the cluster in which the player is included. An example of this encoding/decoding is depicted in Figure 2.

Fractional row-based encoding was proposed by Gonçalves and Resende [55] in the context of Manufacturing Cell Formation. In this setting, each solution is represented by an array of $(n+1)$ components with values in the interval $[0, 1]$. The decoding process is as follows. The last component of the array is multiplied by the number of players n, which yields the number of clusters. Let n_c be this number. Now, the interval $[0, 1]$ is divided into n_c subintervals of equal length: $[0, 1/n_c]$, $(1/n_c, 2/n_c]$, $\ldots$, $((n_c - 1)/n_c, 1]$. Each subinterval represents one of the n_c clusters. Finally, the players—represented by the first n components of the array—are assigned to the clusters according to the component value and the subinterval to which it belongs. An example of this encoding/decoding is depicted in Figure 3.

The integer and fractional row-based encoding share some common features. First, they are phenotype feasible, which means that chromosomes always generate valid partitions. Moreover, conventional crossover and mutation operators can be directly applied. However, these encoding face redundancies due to the interchangeable role of clusters. For example, consider the coalition structure $\{1, 3, 4\}, \{2, 5\}$. In row-based integer encoding, the partition above is represented not only by the chromosome $[1\,|\,2\,|\,1\,|\,1\,|\,2]$ but also by $[2\,|\,1\,|\,2\,|\,2\,|\,1]$, $[1\,|\,3\,|\,1\,|\,1\,|\,3]$, $[1\,|\,4\,|\,1\,|\,1\,|\,4]$, and so on. In general, the number of redundant chromosomes for a coalition structure is given by equation (3), where n represents the clusters and $|\Pi|$ the formed coalitions:

$$R_{\text{Int–row}} = \left(\begin{matrix} n \\ |\Pi| \end{matrix} \right). \tag{3}$$

On the contrary, in the row-based fractional encoding, the amount of redundancies is larger since we deal with both cluster interchangeability and machine precision. For example, consider the chromosome of Figure 3. In this case, the value of the first allele could be replaced for any number in interval $[0, 0.5)$ without changing the resulting coalition structure. Let M be the number of values that a machine can distinguish in the interval $[0, 1]$. Then, the number of redundant chromosomes that maps the coalition structure Π is given by equation (4), where $(M/n)^{n+1}$ denotes the number of ways that every component can be replaced by a value in the same subinterval:

$$R_{\text{Frac–row}} = \left(\frac{M}{n}\right)^{n+1} R_{\text{Int–row}}. \tag{4}$$

3.2.3. Order-Based Encoding. Order-based encoding was proposed by Sen and Dutta [42] for tackling coalition structure generation problems. In this scheme, coalition structures are coded using permutations on the set $\{1, \ldots, 2n-1\}$. In the chromosome, $\{1, \ldots, n\}$ represent the players, whereas the numbers $\{n+1, \ldots, 2n-1\}$ stand for *separators* or *coalition breakers*, which determine where a coalition ends and the next one begins. An example of this encoding/decoding is depicted in Figure 4.

The idea behind order-based encoding has been widely applied in the Traveling Salesman Problem (TSP) setting [63]. In this scheme, the conventional operators, such as

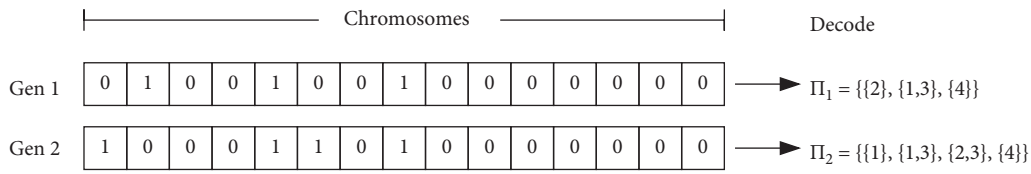

Figure 1: Column-based encoding/decoding for 4 players.

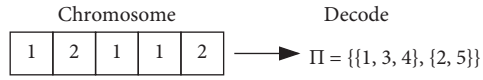

Figure 2: Integer row-based encoding/decoding for 5 players.

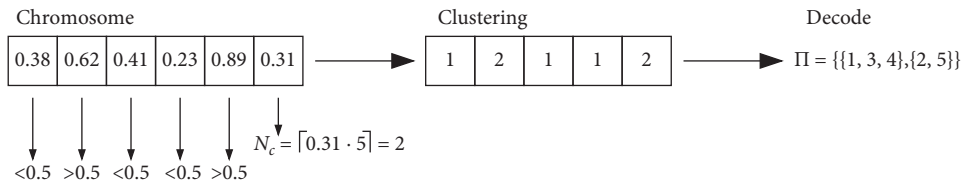

Figure 3: Fractional row-based encoding/decoding for 5 players.

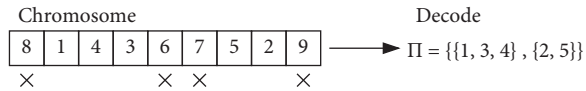

Figure 4: Order-based encoding/decoding for 5 players. The cross at the bottom denotes the coalition breakers.

multipoint crossover or backflip mutation, fail to create new valid chromosomes since the children are typically not permutations. Specific operators have been developed to deal with this issue. Examples are the PMX-crossover (PMX) [64], the Edge Recombination (ER), and the Swap-mutation [65].

Sen and Dutta [42] computed a lower bound for the number of redundancies faced by order-based encoding. To compare with the other encoding, we improve the analysis by determining the exact number of redundancies. Let Π be a coalition structure formed by $|\Pi|$ coalitions. Proposition 1 provides the number of redundancies for the order-based encoding.

Proposition 1. *Let Π be a coalition structure formed by $|\Pi|$ coalitions. Then, the number of chromosomes representing Π in the order-based encoding is given by*

$$R_{\mathrm{OB}} = (n-1)! \prod_{P \in \Pi} |P|! |\Pi|! \binom{n}{|\Pi|}. \tag{5}$$

Proof. There are four sources of redundancies in the order-based encoding. First, the role of separators is interchangeable, so we have to include the factor $(n-1)!$. Second, players within the same coalition can be reordered without affecting the formed coalition, then the factor $|P|!$ appears for each formed coalition P. Similarly, the position of coalitions in the chromosome is also interchangeable. Then we have to include the factor $|\Pi|!$. Finally, note that to represent Π, we need $|\Pi| - 1$ separators. As the encoding always considers $(n-1)$ separators, the extra $n - |\Pi|$ can be placed either next to an already fixed separator or in the extremes of the chromosome. This yields a combinatorial number that is equivalent to the number of vectors $(x_1, \ldots, x_{|\Pi|+1})$ with nonnegative integers that satisfies $x_1 + \cdots + x_{|\Pi|+1} = n - |\Pi|$. This problem is well-known in combinatorics in which the general solution is $\binom{n' + k' - 1}{k' - 1}$, where $k\prime$ the size of the vector and $n\prime$ is the amount we want to sum. In our case $\binom{(n - |\Pi|) + (|\Pi| + 1) - 1}{(|\Pi| + 1) - 1} = \binom{n}{|\Pi|}$, which is the last factor in the expression. □

3.2.4. Random-Key Encoding. Random-key encoding was proposed by Bean [66] for tackling several sequencing problems. In this scheme, a solution is encoded with random numbers drawn from $[0, 1]$. The decoding process, on the contrary, is carried out by sorting the alleles in ascending order, which results in a permutation on the set $\{1, \ldots, n\}$. Then, note that, in the context of coalition structure generation problems, a random-key encoding with $2n - 1$ alleles decodes into a permutation that can be interpreted as a chromosome of the order-based encoding. An example of this encoding/decoding is depicted in Figure 5.

An advantage of using a random-key encoding is that it enables the use of simple genetic operators since the chromosome is represented quite clearly. However, redundancy increases considerably due to the use of real-valued genes.

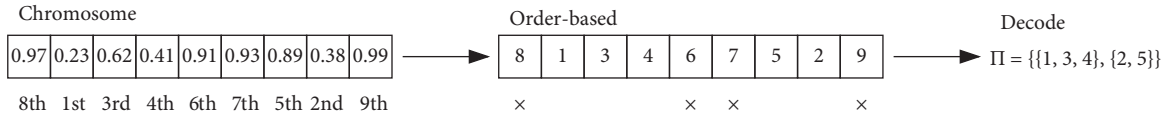

Figure 5: Random-key encoding/decoding for 5 players.

3.3. A New Codification Scheme: The Order-Based Bit-Key Encoding. In this paper, we propose a new encoding scheme for tackling coalition structure generation problems: order-based bit-key encoding (OBBK). The proposed encoding is based on both order-based and random-key encoding. In this scheme, partitions are encoded using an order-based chromosome of length n and a bit-string key with $n-1$ components. The order-based segment provides an order for the players, while the values in the key indicate the coalition breaks. Therefore, if player x_i is next to player x_{i+1} in the order-based chromosome, the value of the ith component of the key indicates whether the players belong to the same coalition or not. An example of this encoding/decoding is depicted in Figure 6.

Despite the similarities with order-based encoding, OBBK encoding avoids several sources of redundancies. Proposition 2 provides the number of redundancies for this encoding.

Proposition 2. *Let Π be a coalition structure formed by $|\Pi|$ coalitions. Then, the number of chromosomes representing Π in the OBBK encoding is given by*

$$R_{\text{OBBK}} = \prod_{P\in\Pi} |P|!)|\Pi|!. \tag{6}$$

Proof. Following the same proof as in Proposition 1, we note that OBBK encoding faces neither the first nor the fourth sources of redundancies, namely, interchangeability and excess of separators. On the contrary, OBBK still suffers from players and coalition interchangeability; thus, the number of redundancies is given by $(\prod_{P\in\Pi}|P|!)|\Pi|!$.

We also develop specific genetic operators for the OBBK encoding. These operators involve the use of conventional operators for order-based and bit-string encoding, which are applied separately to each segment of the array. For instance, in the crossover step, it is possible to recombine the order-based section using the PMX operator, while the key section is recombined through the single-point operator. An advantage of this strategy is that the use of standard operators, such as PMX or ER, is quite simple because the order-based segment is smaller, particularly, n components for OBBK against $2n-1$ in the OB encoding. Figure 7 depicts the use of genetic operators when using OBBK encoding. □

3.4. A Summary of the Encoding Features. Table 1 summarizes the encoding reviewed and their main features. Column *G-space* indicates the size of the genotype space, which is the search space for GAs. A straightforward comparison reveals that OBBK encoding provides the smallest search space, hence the smallest number of redundancies.

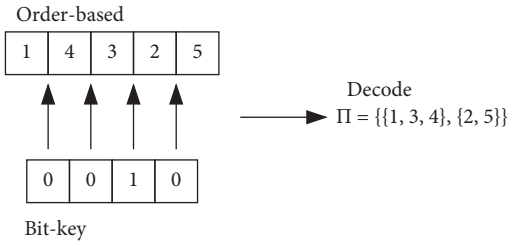

Figure 6: OBBK encoding/decoding for 5 players.

4. Numerical Experiments

In this section, we detail the experiments and numerical results. Particularly, in Section 4.1, we show how to generate instances for the coalition structure generation problems; in Section 4.2, we discuss the implementation of genetic algorithms in this context; and in Section 4.3, we analyze the insights gathered from the numerical experiments.

4.1. Generating Instances for Coalition Structure Generation Problems. Instances for coalition structure generation problems can be divided into four groups: structured CFGs, structured PFGs, irregular CFGs, and irregular PFGs. We define as structured instances those in which the objective function defines a simple structure on the partition space. For this case, due to the regularity induced, the optimal solution can be easily computed a priori. Several real-life applications, however, involve problems with less or even no structure at all. We define as irregular instances those instances constructed with a specific objective function in which the cost of individual coalitions is sampled from random values. In this case, the optimal solution is unknown a priori.

In either case, structured or not, the GAs developed in this section have no preliminary notions about the data nor do they make any assumption about the objective function.

4.1.1. Structured CFGs and PFGs Instances. Sen and Dutta [42] propose a novel framework for generating CFGs and PFGs instances with some regularity in the partition space. In this paper, we use their approach to generate instances for both structured CFGs and PFGs.

For CFGs, on the one hand, structured instances could be obtained by using a coalition cost that depends on both the size of the coalition and an intracoalition distance function. Consider a set $\{1,\ldots,n\}$ of players. Then, given a coalition $P = \{i_1,\ldots,i_{|P|}\}$, the cost of P, $c(P)$, is computed according to

$$c(P) = -\phi(|P|) + \min_{j\in P\setminus\{i\}} |i-j|. \tag{7}$$

In equation (7), the function ϕ: $\mathcal{P}_n \longrightarrow \mathbb{R}_+$ works as a penalty function that establishes preferences for the size of

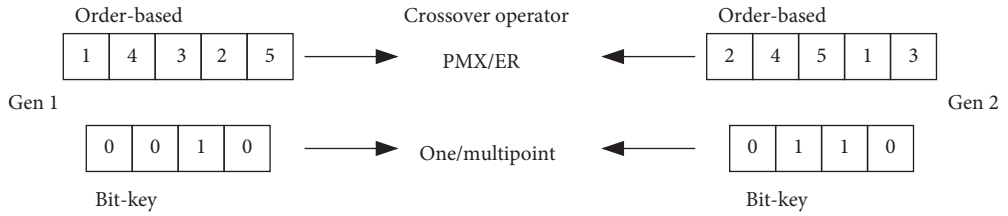

Figure 7: Genetic operators for Order-based with Bit-key encoding.

Table 1: Encoding and their features.

Encoding	Genes	Feasibility	G-space	Redundancies of coalition P
Column-based	$2^n - 1$ (bit)	✖	2^{2^n-1}	0
Int-row-based	n (int)	✓	n^n	$\binom{n}{\lvert\Pi\rvert}$
Frac-row-based	$n+1$ (float)	✓	M^{n+1}	$(M/n)^{n+1}\binom{n}{\lvert\Pi\rvert}$
Order-based	$2n-1$ (int)	✓	$(2n-1)!$	$(n-1)!\binom{n}{\lvert\Pi\rvert}\left(\prod_{P\in\Pi}\lvert P\rvert!\right)\lvert\Pi\rvert!$
Random-key	$2n-1$ (float)	✓	M^{2n-1}	$\gg (n-1)!\binom{n}{\lvert\Pi\rvert}\left(\prod_{P\in\Pi}\lvert P\rvert!\right)\lvert\Pi\rvert!$
OBBK	n (int) $+ n - 1$ (bit)	✓	$n!2^{n-1}$	$\left(\prod_{P\in\Pi}\lvert P\rvert!\right)\lvert\Pi\rvert!$

the formed coalitions. Particularly, if we seek coalitions with K players in the optimal coalition structure, then we can accomplish this by using.

$$\phi(x) = \begin{cases} x^2, & \text{if } 0 < x < K, \\ (2K - x)^2, & \text{if } K \le x \le 2K, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

For generating structured PFG instances, on the other hand, the intracoalition distance function of equation (7) is replaced by an intercoalition distance function. The new distance function penalizes coalition structures in which the formed coalitions are *far* from each other. For this case, given a coalition structure Π and a coalition $P \in \Pi$, we compute the cost of P by

$$c(P, \Pi) = \phi(|P|) + \max_{Q \in \Pi \setminus \{P\}} \left| \sum_{i \in P} i - \sum_{j \in Q} j \right|. \tag{9}$$

Note that the use of an intracoalition distance makes the cost of a coalition independent from the other formed coalitions. On the contrary, the intercoalition distance depends explicitly on the other coalitions within the coalition structure, so the problem becomes a PFG. In both cases, i.e., structured CFGs and PFGs instances, the optimal solution is given by (10), assuming a function ϕ given by (8):

$$\Pi^* = \left\{ \{1, \ldots, K\}, \{K+1, \ldots, 2K\}, \ldots, \left\{ \left\lfloor \frac{n}{K} \right\rfloor + 1, \ldots, n \right\} \right\}. \tag{10}$$

4.1.2. Irregular CFGs and PFGs Instances. To generate irregular CFGs instances, we develop a methodology similar to the NDCS (Normally Distributed Coalition Structures) developed by Rahwan et al. [15]. In their work, the cost of each coalition P is sampled from an independent normal distribution with parameters $\mu_P = |P|$ and $\sigma_P = \sqrt{|P|}$. With this choice, the value of coalition structure is proved to satisfy $v(\Pi) \sim N(n, n)$ (Theorem 6 in Rahwan et al. [15])0. In this paper, we generalize their approach so that the value of coalition structures satisfy $v(\Pi) \sim N(\mu, \sigma^2)$ for any parameters values μ and σ defined by the user. Consider Proposition 3.

Proposition 3. *Let Π be a coalition structure and let P be a coalition of size $|P|$. If the cost of coalition P, $c(P)$, is sampled from a normal distribution $N(\mu_P, \sigma_P^2)$, with mean and variance given by $\mu_P = (|P|/n)\mu$ and $\sigma_P^2 = (|P|/n)\sigma^2$, then the cost of partition Π, $v(\Pi)$, follows a normal distribution $N(\mu, \sigma^2)$.*

Proof. Let Π be a coalition structure. Then, the cost of this partition is given by

$$\begin{aligned} v(\Pi) = \sum_{P \in \Pi} c(P) &\sim \sum_{P \in \Pi} N\left(\mu_P, \sigma_P^2\right) \\ &\sim N\left(\sum_{P \in \Pi} \mu_P, \sum_{P \in \Pi} \sigma_P^2 \right) \\ &\sim N\left(\frac{\mu}{n} \sum_{P \in \Pi} |P|, \frac{\sigma^2}{n} \sum_{P \in \Pi} |P| \right). \end{aligned} \tag{11}$$

Now, for any coalition structure Π, it holds that $\sum_{P \in \Pi} |P| = n$. Therefore, $v(\Pi) \sim N(\mu, \sigma^2)$, and all coalition structures are equally likely to be optimal.

For generating irregular PFGs instances, on the contrary, we assign a partial cost $\widetilde{c}(P)$ to each coalition P. We sample the partial costs independently from the same distribution. Then, for each coalition $P \in \Pi$, the cost of P in Π is given by

$$c(P, \Pi) = \frac{1}{|\Pi|}\widetilde{c}(P). \tag{12}$$

Finally, the cost of a coalition structure is computed by adding up the costs of all the coalitions. This is given by

$$v(\Pi) = \sum_{P \in \Pi} c(P, \Pi) = \frac{1}{|\Pi|} \sum_{P \in \Pi} \widetilde{c}(P). \tag{13}$$

Since the cost depends on the number of formed coalitions, this kind of problem belongs to the PFG class. Moreover, if we assume (12) and (13), the problem corresponds to a PFG with negative externalities. To prove this, let Π' denote a coalition structure resulting from merging two coalitions in Π. Then, $|\Pi'| < |\Pi|$, and hence $c(P, \Pi) - c(P, \Pi') < 0$ for all $P \in \Pi' \cap \Pi$ which is the negative externalities property (see Section 2).

In this paper, we sample the partial costs $\widetilde{c}(P)$ independently from three distributions, namely, a uniform distribution with values in $[0, 100]$; an exponential distribution with parameter $\lambda = 1/50$; and a normal distribution with parameters $\mu = 50$ and $\sigma^2 = 100$. □

4.2. Implementation Details. In this section, we discuss further the implementation of GAs for the coalition structure generation problem. Consider Algorithm 2.

Step 1 creates the initial population by randomly selecting chromosomes in the genotype space. Step 4 copies the best individuals into the next population to preserve the good chromosomes. Step 6 performs a fitness-based random selection. Note that the best fitness-valued individuals are more likely to participate in the recombination procedure. For the rest of the individuals, Step 11 performs a randomly uniform selection providing a chance for bad fitness-valued individuals to appear in the next generation avoiding premature homogenization. Step 5 uses the crossover rate parameter CR, which establishes the number of individuals that should be obtained as mutated children in the next generation. Step 8 considers the mutation probability parameter MP, which usually is taken as a small value. Note that the size of the population remains constant and equal to N. Finally, Step 14 states the stopping criterion.

Table 2 summarizes the value of the parameters used in for all the algorithms. Note that the population size depends on the number of players n. We proceed in that way, considering that, as the numbers of players increases, the feasible set size grows according to Bell numbers (see Section 1).

4.3. Numerical Results. In this section, we perform several numerical experiments for the problems defined in the previous ections, that is, for the structured CFG and PFG problems and irregular CFG and PFG problems. As stated before, the optimal solution for the structured problems is known in advance and given by equation (10). For the irregular CFG, on the contrary, although the solutions are not previously known, the optimal solutions can be computed using the ODP-IP algorithm as long as the number of players does not exceed 25. In the case of irregular PFG problems, we define three types of instances associated with three different probability distributions: uniform, exponential, and normal. In summary, we have six different types of problems, and for each of them, we generate 23 instances. Considering that each instance has a different number of players varying between 8 and 30 players, we solved, therefore, 138 instances.

Moreover, we compare the performance of six GAs, one for each reviewed encoding scheme. Note that each encoding may work with several crossover and mutation operators, which yield to a combinatorial number of cases. For the sake of simplicity, we limited the computational experiments to testing each encoding with its classic operators. Table 3 summarizes the encoding and the operators used.

In order to mimic a multistart technique, the genetic algorithms are executed ten times, starting from different initial populations at each run. Note that the generation of the initial population (Step 1 of Algorithm 2) depends on the encoding scheme. Thus, all algorithms start from different initial populations.

We compute the relative gaps in order to measure the performance of the algorithms [16]. For the instances whose optimal solutions are known, the relative optimality gaps are computed using equation (14). In this equation, v denotes the best value reported by the algorithm, whereas v^* denotes the known optimal solution. On the contary, for those instances where the optimal solution v^* are not known, i.e., for irregular PFGs, we use the same expression (equation (14)), but replacing v^* by the best lower bound:

$$\text{GAP} = \frac{v - v^*}{v^*} 100\%. \tag{14}$$

First, we discuss the results for structured instances, which are depicted in Figure 8. The x−axis shows the number of players. The y−axis shows the average GAP attained by each algorithm in each instance. At the top of the bar, we point out the algorithm that achieves the lowest average GAP and the corresponding value. Since the instances are independent, it is expected that the average GAP differs from one instance to another. However, there is a general trend to have larger GAPs for instances with a greater number of players. The OBBK algorithm shows the best performance in all instances for coalition structure generation problems in both CFGs and PFGs. Even for a large number of players, this algorithm keeps the GAP below 5%. The OBBK algorithm, moreover, reaches a zero GAP in many of the instances, which means that the algorithm finds the optimal solution for the ten runs. Besides, in cases where the average GAP is not zero, the algorithm still finds the optimal solution in some of the runs. Specifically, this happens 40% of the tested instances for the CFG with 30 players, and 20% of the instances for the PFG with 27 players.

Now, we focus on the results for the irregular CFG instances. In these instances, the optimal solution is

(1) Create an initial population Q with N individuals.
(2) **repeat**
(3) Evaluate the current population Q.
(4) Set $Q^{next} \longleftarrow \{x_{(1)}, \ldots, x_{(N_{elite})}\}$, where $\{x_{(i)}\}_i$ are the individuals sorted by fitness.
(5) **for** $j = 1$ to $\lceil CR \cdot N \rceil$ **do**
(6) Select two parents from the current population Q.
(7) Generate one child by applying the crossover operator.
(8) Mutate the generated child with probability PM.
(9) Add the child into the next population Q^{next}.
(10) **while** the size of Q^{next} be less than N **do**
(11) Select an individual x in the current population Q.
(12) Add x into Q^{next}.
(13) Update $Q \longleftarrow Q^{\text{next}}$
(14) **until** The generations limit is reached
(15) Return the best individual in Q.

Algorithm 2

Table 2: Parameters for the GAs.

Population size	100 n
Generations	30
Crossover rate	0.8
Mutation probability	0.2
Elite count	5

Table 3: Encoding and genetic operators used.

Name	Encoding	Crossover	Mutation
Bit-col	Column-based	Multipoint	Backflip
Int-row	Integer row-based	Multipoint	Backflip
Frac-row	Fractional row-based	Multipoint	Backflip
OB	Order-based	PMX	Swap
Rand-key	Random-key	Multipoint	Backflip
OBBK	Order-based bit-key	PMX-multipoint	Swap-backflip

unknown a priori. To compute the optimality GAP, we use the ODP-IP method proposed by Michalak et al. [16] to search for the optimal solution. Note that this method and its implementation[1] are defined for maximization problems. However, our formulation is stated as a minimization problem. To cope with this issue, for M large enough, we use as input the reward function:

$$v(P, \Pi) = M|P| - c(P, \Pi), \quad \forall \Pi \in \mathscr{H}_n, \quad \forall P \in \Pi. \tag{15}$$

Therefore,

$$\begin{aligned} \arg\max_{\Pi \in \mathscr{H}_n} \sum_{P \in \Pi} v(P, \Pi) &= \arg\max_{\Pi \in \mathscr{H}_n} \left(Mn - \sum_{P \in \Pi} c(P, \Pi) \right) \\ &= \arg\min_{\Pi \in \mathscr{H}_n} \sum_{P \in \Pi} c(P, \Pi). \end{aligned} \tag{16}$$

The ODP-IP method works, however, for instances of up to 25 players. For instances with more than 25 players, instead of computing the relative error, we compute the average minimum value for each algorithm. These values are reported in Table 4.

Figure 9 shows that the average GAP increases when the number of players increases. In general terms, Bit-col and Int-row algorithms show poor performance. Again, for the same initial population size and the same number of generations, the OBBK algorithm has the best performance in all instances. The differences in the average GAPs are now much clearer, especially for larger instances, for which the other algorithms have a GAP up to five times larger than OBBK. Even though the OBBK algorithm shows the best performance, the GAP increases up to 24% for the largest instance, far from the 5% reached in the case of structured instances.

Figure 10 shows 95% confidence intervals for the GAP of the OBBK algorithm when solving irregular CFG problems. We considered only 25 players since, as stated before, it is the maximum number of players for which the ODP-IP algorithm computes the optimal solution [16]. Note that the length of the confidence intervals grows as the class of problems become more complex, that is, as the number of players increases.

We now focus on the results for the irregular PFG instances. Consider Figures 11–13. The interpretation of these results is not straightforward. For the three distributions considered, we obtain bar graphs with different characteristics. For the uniform case, the GAP roughly decreases as the number of players increases. For the exponential case, the behavior is similar to the structured instances, while, for the normal case, there is no correlation between the GAP and the number of players. An explanation for this phenomenon could be the procedure used to compute the GAP. As we mentioned before, for this type of instance, the optimum is unknown, so there are no tools for computing this value in reasonable computing time. When there is no guarantee for reaching the optimal solution, the error is computed using different lower bounds, which depends on each distribution. In the case of uniform and exponential distributions, the minimum value is 0, which is set as the lower bound. For the normal distribution, we use as lower bound the value 20 that corresponds to $\mu - 3\sigma$.

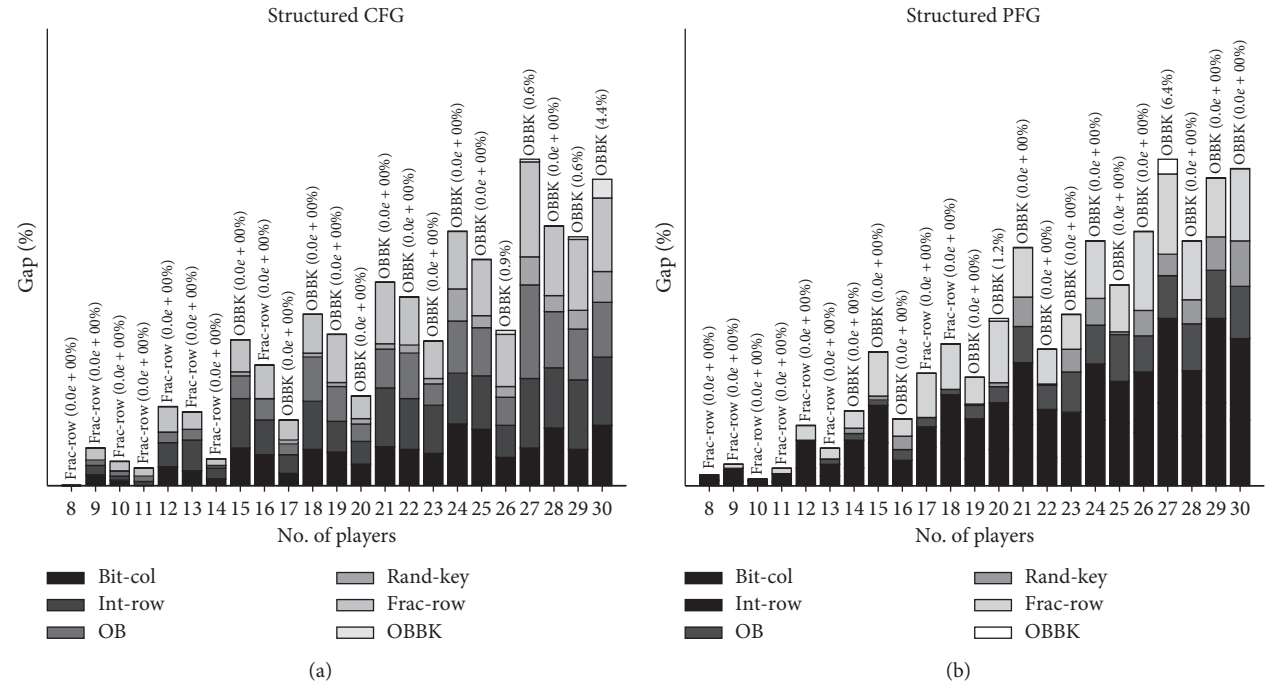

Figure 8: Average GAP for structured CFG and PFG instances.

Table 4: Average minimum values for instances of Normal CFG with more than 25 players.

Players	Bit-col	Int-row	Frac-row	OB	Rand-key	OBBK
26	154.3	132.4	87.31	125.6	115.6	**80.38**
27	148.6	131.5	101.9	118.2	128.1	**90.61**
28	152.1	145.7	103.6	113.1	139	**101.5**
29	175.0	167.1	132.2	162.7	164.9	**100.4**
30	168.9	177.7	144.6	168.0	170.4	**121.2**

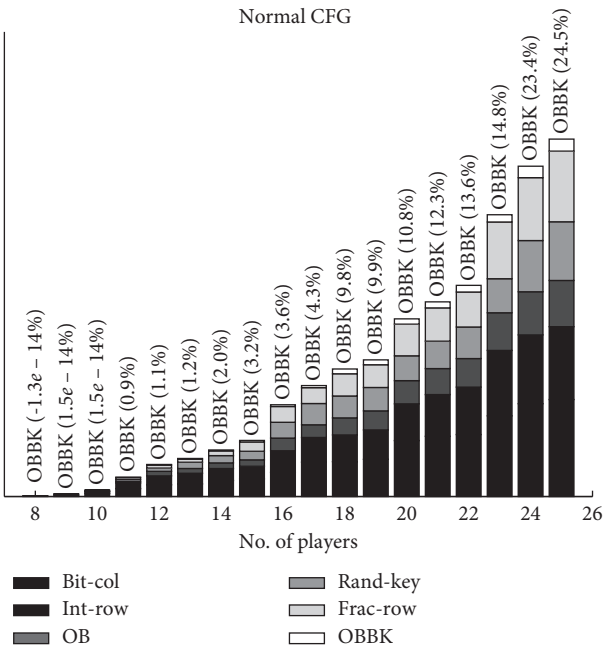

Figure 9: Average GAP for irregular CFG instances.

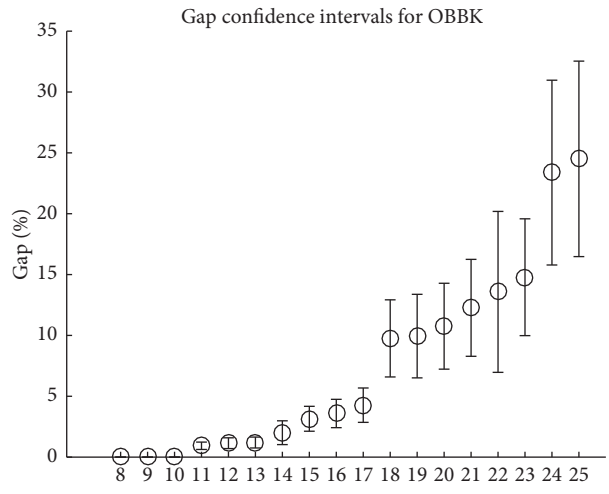

Figure 10: Confidence intervals for the OBBK algorithm.

In this case, the performance of the algorithms improves considerably compared to the Normal CFG instances. The optimality GAP values do not exceed 20% and remain below 5% even for the largest cases. Once again, the OBBK

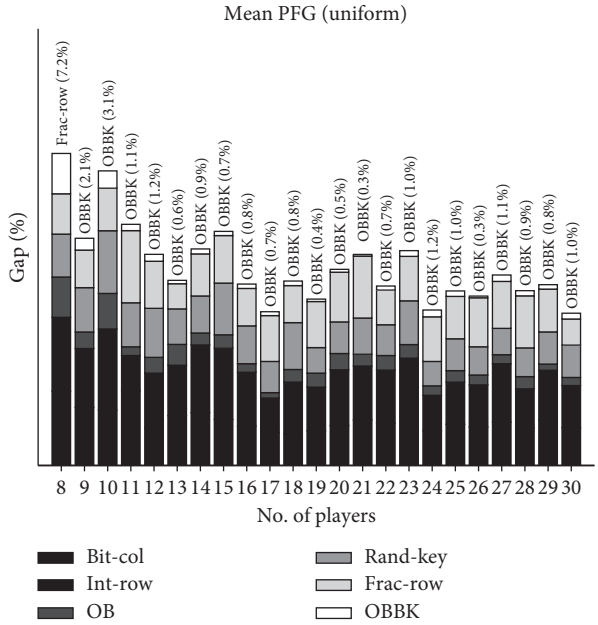

Figure 11: Average GAP for Mean PFG instances with uniform distribution.

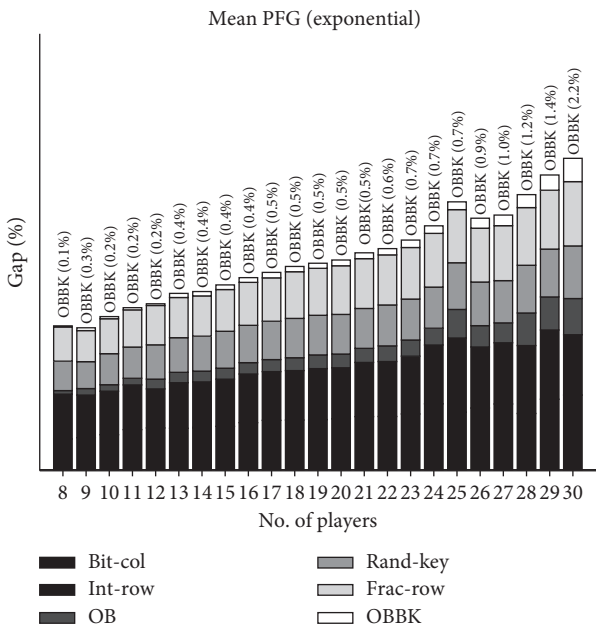

Figure 12: Average GAP for Mean PFG instances with exponential distribution.

algorithm reports the best performance in almost all instances. The Frac-row algorithm also has a good performance, reaching better optimality GAPs than OBBK in some instances with few players. This last algorithm was also competitive in the Normal CFG instances. Nevertheless, for the structured instances, the second-best algorithm is OB.

Finally, it is needed to point out that the OBBK algorithm achieve an effectiveness of 80% in 15 of the 18 studied instances, i.e., in 8 of 10 runs, this algorithm finds the optimal solution. Frac-row is the closest algorithm in terms of effectiveness, reaching the same 80% but only in 4 of the 18 instances studied.

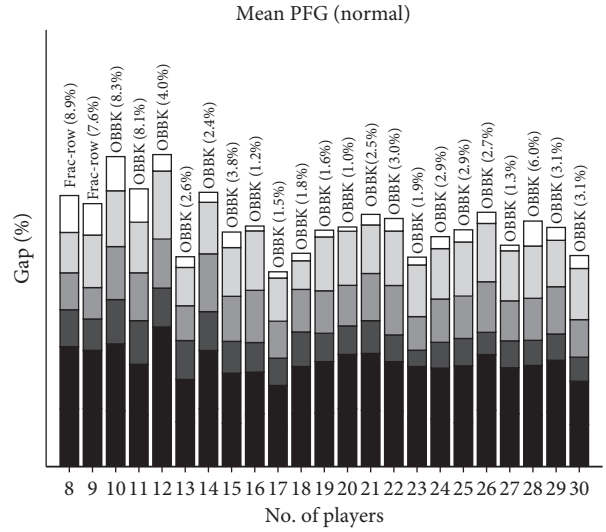

Figure 13: Average GAP for Mean PFG instances with normal distribution.

5. Concluding Remarks

In this paper, we review and compare several encoding used in the literature for tackling coalition structure generation problems using GAs. Moreover, we propose a novel hybrid encoding that outperforms the existing ones in all the studied instances. Even for difficult cases, such as the irregular CFG and PFG instances, the proposed approach allows us to obtain good solutions in reasonable computing times. We also show that the encoding relevance increases when the number of players increases. For the specific case of irregular CFG, and for small instances, our solution approach can find near-optimal solutions. For larger instances, namely, more than 25 players, which is the limit of the previous exact methodologies, our solution method shows a good performance for totally different instances, which proves the robustness of our procedure. This last is a distinctive feature of the OBBK encoding since, in general, the other encodings work well in specific structures and tend to provide worse results when these structures change. Finally, it is worth mentioning that GAs provide a solution approach for the unstructured PFG, which until now, is optimally unsolvable for large-sized instances.

Data Availability

The simulated data used to support the findings of this study are available from the corresponding author upon request

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors gratefully acknowledge the financial support from the Complex Engineering Systems Institute, ISCI (grant CONICYT PIA/BASAL AFB180003). Mauricio Varas thanks a grant from Science, Technology, Knowledge and Innovation Ministry of Chile (FONDECYT Project 11190892). Juan Pablo Contreras also thanks the National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2019-21190161 for their support.

References

[1] R. Paige and R. E. Tarjan, "Three partition refinement algorithms," *SIAM Journal on Computing*, vol. 16, no. 6, pp. 973–989, 1987.

[2] R. J. Aumann and J. H. Dreze, "Cooperative games with coalition structures," *International Journal of Game Theory*, vol. 3, no. 4, pp. 217–237, 1974.

[3] T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings, "Coalition structure generation: a survey," *Artificial Intelligence*, vol. 229, pp. 139–174, 2015.

[4] M. Guajardo and M. Rönnqvist, "Operations research models for coalition structure in collaborative logistics," *European Journal of Operational Research*, vol. 240, no. 1, pp. 147–159, 2015.

[5] H. M. Salkin and C. H. Lin, "Note-aggregations of subsidiary firms for minimal unemployment compensation payments via integer programming," *Management Science*, vol. 25, no. 4, pp. 405–408, 1979.

[6] C.-H. M. Lin and H. M. Salkin, "An efficient algorithm for the complete set partitioning problem," *Discrete Applied Mathematics*, vol. 6, no. 2, pp. 149–156, 1983.

[7] F. Basso, M. Guajardo, and M. Varas, "Collaborative job scheduling in the wine bottling process," *Omega*, vol. 91, p. 102021, 2020.

[8] F. Basso, S. D'Amours, M. Rönnqvist, and A. Weintraub, "A survey on obstacles and difficulties of practical implementation of horizontal collaboration in logistics," *International Transactions in Operational Research*, vol. 26, no. 3, pp. 775–793, 2019.

[9] M. Frisk, M. Göthe-Lundgren, K. Jörnsten, and M. Rönnqvist, "Cost allocation in collaborative forest transportation," *European Journal of Operational Research*, vol. 205, no. 2, pp. 448–458, 2010.

[10] J.-F. Audy, S. D'Amours, and L.-M. Rousseau, "Cost allocation in the establishment of a collaborative transportation agreement-an application in the furniture industry," *Journal of the Operational Research Society*, vol. 62, no. 6, pp. 960–970, 2011.

[11] E. T. Bell, "Exponential polynomials," *The Annals of Mathematics*, vol. 35, no. 2, pp. 258–277, 1934.

[12] G.-C. Rota, "The number of partitions of a set," *The American Mathematical Monthly*, vol. 71, no. 5, pp. 498–504, 1964.

[13] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1-2, pp. 209–238, 1999.

[14] D. Y. Yeh, "A dynamic programming approach to the complete set partitioning problem," *BIT Numerical Mathematics*, vol. 26, no. 4, pp. 467–474, 1986.

[15] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci, "An anytime algorithm for optimal coalition structure generation," *Journal of Artificial Intelligence Research*, vol. 34, pp. 521–567, 2009.

[16] T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, and N. R. Jennings, "A hybrid exact algorithm for complete set partitioning," *Artificial Intelligence*, vol. 230, pp. 14–50, 2016.

[17] T. Yang, "An evolutionary simulation-optimization approach in solving parallel-machine scheduling problems—a case study," *Computers & Industrial Engineering*, vol. 56, no. 3, pp. 1126–1136, 2009.

[18] J. Holland, *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, USA, 1975.

[19] D. Levine, "A parallel genetic algorithm for the set partitioning problem," in *Meta-Heuristics*, pp. 23–35, Springer, Berlin, Germany, 1996.

[20] P. C. Chu and J. E. Beasley, "Constraint handling in genetic algorithms: the set partitioning problem," *Journal of Heuristics*, vol. 4, no. 4, pp. 323–357, 1998.

[21] S. Vaziri, F. Etebari, and B. Vahdani, "Development and optimization of a horizontal carrier collaboration vehicle routing model with multi-commodity request allocation," *Journal of Cleaner Production*, vol. 224, 2019.

[22] S. Ronald, "Robust encodings in genetic algorithms: a survey of encoding issues," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pp. 43–48, IEEE, Indianapolis, IN, USA, April 1997.

[23] D. Schmeidler, "The nucleolus of a characteristic function game," *SIAM Journal on Applied Mathematics*, vol. 17, no. 6, pp. 1163–1170, 1969.

[24] J. P. Arabeyre, J. Fearnley, F. C. Steiger, and W. Teather, "The airline crew scheduling problem: a survey," *Transportation Science*, vol. 3, no. 2, pp. 140–163, 1969.

[25] M. Deveci and N. Ç. Demirel, "A survey of the literature on airline crew scheduling," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 54–69, 2018.

[26] C.-H. Lin, *Corporate tax structures and a special class of set partitioning problems*, Ph.D. thesis, Case Western Reserve University, Cleveland, OH, USA, 1975.

[27] H. Horn and L. Persson, "The equilibrium ownership of an international oligopoly," *Journal of International Economics*, vol. 53, no. 2, pp. 307–333, 2001.

[28] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*, Cambridge University Press, Cambridge, UK, 2012.

[29] F. Basso, *Collaborative systems in logistics and transportation*, Ph.D. thesis, Universidad de Chile, Santiago, Chile, 2018.

[30] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[31] A. Björklund, T. Husfeldt, and M. Koivisto, "Set partitioning via inclusion-exclusion," *SIAM Journal on Computing*, vol. 39, no. 2, pp. 546–563, 2009.

[32] T. Rahwan, T. Michalak, N. Jennings, M. Wooldridge, and P. McBurney, "Coalition structure generation in multi-agent systems with positive and negative externalities," in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, Pasadena, CA, USA, July 2009.

[33] B. Banerjee and L. Kraemer, "Coalition structure generation in multi-agent systems with mixed externalities," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Toronto, Canada, pp. 175–182, May 2010.

[34] T. Rahwan, T. Michalak, M. Wooldridge, and N. R. Jennings, "Anytime coalition structure generation in multi-agent

systems with positive or negative externalities," *Artificial Intelligence*, vol. 186, pp. 95–122, 2012.

[35] S. Ueda, A. Iwasaki, V. Conitzer, N. Ohta, Y. Sakurai, and M. Yokoo, "Coalition structure generation in cooperative games with compact representations," *Autonomous Agents and Multi-Agent Systems*, vol. 32, no. 4, pp. 503–533, 2018.

[36] S. Ieong and Y. Shoham, "Marginal contribution nets: a compact representation scheme for coalitional games," in *Proceedings of the 6th ACM Conference on Electronic Commerce*, pp. 193–202, Vancouver, Canada, July 2005.

[37] T. Michalak, D. Marciniak, M. Szamotulski et al., "A logic-based representation for coalitional games with externalities," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Toronto, Canada, May 2010.

[38] O. Skibski, T. P. Michalak, Y. Sakurai, M. Wooldridge, and M. Yokoo, "A graphical representation for games in partition function form," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, TX, USA, January 2015.

[39] X. Liao, M. Koshimura, K. Nomoto, S. Ueda, Y. Sakurai, and M. Yokoo, "Improved WPM encoding for coalition structure generation under MC-nets," *Constraints*, vol. 24, no. 1, pp. 25–55, 2019.

[40] A. Zha, K. Nomoto, S. Ueda, M. Koshimura, Y. Sakurai, and M. Yokoo, "Coalition structure generation for partition function games utilizing a concise graphical representation," in *Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems*, pp. 143–159, Springer, Nice, France, Nice, France, October 2017.

[41] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[42] S. Sen and P. S. Dutta, "Searching for optimal coalition structures," in *Proceedings of the Fourth International Conference on MultiAgent Systems, 2000*, pp. 287–292, IEEE, Boston, MA, USA, Boston, MA, USA, July 2000.

[43] H. Keinänen, "Simulated annealing for multi-agent coalition formation," in *Proceedings of the KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Springer, Serena Villata, pp. 30–39, May 2009.

[44] N. Di Mauro, T. M. Basile, S. Ferilli, and F. Esposito, "Coalition structure generation with grasp," in *Proceedings of the International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pp. 111–120, Springer, Varna, Bulgaria, Varna, Bulgaria, September 2010.

[45] D. S. Dos Santos and A. L. C. Bazzan, "Distributed clustering for group formation and task allocation in multiagent systems: a swarm intelligence approach," *Applied Soft Computing*, vol. 12, no. 8, pp. 2123–2131, 2012.

[46] A. Farinelli, M. Bicego, S. Ramchurn, and M. Zucchelli, "C-link: a hierarchical clustering approach to large-scale near-optimal coalition formation," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, Beijing, China, July 2013.

[47] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier, Amsterdam, Netherlands, 2014.

[48] D. Whitley and A. M. Sutton, "Genetic algorithms—a survey of models and methods," in *Handbook of Natural Computing*, pp. 637–671, Springer, Berlin, Germany, 2012.

[49] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[50] D. Beasley, D. R. Bull, and R. R. Martin, *An Overview of Genetic Algorithms: Part 1, Fundamentals*, pp. 56–69, University of Cardiff, Cardiff, UK, 1993.

[51] D. Beasley, D. R. Bull, and R. R. Martin, *An Overview of Genetic Algorithms: Part 2, Research Topics*, pp. 170–181, University of Cardiff, Cardiff, UK, 1993.

[52] C. K. H. Lee, "A review of applications of genetic algorithms in operations management," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 1–12, 2018.

[53] A. K. Kar, "Bio inspired computing—a review of algorithms and scope of applications," *Expert Systems with Applications*, vol. 59, pp. 20–32, 2016.

[54] V. Venugopal and T. T. Narendran, "A genetic algorithm approach to the machine-component grouping problem with multiple objectives," *Computers & Industrial Engineering*, vol. 22, no. 4, pp. 469–480, 1992.

[55] J. F. Gonçalves and M. G. Resende, "An evolutionary algorithm for manufacturing cell formation," *Computers & Industrial Engineering*, vol. 47, no. 2-3, pp. 247–273, 2004.

[56] M. Boulif, "Genetic algorithm encoding representations for graph partitioning problems," in *Proceedings of the 2010 International Conference on Machine and Web Intelligence (ICMWI)*, pp. 288–291, IEEE, Algiers, Algeria, Algiers, Algeria, January 2010.

[57] M. Pirlot, "General local search methods," *European Journal of Operational Research*, vol. 92, no. 3, pp. 493–511, 1996.

[58] P. Chu and J. Beasley, *A Genetic Algorithm for the Set Partitioning Problem*, Imperial College, London, UK, 1995.

[59] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[60] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.

[61] M. Solar, V. Parada, and R. Urrutia, "A parallel genetic algorithm to solve the set-covering problem," *Computers & Operations Research*, vol. 29, no. 9, pp. 1221–1235, 2002.

[62] K. Kotecha, G. Sanghani, and N. Gambhava, "Genetic algorithm for airline crew scheduling problem using cost-based uniform crossover," in *Proceedings of the Asian Applied Computing Conference*, pp. 84–91, Springer, Kathmandu, Nepal, Kathmandu, Nepal, October 2004.

[63] S. Chatterjee, C. Carrera, and L. A. Lynch, "Genetic algorithms and traveling salesman problems," *European Journal of Operational Research*, vol. 93, no. 3, pp. 490–510, 1996.

[64] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pp. 154–159, Lawrence Erlbaum, Hillsdale, NJ, USA, 1985.

[65] L. D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesmen: the genetic edge recombination operator," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 133–140, Fairfax, VA, USA, June 1989.

[66] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.