Swarm and Evolutionary Computation 53 (2020) 100643

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Metaheuristics to solve grouping problems: A review and a case study

Octavio Ramos-Figueroa [a,*], Marcela Quiroz-Castellanos [a], Efrén Mezura-Montes [a], Oliver Schütze [b,c]

[a] Universidad Veracruzana, Artificial Intelligence Research Center, Xalapa, Mexico
[b] Cinvestav-IPN, Campus Zacatenco, Computer Science Department, Mexico City, Mexico
[c] Universidad Autonóma Metropolitana, Campus Cuajimalpa, Dr. Rodolfo Quintero Ramirez Chair, Mexico City, Mexico

ARTICLE INFO

Keywords:
Combinatorial optimization
Grouping problems
Metaheuristics
Parallel-machine scheduling

ABSTRACT

Grouping problems are a special type of combinatorial optimization problems that have gained great relevance because of their numerous real-world applications. The solution process required by some grouping problems represents a high complexity, and currently, there is no algorithm to find the optimal solution efficiently in the worst case. Consequently, the scientific community has classified such problems as NP-hard. For the solution of grouping problems, numerous elaborate procedures have been designed incorporating different techniques. The specialized literature includes enumerative methods, neighborhood searches, evolutionary algorithms as well as swarm intelligence algorithms. In this study, a review of twenty-two NP-hard grouping problems is carried out, considering seventeen metaheuristics. The state-of-the-art suggests that Genetic Algorithms (GAs) have shown the best performance in most of the cases, and the group-based representation scheme can be used to improve the performance of different metaheuristics designed to solve grouping problems. Finally, a case study is presented to compare the performance of three metaheuristic algorithms with different representation schemes for the Parallel-Machine Scheduling problem with unrelated machines, including the GA with extended permutation encoding, the Particle Swarm Optimization (PSO) with machine-based encoding, and the GA with group-based encoding. Experimental results suggest that the GA with the group-based encoding is the best option to address this problem.

1. Introduction

Combinatorial optimization is a field of research of great interest to the scientific community in areas such as artificial intelligence and operations research. Applications of combinatorial optimization problems arise in several relevant contexts such as engineering, science, economics and everyday life. For this reason, several studies have been carried out to produce practical and theoretical knowledge that help to solve these problems efficiently.

In general, the optimization of a combinatorial problem consists of finding suitable values for a set of discrete variables, so that the objective function is minimized or maximized, according to the case, satisfying the given conditions and constraints. Algorithms developed to solve this type of problems look for a feasible disposition, grouping, order or selection of discrete objects, usually finite in number, that optimizes the case study [1].

Dealing with combinatorial optimization problems is considered an intriguing issue. Many of these problems are quite easy to define but require a solution process of high complexity; and currently, there is no algorithm to find their optimal solution efficiently, in the worst case. Such problems have been classified by the scientific community as belonging to the NP-hard class and are considered intrinsically intractable from a computational point of view [2].

Specialized literature contains an extensive diversity of algorithms designed to solve NP-hard combinatorial optimization problems. Nevertheless, the most relevant results have been obtained using metaheuristic algorithms, solution methods with the ability to find high-quality solutions in a relatively short time [3]. However, despite the efforts of the scientific community in the development of new strategies, to date, there is no algorithm considered the best option for all possible situations [4]. Accordingly, the development of high-performance algorithms for NP-hard combinatorial optimization problems is an open

* Corresponding author.
E-mail addresses: oivatco.rafo@gmail.com (O. Ramos-Figueroa), maquiroz@uv.mx (M. Quiroz-Castellanos), emezura@uv.mx (E. Mezura-Montes), schuetze@cs.cinvestav.mx (O. Schütze).

https://doi.org/10.1016/j.swevo.2019.100643
Received 31 August 2019; Received in revised form 8 December 2019; Accepted 31 December 2019
Available online 6 January 2020

field of research and specialists throughout the world work on the creation of new solution strategies.

In this research, a particular class of combinatorial optimization problems is considered, knowing as grouping problems. Nowadays, solving such problems has become an important issue, mainly because many of them occur in different applications of industry, hospitals, and seaports, to mention a few examples. The foremost motivation for writing this paper is to present an overview of the current trends in metaheuristic algorithms designed to tackle grouping problems. Furthermore, a case study is presented to analyze three different approaches for the solution of the Parallel-Machine Scheduling problem with unrelated machines and *makespan* minimization, applying, for the first time, two state-of-the-art metaheuristics: a Grouping Genetic Algorithm (GGA) and a Particle Swarm Optimization (PSO).

The document continues as follows: in Section 2 we describe the main characteristics of grouping problems, as well as some ways to classify them, we also present a list of the grouping problems considered for this survey. In Section 3, we review the most representative metaheuristics for solving grouping problems, considering neighborhood searches, evolutionary algorithms, and swarm intelligence techniques. In Section 4, we present the conclusions of the survey, and we discuss further research directions and problems open to research. In Section 5, we present the study of the performance of three metaheuristics for the Parallel-Machine Scheduling problem. Finally, in Section 6, we summarise our findings and conclusions.

2. Grouping problems

Grouping problems are a type of combinatorial optimization problems where a set V of n items is usually partitioned into a collection of mutually disjoint subsets (groups) G_i, so that: $V = \cup_{i=1}^{D} G_i$ and $G_i \cap G_j = \emptyset, i \neq j$. In this way, an algorithm designed to solve a grouping problem seeks the best possible distribution of the n items of the set V in D $(1 \leq D \leq n)$ different groups, such that each item is exactly in one group. In most grouping problems, the assignment of the items has to be carried out in such a way that a set of constraints is satisfied. Therefore, all possible groupings are not allowed [5]. As a result of these characteristics, real-world grouping problems are very difficult to solve. Some of the most important grouping problems are in fact NP-hard, which implies there exists no algorithm finding in polynomial time for every instance an optimal solution [2]. This review contemplates twenty-two grouping problems. Each of these problems has been proven to be NP-hard, and complexity studies have motivated the development of metaheuristic algorithms to solve them. In the next paragraphs, we present a description of each problem considered in this survey.

Bin Packing (BP) is a well-known problem dealing with the challenge of packing a set $V = \{1, \ldots, n\}$ of n indivisible items with associated sizes s_i $(i = 1, \ldots, n)$, into the minimum number of bins D without violating any of the capacity constraints. BP is a classical grouping problem with many potential real-world applications; therefore, there exist several variants and generalizations of this problem. This survey includes fourteen examples of metaheuristic algorithms that have been applied to solve BP [6–19].

Load Balancing (LB) is a critical issue for the efficient operation of scheduling systems. The goal is to supply the demand for shared resources, providing inclusive access to the resources according to the needs, with low management scramble and interplay. Hence, the problem consists of organizing the D available resources to assign them to the collection $V = \{1, \ldots, n\}$ of n requested tasks, promoting a LB. This problem has been extensively studied, mainly in applications related to scheduling in cloud environments. This work holds eleven metaheuristic algorithms implemented to solve LB [20–30].

Assembly Line Design (ALD) is a grouping problem emerging in the industry, specifically in manufacturing. There exist several variants of this problem, comprising different types of flow lines producing all kinds of products. The general definition of ALD is as follows: given a set of D stations connected through a material handling system and a collection $V = \{1, \ldots, n\}$ of n tasks needed to manufacture a product (or service), design the most efficient assembly line (maximizing the throughput and reducing the costs). ALD is one of the most studied grouping problems; therefore, the state-of-the-art holds several metaheuristics designed to solve it. This review includes examples of fifteen metaheuristics [31–45].

Vehicle Routing Problem (VR) is a much-studied problem because it emerges in many real-world situations, mainly related to logistics and transportation. In this problem, given a depot company with known location and unlimited capacity, a fleet of vehicles D with known capacity, and a set $V = \{1, \ldots, n\}$ of n customers with known demand and location. The objective is to find an efficient distribution of the products from the depot to the customers, supplying the total demand of the customers without exceeding the capacity of vehicles, and minimizing the total distance of the route and the time spent by the fleet of vehicles. The state-of-the-art includes several metaheuristics designed to solve VR. This review contemplates fourteen examples [46–59].

Cell Formation (CF) is a problem arising during the layout of the restructuring or the study of new production systems. Specifically, when using the principle of group technology, which promotes the separation of a whole workgroup into small cells. In those situations, the design of independent cells is not possible most of the time. Therefore, the challenge in the CF problem is to minimize traffic of a set $V = \{1, \ldots, n\}$ of n items between D cells, for a given maximum cell size. Over the last years, this problem has been addressed using different metaheuristic algorithms. This survey involves thirteen examples [60–72].

Multiple Traveling Salesperson (MTS) is an extension of the well-known traveling salesperson problem. In this variant, given a set D of salespersons and a collection $V = \{1, \ldots, n\}$ of n cities to visit, the objective is to find a suitable assignation of the n cities to the D salespersons, coupled with an efficient visit order of the cities assigned to each salesperson. The characteristics of MTS are present in many real-life applications, which has motivated the design of numerous metaheuristic algorithms to solve it. This review contemplates eleven metaheuristic algorithms used to solve this problem [73–83].

Facility Location (FL) is a problem of great practical importance for its economic profit. This problem deals with the location of a collection of D facilities (e.g., industrial plants, warehouses, and networks), to minimize the cost (or to maximize the profit), by supplying the demand of all the set $V = \{1, \ldots, n\}$ of n clients. FL is one of the grouping problems most studied over the last years. This review contemplates thirteen examples of metaheuristic algorithms used to solve it [84–96].

Job Shop Scheduling (JSS) is a problem consisting of a set of D machines that perform operations on jobs. In this problem, each job has a specific processing order (i.e., a job is made up of a collection of operations ordered according to the time required by the machine that processes it). Therefore, the problem is to find an efficient schedule of a set $V = \{1, \ldots, n\}$ of n operations of each job in the D available machines. According to Ref. [97], more than 60% of the proposals to solve the Job Shop Scheduling problems are related to swarm intelligence and evolutionary algorithms, due to their exponential complexity. In the state-of-the-art, JSS is one of the most studied grouping problems; this survey contemplates sixteen examples of metaheuristics used to solve it [97–113].

Team Formation (TF) is a problem facing the challenge of identifying D individuals that match a required set $V = \{1, \ldots, n\}$ of n skills to create a group that maximizes one or several social positive attributes. TF has been used to improve the management of human resources in companies and organizations. This review considers eight metaheuristic algorithms designed to solve this problem [114,114–120].

Modular Product Design (MPD) is a production architecture, in which different products are generated, by forming distinct combinations of modules. Thus, a modular product consists of detachable modules sharing functional, physical, and other interface characteristics of a specific product family. MPD has been addressed as a grouping problem, consisting of packing a set $V = \{1, \dots, n\}$ of n components into an appropriate number of modules D, maximizing the physical and functional relations among the components. Currently, the state-of-the-art includes scarce studies on metaheuristics solving MPD problems; therefore, this review only considers four examples of metaheuristic algorithms used to solve it [121–124].

Multivariate Microaggregation (MM) is a method used for masking microdata to protect privacy in databases and location-based services. Given a microdata set $V = \{1, \dots, n\}$ of n records with x attributes, this technique group the records in D clusters, maximizing the within-groups homogeneity, and gives them a value z, in such a way that all the records are indistinguishable using these values. The process of MM has been formulated as a grouping problem and tackled using few metaherustics; consequently, this survey only contemplates three exemples [125–127].

Multiple Knapsack (MK) is an extension to the traditional Knapsack Problem, consisting of assigning a set $V = \{1, \dots, n\}$ of n objects of different weights and values (profits) to a collection of knapsacks of various capacities, to maximize the total profit. This problem has several variants and generalizations, and it is present in many real-world applications like continuous double-call auctions, multiprocessor scheduling, and vehicle/container loading. This review considers eleven metaheuristics designed to solve MK [128–138].

Cutting Stock (CS) is a problem frequently emerging in industries with a manufacturing process requiring to cut material with standard dimensions into pieces of different sizes and forms. The objective is to minimize the waste and the costs of the cutting process tackled considering its technical constraints. Thus, CS problem can be generalized as follows. Given a set $V = \{1, \dots, n\}$ of n pieces with different sizes and forms, group them into D pieces of the material with standard dimension to be cut, to minimize the waste. CS is one of the most studied grouping problems; as a result, it has been addressed using different metaheuristics. This survey contemplates eleven exemples [139–149].

The Timetabling Problem (TP) emerges in many real-world applications related to the assignation of timetables periods to specific entities (e.g., people, tasks, vehicles, lectures, exams, meetings, etc.) considering simultaneously several criteria (e.g., length of the schedule, utilization of resources, the satisfaction of people's preferences, and compliance with regulations). A typical grouping problem with those characteristics is the well-known university timetabling problem, consisting of assigning a set $V = \{1, \dots, n\}$ of n events in t timetables periods and D rooms taking into account the constraints of the addressed problem. Currently, specialized literature includes several metaheuristics designed to solve TP. This survey considers thirteen examples [150,150–161].

Clustering Problem (CP) is given by an unsupervised learning technique designed to group data objects into groups of disjoint clusters. Over the last decades, a large variety of clustering problems have been identified, including science, engineering, and economics problems. CP can be declared as follows. Given a dataset $V = \{1, \dots, n\}$ of n tuples, finding the minimum number of clusters D required to distribute the n tuples, reducing as much as possible the overlap. This survey includes sixteen examples of metaheuristic algorithms that have been applied to solve CP [162–177].

Maximally Diverse (MD) is a problem consisting of clustering a collection $V = \{1, \dots, n\}$ of n elements into D mutually disjoint groups, maximizing the diversity among the elements in every group. This phenomenon frequently occurs in many applications related to data mining. The state-of-the-art includes some proposals that tackle MD using different metaheuristics. This review contemplates seven examples [178–182].

Supplier Selection (SS) is a problem consisting of electing the best supplier from a group of D prespecified candidates, considering a set $V = \{1, \dots, n\}$ of n qualitative and quantitative attributes. Therefore, the objective is to find the supplier with suitable trade-off between such quantifiable and unquantifiable attributes. According to the state-of-the-art, this problem has been addressed using different metaheuristics. This survey considers eight examples [183,183–189].

Graph Coloring (GC) is a problem that focuses on the assignation of colors to certain elements of a graph, considering different constraints. Considering its most common version, the Vertex Coloring problem, GC can be defined as follows, given D colors, find a way of coloring a set $V = \{1, \dots, n\}$ of n vertices of a graph, taking into account that the same color can not be used to color any pair of adjacent vertices. This survey includes thirteen examples of metaheuristic algorithms that have been applied to solve GC [190–201].

Parallel Machine Scheduling (PMS) is a problem emerging in industrial situations where there is a collection of machines that perform one or more jobs. PMS can be generalized as scheduling a collection of n jobs $V = \{1, \dots, n\}$ in a set of D machines, in such a way that each machine can process one job at the time, and every job is assigned to exclusively one machine. The aim is to find a schedule that optimizes a certain performance measure. This review contemplates fourteen examples of metaheuristic algorithms that have been used to solve PMS [202–214].

Order Batching (OB) is a problem focused on the efficient planning and control of warehouse operations (e.g., receiving, storage, order picking, and shipping), for managing supply chains so that the customer service is satisfactory provided with the lowest possible cost. Given the above, the picking OB deals with the problem of partitioning a set $V = \{1, \dots, n\}$ of n orders into D groups, to minimize the total length of all needed tours through the warehouse for collecting all requested articles. This review considers eleven examples of metaheuristic algorithms that have been employed to solve OB [19,19,215–223].

Home Health Care (HHC) is a problem centered on the improvement of medical and paramedical services to people who need specialized care at their homes (e.g., patients with physical or mental challenges, patients with a terminal illness, and elderly people, to mention some cases). Considering one of the most common variants, the HHC staff schedule, this problem can be defined as follows, given a set $V = \{1, \dots, n\}$ of n patients that can be visited by one of D caregivers, look for the schedule that minimize the cumulative cost incurred without exceeding the cumulative load of none caregiver [224]. This survey contemplates nine examples of metaheuristic algorithms that have been employed to solve HHC [224–232].

Stock Portfolio (SP) is a problem facing up the recurrent change of the financial market. SP consist of dividing a set $V = \{1, \dots, n\}$ of n stocks into D portfolios to optimize the portfolio satisfaction factors. During the optimization of an SP from a given financial dataset, several factors should be considered. The state-of-the-art holds different metaheuristics proposed for efficient asset management in stock portfolio optimization. This survey contemplates thirteen examples [233–245].

Table 1 lists the NP-hard grouping problems contemplated for this review. In this chart, the first column indicates the name of each grouping problem, followed by the abbreviation used to refer to it in the future, as well as the number of related works reviewed. This research considers only twenty-two grouping problems. Nevertheless, it is important to highlight that there exist several grouping problem variants and real-world applications of the selected problems. Moreover, the list of problems and papers presents only a sample of the recent work in this field.

Usually, the objective function of grouping problems is stated based on the composition of each group and relations among the groups and the elements belonging to the same collection. In this sense, there are different ways to classify grouping problems. For example, Kashan et al. proposed to organize them based on 1) the number of groups, 2) the type of groups and, 3) the dependence on the groups' order [246].

According to the number of groups, grouping problems can have a *constant* or *variable* number of groups [246]. In this sense, *constant* grouping problems receive the number of groups (D) as an input to the problem. That is the case of the Parallel-Machine Scheduling (PMS)

Table 1
Grouping problems included in this survey. Problem: name of the problem. Abbr: abbreviation. P: number of papers reviewed for each problem.

Problem	Abbr.	P
Bin Packing	BP	14
Load Balancing	LB	11
Assembly Line Design	ALD	15
Vehicle Routing	VR	14
Cell Formation	CF	13
Multiple Traveling Salesperson	MTS	11
Facility Location	FL	13
Job Shop Scheduling	JSS	16
Team Formation	TF	8
Modular Product Design	MPD	4
Multivariate Microaggregation	MM	3
Multiple Knapsack	MK	11
Cutting Stock	CS	11
Timetabling Problem	TP	13
Clustering Problem	CP	16
Maximally Diverse	MD	7
Supplier Selection	SS	8
Graph Coloring	GC	13
Parallel-Machine Scheduling	PMS	14
Order Batching	OB	11
Home Health Care	HHC	9
Stock Portfolio	SP	13

problem [247], where the objective is to find an efficient scheduling of a set of jobs in D parallel machines. In contrast, in *variable* grouping problems the value of D is not known. Thus, it is necessary to find a feasible grouping that optimizes D. For example, Bin Packing (BP) [18] is a *variable* grouping problem, where it is needed to determine a suitable packing of a given set of items, such that all the items are packed using as few bins (D) as possible.

On the other hand, grouping problems can be divided into problems with *identical* and *non − identical* groups, considering the characteristics of their groups [246]. In this fashion, if exchanging all the elements of two groups in a solution of a particular problem makes a different solution, that problem has *non − identical* groups. Contrariwise, if the said swapping process does not change the properties of the solution, such grouping problem belongs to the *identical* groups class. In this manner, the Graph Coloring (GC) problem [195] has *identical* groups since exchanging the color of two groups of vertices does not affect a solution; while the Multiple Knapsack (MK) problem [131] is a well-known grouping problem with *non − identical* groups since the knapsacks have different capacities and exchanging the items of two knapsacks could even produce an unfeasible solution.

Finally, in several grouping problems, the quality of the solutions depends on the groups' order. Such problems are called *order dependent* [246]. One of the most well-known problems in this class is the Timetabling Problem (TP) [160]. For example, the timetabling of the exams in a university, where the task is to organize a set of exams into a certain number of time-slots, such that students are not forced to be in two places at once. Conversely, all the grouping problems without such order dependency are so-called *not order dependent*.

It is important to note that each grouping problem can be classified using the three criteria already mentioned. In this way, the Parallel-Machine Scheduling problem with unrelated machines is a grouping problem where the number of groups is *constant*, the groups characteristic are *not identical*, and the solution quality is *order dependent* [204].

In addition to this, Mutingi and Mbohwa organized the grouping problems based on their application area [248]. They divided the grouping problems into healthcare, tertiary education, warehouse and logistics, manufacturing systems, design, maintenance management, business economics, and human resource management. In this fashion, Bin Packing (BP) [6] belongs to the warehouse and logistics category; Assembly Line Design (ALD) [31], Cell Formation (CF) [65], and Parallel-Machine Scheduling (PMS) [249] are in the manufacturing systems section; the Home Health Care (HHC) problems [225] are in the healthcare class; while Modular Product Design (MPD) [122] belongs to design class.

3. Metaheuristic algorithms in grouping problems

The specialized literature includes several algorithms designed to solve grouping problems using different approaches like traditional mathematical methods, dynamic programming, simple heuristics, enumerative methods, and metaheuristics. In this research, only the most representative metaheuristics are surveyed, comprising four neighborhood searches, seven evolutionary algorithms, as well as six swarm intelligence algorithms. Metaheuristic methods have been broadly used to solve grouping problems, because of its general nature that allows them to be efficient in different problems without significant changes [69]. It is important to note that the state-of-the-art also includes hybrid metaheuristics (i.e., solution methods made up by a metaheuristic that work as the principal search motor and one local search that works in a second level) [65,204,212,233,247]. In this review, metaheuristics in hybrid or memetic algorithms are identified and considered individually. The readers interested in the development of hybrid metaheuristics are referred to Ref. [250] for useful guidelines.

3.1. Neighborhood searches

Neighborhood searches also known as trajectory searches work with a single solution. In general, the search process of metaheuristics designed using this approach consists of generating and exploring the neighbors of the current solution. Neighbor solutions are created using different techniques; for example, modifying the value of one variable or exchanging two or more elements. Hence, the neighborhood size depends on the strategy used to generate the neighbors. The performance of neighborhood searches has been examined solving different grouping problems. For some problems, it has been demonstrated that simple local search methods can outperform sophisticated hybrid methods, producing high-quality solutions in a short time [251,252]. An example is the work of Santos *et al.*, where the performance of different neighborhood searches designed for solving the unrelated Parallel-Machine Scheduling (PMS) problem with sequence-dependent setup times is investigated, analyzing different neighborhood structures, diversification and intensification strategies and parameter-tuning challenges [253]. For this review, four neighborhood metaheuristics are considered: Hill Climbing (HC) [6], Variable Neighborhood Search (VNS) [216], Simulated Annealing (SA) [8] and Tabu Search (TS) [49].

The Hill Climbing (HC) algorithm receives its name because it manages an iterative improvement strategy. In this form, the neighborhood of the current solution is used to increase (improve) its quality each cycle of the search process. Regularly, HC starts from an arbitrary solution (current solution). Then, iteratively tests new candidate solutions in the neighborhood of the current solution and adopts the new ones if they are better. HC techniques have been very efficient when they are combined with other approaches like dynamic programming and genetic algorithms to create hybrid and memetic algorithms. Until now, HC has been used to address thirteen of the twenty-two grouping problems contemplated in this review [6,20,33,47,85,100,129,139,150,162,190,215,233]. For example, Kato *et al.* introduced one of the last related works on solving grouping problems using HC [254]. In this work, the authors present a hybridization of HC with Particle Swarm Optimization (PSO) to address an extension to the traditional Job Shop Scheduling (JSS) widely reported in the literature. The performance of the proposed algorithm was compared against other hybridizations of PSO and different local searches of the state-of-the-art algorithms, showing better results.

Similarly, Variable Neighborhood Search (VNS) is an iterative improvement search; however, this method explores increasingly distant neighborhoods, one at the time. Starting from an initial incumbent solution, a random solution is generated from the current neighborhood and a local search is applied to get a local optimum. If a new incumbent solution is found (the local optimum better than the current incumbent solution), the process is repeated starting with the first neighborhood. Otherwise, the process is repeated with the next neighborhood (which is typically larger). VNS has proven its efficiency in several grouping problems, working within cooperative approaches with exact techniques and as a part of hybrid algorithms combined with other metaheuristics. Until now, VNS has been adapted to tackle seventeen of the twenty-two grouping problems contemplated in this review [9,21,34,48,62,73,86,99,128,140,151,163,179,192,202,216,225]. Among the most recent efforts to address grouping problems using VNS is the work presented by Santos *et al.*, in 2019 [9]. In this study, the authors solve large instances of a new variant of the Bin Packing (BP) problem with a simple VNS, improving the results of sophisticated procedures.

On the other hand, Simulated Annealing (SA) takes as inspiration the chemical process of metal annealing. SA attempts to reduce the probability of becoming stuck in a local optimum by sometimes accepting neighbors with a lower quality than the current solution. In this algorithm, if a new solution in the neighborhood if better than the current solution, it will always replace it. But if the neighbors of the current solution are worse than the current solution, they are evaluated to decide probabilistically if a transition to a new solution is made or not. SA has been applied successfully to numerous grouping problems, and several modifications of the accepting rules and hybridizations with other metaheuristics have improved the performance of this algorithm. According to the scope of this review, eighteen of the twenty-two grouping problems contemplated in this review have been addressed employing SA [8,22,31,46,60,74,87,98,114,141,152,164,179,191,203,217,226,234]. In 2019, Leite *et al.* presented one of the last proposals to solve grouping problems with SA [152]. In this work, the authors assessed the performances of a fast SA to solve the Timetabling Problem (TP). They performed an experimental study solving the 2nd International Timetabling Competition (ITC 2007) benchmark set with SA. Experimental results indicate that SA improves on one out of twelve instances, and ranks third among the five best algorithms.

The last neighborhood search contemplated for this review is Tabu Search (TS), which uses the concept of memory and implements it through simple structures. TS works as an ordinary descent method that only permits to move to neighbors that improve the quality of the current solution. However, a short-term memory, known as the tabu list, stores recently visited solutions (or their attributes) to expand the local search and escape from local optimums. Thus, the neighborhood of the current solution is restricted, considering the solutions that do not belong to the tabu list. Over the years, several developments and refinements of TS have been proposed, including different forms of memories, as well as hybridizations with other techniques. Thanks to these characteristics, TS has been used to deal favorably with nineteen of the twenty-two grouping problems contemplated in this survey [7,23,32,49,61,75,84,101,130,142,153,165,178,190,204,218,227,235,255]. One of the last applications of TS for solving grouping problems is the work presented by Peng *et al.*, who addressed the Job Shop Scheduling (JSS) problem combining TS with a genetic algorithm. Experimental results pointed out that this hybrid procedure has high optimization performance and practical value in the field of JSS [256].

It is important to note that the state-of-the-art holds other local search metaheuristic algorithms applied to solve grouping problems, like Greedy Randomized Adaptive Search Procedure (GRASP) [257,258] and Iterated Local Search (ILS) [259,260]. However, this review considers only the most representative, that is, the four already mentioned. Table 2 indicates the neighborhood algorithms (rows) used to solve each grouping problem (columns), according to the main research results in the field. Cells contain related work references, while empty cells indicate that we did not find any reference in specialized literature about it. Therefore, Tabu Search is the most used neighborhood search since it has been used to solve nineteen grouping problems. In contrast, Hill Climbing has been applied to solve thirteen problems only.

3.2. *Evolutionary algorithms*

Evolutionary algorithms are stochastic search methods inspired by the natural biological evolutionary process. These methods simulate some mechanisms of organic evolution, such as mutation, crossover, and natural selection. Until now, different evolutionary algorithms have been proposed, which use several variants of the before-mentioned mechanisms. Evolutionary algorithms used to address grouping problems, so far, include Genetic Algorithms (GA), Evolution Strategies (ES), Evolutionary Programming (EP), Genetic Programming (GP), and Differential Evolution (DE). Over the years, EAs have been widely applied to solve grouping problems with a good measure of success. An excellent overview of algorithms performance and current trends in EAs for Clustering Problems (CP) is presented by Hruschka *et al.* [261]. This paper discusses key issues on the design of EAs for data partitioning problems, such as usually adopted representations, evolutionary operators, and fitness functions.

The Genetic Algorithm (GA) is the most widely known evolutionary algorithm. The basic GA is very generic, and many aspects can be implemented differently according to the problem. The process starts generating a random population of solutions. Then, for a certain number of generations, selected individuals are recombined and mutated to produce better solutions. First, a selection strategy is used to choose individuals, considering their fitness value. Next, the crossover operator is applied to the selected individuals to produce offspring, and these offspring are introduced to the population, employing a replacement strategy. Finally, some individuals are selected to be the subject of slight random perturbations through the mutation operator. The algorithm iterates a predefined number of generations, or until some stopping criterion, related to the problem or the algorithm performance, is met. Many variants of GAs have been developed and have been applied to solve a wide range of grouping problems, including GAs with different schemes for representation of solutions, selection, crossover, replacement, mutation, etc. This metaheuristic has shown promising results in solving grouping problems because it can incorporate new general or specific ideas easily. GAs have been hybridized with many other techniques, achieving high-quality results. Consequently, it has been used to solve the twenty-two grouping problems considered in this review [17,24,39,51,65,76,88,102,115,121,126,132,144,150,166,179,183,193,205,219,228,236]. In Ref. [17] is presented one of the last applications of GA solving grouping problems. In this work, Laabadi *et al.* used GA to solve the a variant of Bin Packing (BP) problem, finding promising results. In Ref. [76], Zhu and Wu present another recent application of GA to solve grouping problems. In that work, the authors proposed an improved model for the optimization of Multiple Traveling Salesperson (MTS) problems with complex topology structure, the model was solved with the GA, showing excellent results.

Evolution Strategies (ES) paradigm was proposed originally by Rechenberg and Schwefel to work in continuous spaces, and later it was adapted to the discrete domain. In general, this evolutionary algorithm begins with a random population consisting of μ solutions; then, in each generation, the μ parents produce λ descendants through recombination and mutation, and the selection operator determines the μ fitter individuals to become the parents of the next generation. Initially, ES was designed to work with one parent and one child. And later, $(\mu + \lambda)$-ES and (μ, λ)-ES extensions were proposed to consider differ-

Table 2
Grouping problems addressed using neighborhood searches.

Metaheuristic	Problem																					
	BP	LB	ALD	VR	CF	MTS	FL	JSS	TF	MPD	MM	MK	CS	TP	CP	MD	SS	GC	PMS	OB	HHC	SP
HC	[6]	[20]	[33]	[47]			[85]	[100]				[129]	[139]	[150]	[162]			[190]		[215]		[233]
VNS	[9]	[21]	[34]	[48]	[62]	[73]	[86]	[99]				[128]	[140]	[151]	[163]	[179]		[192]	[202]	[216]	[225]	
SA	[8]	[22]	[31]	[46]	[60]	[74]	[87]	[98]	[114]				[141]	[152]	[164]	[179]		[191]	[203]	[217]	[226]	[234]
TS	[7]	[23]	[32]	[49]	[61]	[75]	[84]	[101]	[255]			[130]	[142]	[153]	[165]	[178]		[190]	[204]	[218]	[227]	[235]

ent selection schemes, giving rise to the self-adaptation of parameter mechanisms that encode strategy parameters directly onto the chromosome. Nowadays, some efforts have been carried out using ES to solve eleven of the twenty-two grouping problems contemplated in this survey [35,53,63,89,104,116,154,167,180,206,237]. In 2018, Wang *et al.* presented one of the last studies related to the application of ES to solve grouping problems [116]. In this work, the authors analyzed the performance of ES on solving 6000 random classic instances of a variant of the Team Formation (TF) problem, where ES reached the state-of-the-art results. Besides, Wang *et al.* studied the performance of ES addressing 1556 realistic instances, where ES showed outstanding results.

Evolutionary Programming (EP) is quite similar to ES. Nevertheless, in EP, there is no recombination operator because each individual corresponds to a distinct species; furthermore, the selection mechanism is different. In general, EP begins with a random population consisting of μ solutions; then, for a certain number of generations, the μ parent solutions of the current population are mutated to generate μ children. Next, parents and children compete in stochastic round-robin tournaments to be parents of the next generation. According to the state-of-the-art, EP has been used to address six of the twenty-two grouping problems contemplated in this review: Multiple Traveling Salesperson (MTS) [77], Cutting Stock (CS) [145], Cell Formation (CF) [66], Job Shop Scheduling (JSS) [106], Parallel Machine Scheduling (PMS) [207], and Clustering Problem (CP) [168]. For example, Chiong *et al.* used EP to solve the CS problem in Ref. [145]. In this work, the authors conducted an experimental study on solving benchmark problems to asses the performances of EP, getting promising results.

The paradigm of Genetic Programming (GP) adopts a similar search strategy as a GA for creating computer programs. In GP, the chromosomes of the population are not solutions as used in GAs, they are algorithms, represented by tree structures, that, when executed, allow to obtain candidate solutions to the problem at hand. GP starts with an initial population of randomly generated tree structures, iteratively evolved using special genetic operations adapted to work with tree structures. First, the fitness of each chromosome is evaluated, in terms of its performance, on the problem which it represents. Next, the variation operators are selected probabilistically for producing offspring, and a replacement strategy is applied to select the chromosomes for the new population. GP has allowed creating new techniques of solution, going from single assignment rules to more sophisticated methods like hyper-heuristics and hybrid metaheuristics. To date, GP has been used to address twelve of the twenty-two grouping problems contemplated in this review [15,37,52,67,105,134,155,169,186,194,208,238]. One of the grouping problems most studied using GP is the Job Shop Scheduling (JSS) problem [105]. Derived of this, Nguyen *et al.* summarise existing studies in this field until 2019 to provide new paths of work. They pointed out that the use of GP to solve JSS has contributed enough knowledge to the area. Nevertheless, they observed that there still a lack of efficient representations of dispatching rules to enhance the effectiveness of GP to solve JSS.

Besides, Kenneth Price and Rainer Storn introduced the Differential Evolution (DE) algorithm in 1995. Originally, DE was designed to solve continuous problems. However, the state-of-the-art includes some adaptations of DE to deal with grouping problems. Its principal characteristic lies in the variation operators since it uses vector differences to generate new solutions. DE begins generating an initial uniformly distributed random population of individual vectors. At each generation of the evolution process, for each vector in the population, also called target vector, mutation and crossover are applied to produce a trial vector. First, a mutant vector of the target vector is generated, employing a slight random perturbation. Next, the mutant vector and its corresponding target vector are recombined to generate a trial vector. Then, the selection operator compares the fitness of each trial vector to that of its corresponding target vector to determine which one will be maintained into the next generation. Due to its success in solving continuous problems, DE has been extended to solve grouping problems and combined with other metaheuristics in hybrid methods obtaining effective and competitive results. Until now, DE has been used to solve thirteen of the twenty-two grouping problems contemplated in this review [16,38,54,78,91,107,133,156,170,185,196,209,239]. In Ref. [54], one of the most recent applications of DE to solve grouping problems is presented, addressing a real-world application of the Vehicle Routing (VR) problem. The central idea of this work is to design optimal routes minimizing the emission of direct greenhouse gases (i.e., carbon dioxide (CO2), methane (CH4), and nitrous oxide (N2O). From this study, the authors found interesting results for the design of routes with these characteristics.

In 1992, Emanuel Falkenauer introduced one of the most popular algorithms to solve grouping problems, the Grouping Genetic Algorithm (GGA) [262]. In general, this metaheuristic is an extension of the Genetic Algorithm that incorporates a group-based representation scheme. Thus, grouping problems are tackled considering the groups as the unit instead of the elements in each group. Similar to the Genetic Programming algorithm, variation operators must be adapted to work efficiently with group-based solutions encoding. GGAs have shown notable performance solving grouping problems; consequently, it has been employed to address twenty-one of the twenty-two problems surveyed [18,25,40,50,64,79,90,103,117,122,125,131,146,157,171,181,184,195,229,240,263]. In 2019, Singh and Sundar presented one of the most recent related works on solving grouping problems with Grouping Genetic Algorithms (GGA) [181]. In this work, the authors compared the performances of a GGA hybridized with a local search based on a swap strategy against a Tabu Search (TS) and a Variable Neighborhood Search (VNS) on solving the Maximally Diverse (MD) problem. Experimental results suggest that GGA reaches better results than TS and VNS, particularly for larger instances.

Finally, the success of GGAs in solving grouping problems motivated the development of the Grouping Evolution Strategies (GES). An extension of Evolution Strategies that incorporates the grouping representation scheme and a grouping mutation operator. GES is one of the most recent evolutionary algorithms. However, it has already been used to tackle Assembly Line Design (ALD), Bin Packing (BP), Order Batching (OB), Clustering Problem (CP), and the Parallel-Machine Scheduling (PMS) problem showing outstanding results [19,36,172,264]. In 2018, Nejad *et at.* introduced one of the last applications of GES to solve a grouping problem [36]. In this work, the authors studied the performance of GES solving a variant of the ALD problem. They conducted an experimental study using test instances existing in the literature. Experimental results indicate that GES reaches the global solution of most problems of the high dimensional problems.

In the state-of-the-art, it is notable that there are other evolutionary algorithms used to solve grouping problems such as Water Wave Optimization [265,266]. However, this review considers only the most representative, that is, the seven already mentioned. Table 3 includes the evolutionary algorithms (rows) used to address each grouping problem (columns), according to the main research results in the field. Cells contain related work references. Thus, empty cells indicate that we did not find any reference in specialized literature about it. As it can be seen, Genetic Algorithms and Grouping Genetic Algorithms are the most used evolutionary algorithms since they have been used to solve twenty-two and twenty-one of the grouping problems considered in this survey, respectively. In contrast, Evolutionary Programming and Grouping Evolution Strategies have been used to tackle only six and five grouping problems, respectively.

3.3. Swarm intelligence algorithms

Swarm intelligence algorithms emulate the collective self-organized behavior of natural systems for solving problems. In a swarm intelligence, a population of simple agents work together to produce computational intelligence. In recent years, these algorithms have been

Table 3
Grouping problems addressed using evolutionary algorithms.

Metaheuristic	Problem																					
	BP	LB	ALD	VR	CF	MTS	FL	JSS	TF	MPD	MM	MK	CS	TP	CP	MD	SS	GC	PMS	OB	HHC	SP
GA	[17]	[24]	[39]	[51]	[65]	[76]	[88]	[102]	[115]	[121]	[126]	[132]	[144]	[150]	[166]	[179]	[183]	[193]	[205]	[219]	[228]	[236]
ES			[35]	[53]	[63]		[89]	[104]	[116]					[154]	[167]	[180]			[206]			[237]
EP					[66]	[77]		[106]					[145]		[168]				[207]			
GP	[15]		[37]	[52]	[67]			[105]				[134]		[155]	[169]		[186]	[194]	[208]			[238]
DE	[16]		[38]	[54]		[78]	[91]	[107]				[133]		[156]	[170]		[185]	[196]	[209]			[239]
GGA	[18]	[25]	[40]	[50]	[64]	[79]	[90]	[103]	[117]	[122]	[125]	[131]	[146]	[195]	[157]	[181]	[184]	[171]		[263]	[229]	[240]
GES	[19]		[36]												[172]				[264]	[19]		

applied to a wide variety of grouping problems, showing excellent results. The performance of swarm intelligence strategies has been tested solving different problems [214,267,268]. An example is the work of Milan *et al.*, where the advantages and disadvantages of nature-inspired metaheuristics for solving the Load Balancing (LB) problem are analyzed, identifying the most effective techniques [269]. To date, different algorithms have been proposed following this approach, including Ant Colony Optimization (ACO) [147], Particle Swarm Optimization (PSO) [141], Cuckoo Search (CS) [160], Artificial Bee Colony (ABC) [82] and Firefly Algorithm (FA) [161].

The ability of real ants to find the shortest path between their nest and a source of food was the inspiration to develop Ant Colony Optimization (ACO). This metaheuristic was designed to solve discrete problems, specifically for the traveling salesman problem. ACO encodes a given combinatorial optimization problem instance as a fully connected graph whose nodes are components of solutions, and edges are connections between components. A variable called pheromone is associated with each component, which can be read and modified by each ant. This value serves as a form of memory. At each iteration of the algorithm, several artificial ants are considered. Each of them builds a solution to the problem, step by step, adding a feasible solution component, according to a stochastic mechanism that is biased by the pheromone and heuristic information about the problem. Next, the pheromone values are updated to make solution components belonging to good solutions more desirable for ants in future iterations. Many variants of ACO have been created and applied to a wide range of discrete problems, showing outstanding results. The aforementioned motivated the use of ACO in twenty of the twenty-two grouping problems contemplated in this review [10,26,42,56,69,80,92,108,120,127,135,147,158,173,183,197,210,220,230,241]. One of the last proposals on solving grouping problems with ACO is presented by Selvakumar and Guanasekaran, who introduced an enhanced ACO to solve the Load Balancing (LB) problem [26]. Experimental results indicate that the proposed algorithm has a significant improvement over traditional algorithms, with respect to average execution time, average response time, and total cost.

In contrast, Particle Swarm Optimization (PSO) is a metaheuristic created to optimize continuous search spaces. This metaheuristic was designed using as inspiration the swarming and flocking behaviors in animals. PSO begins generating a population of particles with random positions and velocities on the search space. Each particle is a candidate solution to the problem, defined by three vectors in the d-dimensional search space: a velocity vector, a position vector, and a memory vector, which helps it in remembering its fittest known position discovered so far. At each iteration, directed transformations move each particle in the search space in response to discoveries obtained from the environment. First, the velocity of the particle is dynamically adjusted, considering its fittest known position and the position of the best particle among all the particles in its topological neighborhood. Next, the position of the particle is updated, adding the velocity vector to the position vector. Finally, the fitness of the particle is evaluated and, if necessary, the best-discovered locations are updated. Although PSO is planned to deal with continuous domains, it has also been adapted to tackle twenty of the twenty-two grouping problems contemplated in this review [12,27,44,55,68,81,93,109,119,124,136,141,159,174,188,198,211,221,231,242]. In Ref. [44] is presented one of the most recent applications of PSO to solve grouping problems. In this work, PSO was applied to solve a variant of Assembly Line Design (ALD) problem. Experimental results suggest that the proposed approach reached good solutions for all test instances within a short computational time.

In the same way, Xin-She Yang and Suash Deb introduced the Cuckoo Search (CS) metaheuristic in 2009. They took as inspiration the behavior of some cuckoo species during the breeding stage, that lay eggs in a host's nest, removing others' eggs to increase the hatching probability of their own eggs. In the optimization context, each egg in the nest represents a solution, and the cuckoo's egg represents a new solution. When generating a new solution (a cuckoo's egg), a Lévy flight is performed, which essentially provides a random walk. CS starts generating a population of host nests, then for a certain number of iterations, new and potentially better solutions replace solutions in the nests. Each iteration, a new solution is generated, and its quality is compared with another random old solution in the population. In the case the new solution is better, it will replace the old solution. Therefore, a fraction of the worst solutions are replaced by new solutions. According to the scope of the literature review, fifteen of the twenty-two grouping problems contemplated in this review have been addressed employing CS [13,28,41,58,70,95,111,123,148,160,175,189,199,212,243]. In 2019, Karoum and Elbenani analyzed the performances of CS solving the Cell Formation (CF) problem [70]. The results indicate that this metaheuristic is suitable in addressing this problem since it can reach 32 out of 35 benchmark problems (91.43%).

Similarly, the collective behavior of honey bees during the search and exploration of food sources was used by Karaboga in 2005 to develop the Artificial Bee Colony (ABC). Thus, three types of solutions (employee, onlooker, and explorer) are used to carry out an efficient search process. According to the state-of-the-art, sixteen of the twenty-two grouping problems contemplated in this review have been tackled using ABC [29,43,57,71,82,96,112,118,137,176,182,187,200,213,222,244]. One of the most recent works on solving grouping problems using Artificial Bee Colony (ABC) is presented by Davoodi *et al.*, in 2019. In this work, the authors hybridized ACO with a Genetic Algorithm (GA) to solve the Vehicle Routing (VR) problem [57]. Results suggest that the production of the explorer bees is the most relevant factor in the search process of the proposed metaheuristic on solving VR.

Another swarm intelligence used to solve grouping problems is the Firefly Algorithm (FA), which is inspired by the social communication of fireflies via luminescent flashes and their synchronization. In this metaheuristic, a solution is represented by a firefly with a brightness associated that corresponds to its quality, and each firefly attracts its partners proportionally to its brightness. FA starts creating a population of fireflies (solutions) randomly distributed in the search space. Then, for a certain number of iterations, fireflies will move toward other positions, finding potential candidate solutions. For each firefly in the population, its brightness is compared with all other solutions, and its position is updated considering brighter fireflies. To date, FA has been used to tackle fifteen of the twenty-two grouping problems contemplated in this review [14,30,45,59,72,83,94,113,138,161,177,201,214,232,245]. In 2018, Ezugwu and Akutsan presented one of the last related works on solving grouping problems applying FA [214]. In this work, the authors addressed a variant of the Parallel Machine Scheduling (PMS) problem with FA. Experimental results indicate that the performance of the proposed FA is competitive, fast, and efficient for both small and large problem instances.

The last swarm intelligence technique considered in this review is the Grouping Particle Swarm Optimization (GPSO), an extension of the traditional Particle Swarm Optimization that incorporates the grouping representation scheme, as well as a grouping variation operator. Until now, only five of the twenty-two grouping problems contemplated in this review have been addressed using this technique, because it is a relatively new algorithm [110,149,223,229]. However, it has shown promising results. In 2016, Xu proposed one of the last applications of GPSO on solving grouping problems. In this work, the author studied the performances of GPSO to solve ten real instances from the industry and ten more randomly generated of a variant of the Cutting Stock (CS) problem, showing interesting results.

It is important to remark that the state-of-the-art includes other swarm intelligence algorithms used to solve grouping problems, like Biogeography-Based Optimization (BBO), applied to solve Bin Packing and Graph Coloring [270,271]. However, this review considers only the most representative, that is, the six already mentioned. Table 4 shows which swarm intelligence algorithms (rows) have been used to tackle each grouping problem (columns). Cells contain related work references. As can be seen, Ant Colony Optimization and Particle Swarm Optimization are the most popular swarm intelligence algorithms since they have been used to tackle twenty grouping problems. In contrast, only five problems have been addressed employing the Grouping Particle Swarm Optimization since it is a relatively new metaheuristic.

4. Conclusions of literature review and potential directions

As a result of the literature review, seventeen metaheuristics and twenty-two grouping problems were reviewed, comprising four local search strategies, seven evolutionary algorithms, and six swarm intelligence algorithms. Figs. 1 and 2 show a graphical comparison of the literature review summarized in Tables 1–4 Fig. 1 shows a graphical comparison of the number of grouping problems (indicated on the horizontal axis) addressed by each algorithm (depicted in the vertical axis).

From this plot, one can see that Tabu Search (TS) is the local search used to address the greatest number of the grouping problems considered in this review since it has been applied to solve nineteen of the twenty-two problems. In contrast, Hill Climbing (HC) only has been used to address thirteen problems. Regarding swarm intelligence algorithms, Fig. 1 indicates that Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have been the metaheuristics used to solve the greatest number of problems, applied to solve twenty of the twenty-two problems considered. Finally, Fig. 1 indicates that from the seventeen metaheuristics contemplated, the evolutionary algorithms: Genetic Algorithms (GA) and Grouping Genetic Algorithms (GGA) have been used to solve the wider range of the problems, applied to solve twenty-two and twenty problems, respectively.

On the other hand, the detail of the number of algorithms (horizontal axis) used to address each grouping problem (vertical axis) is presented in Fig. 2. From this plot can be seen that some grouping problems have been investigated using a wide range of metaheuristic algorithms, including Job Shop Scheduling (JSS) and Clustering Problem (CP). In contrast, this picture also shows that there are scarce studies to solve problems, like Modular Product Design (MPD), and Multivariate Microaggregation (MM).

In addition to this, Fig. 3 includes the number of results thrown by google scholar about the term “metaheuristic” for each grouping problem listed in Table 1 from 2015 to 2019. This graph suggests that there have been several studies on solving problems like Bin Packing (BP), Load Balancing (LB), Vehicle Routing (VR), Facility Location (FL), and Job Shop Scheduling (JSS) using metaheuristics over the last years, highlining the clear intensification on the study of the Vehicle Routing (VR) problem. In contrast, Fig. 3 also shows the least studied grouping problems, including Multivariate Microaggregation (MM), Modular Product Design (MPD), Maximally Diverse (MD), and Stock Portfolio (SP). Such problems can be seen as an opportunity niche since there are many metaheuristics that have not been used to solve them. So it would be interesting to know their performance addressing them.

The state-of-the-art suggests that problems with a higher occurrence in real-world applications are the most studied ones. For example, sixteen metaheuristics have been used to address the Clustering Problem (CP). This problem has received increasing attention since it occurs in practical problems as off-line and online search engines, voice and data mining, pattern recognition, image processing, bioinformatics, machine learning, and reports analysis [248]. Another grouping problem quite investigated is the Timetabling Problem (TP), which takes place in nurses schedules, sports schedules, transportation schedules, university schedules, among many other applications [248]. Finally, as Fig. 3 indicates, another widely studied grouping problem is Vehicle Routing (VR), which plays a central role in the fields of physical distribution and logistics. Over the last years, this problem has been significantly studied. As a result, nowadays, there are a wide variety of variants of

Table 4
Grouping problems addressed using swarm intelligence metaheuristics.

Metaheuristic	Problem																					
	BP	LB	ALD	VR	CF	MTS	FL	JSS	TF	MPD	MM	MK	CS	TP	CP	MD	SS	GC	PMS	OB	HHC	SP
ACO	[10]	[26]	[42]	[56]	[69]	[80]	[92]	[108]	[120]		[127]	[135]	[147]	[158]	[173]		[183]	[197]	[210]	[220]	[230]	[241]
PSO	[12]	[27]	[44]	[55]	[68]	[81]	[93]	[109]	[119]	[124]		[136]	[141]	[159]	[174]		[188]	[198]	[211]	[221]	[231]	[242]
CS	[13]	[28]	[41]	[58]	[70]		[95]	[111]		[123]			[148]	[160]	[175]		[189]	[199]	[212]			[243]
ABC		[29]	[43]	[57]	[71]	[82]	[96]	[112]	[118]			[137]			[176]	[182]	[187]	[200]	[213]	[222]		[244]
FA	[14]	[30]	[45]	[59]	[72]	[83]	[94]	[113]				[138]		[161]	[177]			[201]	[214]		[232]	[245]
GPSO	[223]							[110]					[149]							[223]	[229]	

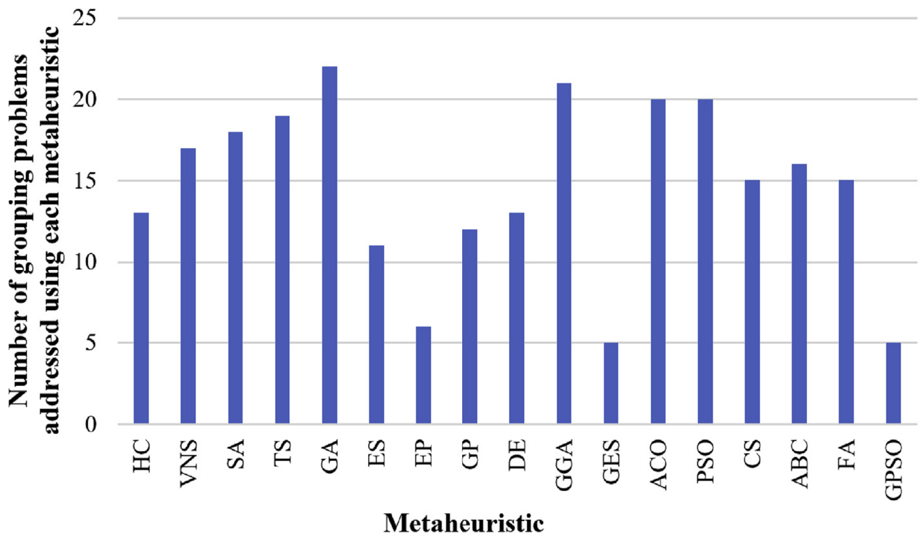


Fig. 1. Comparative graph of the number of grouping problems addressed using each metaheuristic.

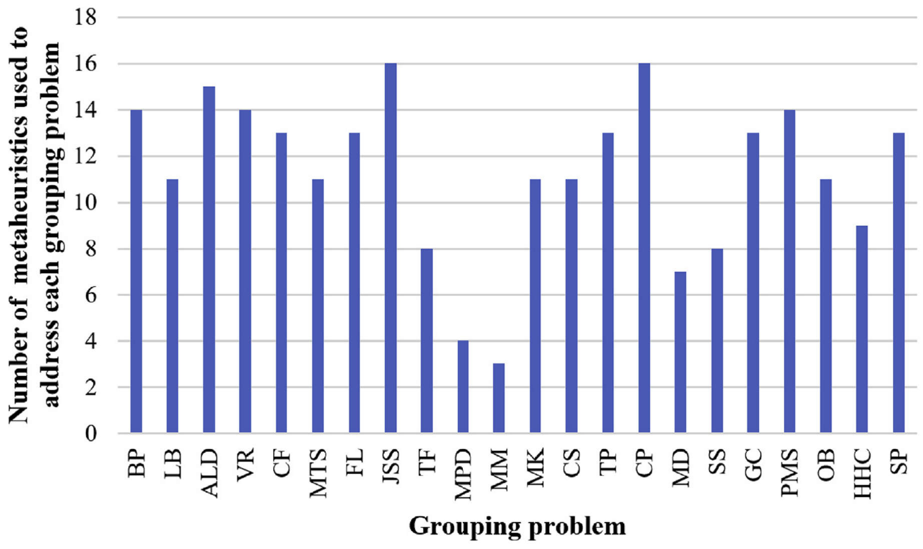


Fig. 2. Comparative graph of the number of metaheuristics used to address each grouping problem.

VR, metaheuristics that solve them, and extensive literature about it [48,51,55].

The literature review also allowed to identify possible directions for the development of more efficient metaheuristic techniques to solve NP-hard grouping problems. Some important issues are discussed below.

a) In recent years, some efforts have been carried out by proposing and adopting new search methods to solve grouping problems; for example, Evolution Strategies (ES) and Evolutionary Programming (EP) in evolutionary computation, as well as the Firefly Algorithm (FA) and Cuckoo Search (CS) regarding the swarm intelligence approach. That is, there exists a trend to explore the algorithmic behavior of new metaheuristics addressing grouping problems. However, none of the metaheuristics in the state-of-the-art has been analyzed to explain the reasons for good or bad behavior in different grouping problems instances.
b) The current trend is to focus on solving real-world problems, mainly those that have a direct impact on society, such is the case of applications related to health. Nevertheless, for most of the real-world problems, it is unclear how to select and integrate the appropriate techniques to solve them and what is the expected performance of different strategies.
c) The group-based representation scheme can be used to improve the performances of other metaheuristics that solve grouping problems besides Genetic Algorithms (GA). But only Particle Swarm Optimization (PSO) and Evolution Strategies (ES) have been adapted to deal with this issue. Not much work has been done on the performance study of these new grouping metaheuristics to different grouping problems.
d) One of the main challenges in the development of high-performance algorithms for grouping problems is the design of efficient strategies that work together with the grouping encoding scheme and the features of the grouping problems instances to find the optimal groups in fewer function evaluations.

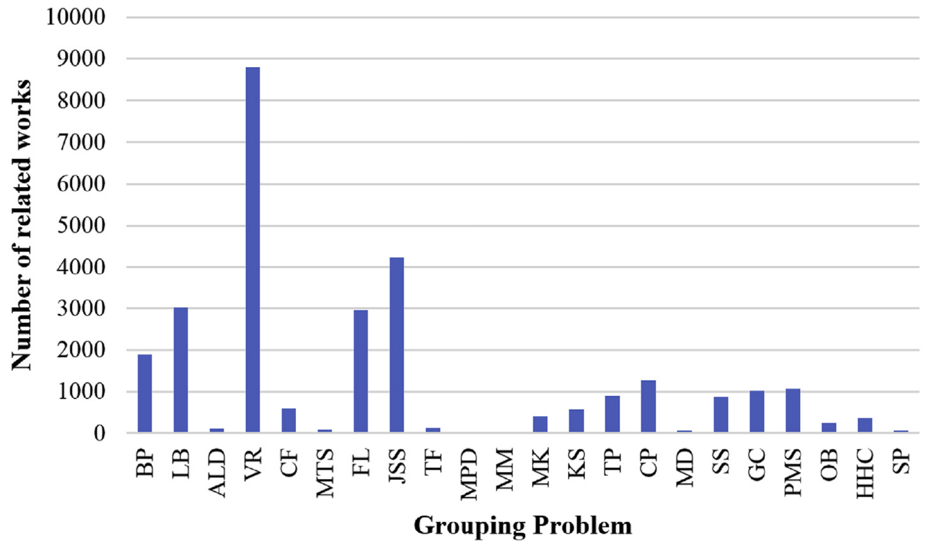


Fig. 3. Results threw by google scholar about the terms "metaheuristic" for each "*grouping problem*" in Table 1.

5. Representation schemes for grouping problems: a case study in swarm intelligence and genetic algorithms for Parallel-Machine Scheduling

Metaheuristics have been successfully applied to solve different NP-hard grouping problems across a broad range of real-world application areas. Among the metaheuristics applied to solve a greater variety of grouping problems are GA, GGA, and PSO. Since, considering the problems contemplated in this review, they have been used to solve twenty-two, twenty-one, and twenty problems, respectively. Some of the best results solving these problems have been obtained combining grouping encoding schemes and search methods adapted to these encodings; the development of grouping metaheuristics is a quick research evolving field. In this section, we illustrate three different approaches to solve a grouping problem.

A comparative study is conducted to evaluate the performance of a Grouping Genetic Algorithm [5] versus a Genetic Algorithms with an extended permutation solution encoding [204] and a Particle Swarm Optimization with a machine-based representation scheme [272] on test problem instances of the Parallel-Machine Scheduling problem with unrelated machines. The experimentation was carried out using 1400 test instances proposed by the specialized literature. Experimental results allow us to validate the conclusions identified during the literature review.

The $R \parallel C_{max}$ Parallel-Machine Scheduling problem is a well-known NP-hard grouping problem. According to the classification presented by Lawler *et al.* in 1982 to organize the Parallel-Machine Scheduling problems [273], $R \parallel C_{max}$ considers unrelated machines, jobs with non-preemptions, and the reduction of the maximum completion time. $R \parallel C_{max}$ consists of scheduling n given jobs $N = j_1, \dots, j_n$ into m available machines $M = i_1, \dots, i_m$, such that the maximum completion time of the jobs (*makespan*) is as small as possible. This problem is known to be NP-hard in the strong sense since it was showed that the special case with identical machines $P \parallel C_{max}$ belongs to that class [2]. Coupled with this, in 1990 Lenstra *et al.* [274] shown that no polynomial time algorithm exists for $R \parallel C_{max}$ with a better worst case ratio approximation than 3/2 unless $P = NP$.

In general, $R \parallel C_{max}$ is established as an assignment problem since the processing job order is not relevant. Hence, $R \parallel C_{max}$ is usually represented as a basic Integer Linear Program (ILP) as follows [275]:

$$\min C_{max} \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in N \tag{2}$$

$$\sum_{j=1}^{n} p_{ij} \cdot x_{ij} \leq C_{max} \quad \forall i \in M \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad \forall j \in N, \forall i \in Mss, \tag{4}$$

where $C_{max} = max(C_i)$, C_i indicates the total completion time for each machine i to process its assigned jobs; p_{ij} is the processing time of job j on machine i; and $x_{ij} = 1$ if the job j is assigned to the machine i, otherwise $x_{ij} = 0$.

In this grouping problems, the quality of every solution is usually calculated by the *makespan* $= max(C_i)$. That is, the quality of a solution is equal to the time required by the machine that finishes processing its assigned jobs at the end.

5.1. Grouping Genetic Algorithm

Recalling from the literature review, the Grouping Genetic Algorithm has become one of the most popular solvers for NP-hard grouping problems. Nevertheless, the Grouping Genetic Algorithm has not been applied to solve the $R \parallel C_{max}$ problem. The aforementioned motivates this study, where the Grouping Genetic Algorithm with Controlled Genes Transmission (GGA-CGT) introduced by Quiroz-Castellanos *et al.* to solve the Bin Packing problem [276], is adapted to address the $R \parallel C_{max}$ Parallel-Machine Scheduling problem. For this study, the heuristic used to generate the initial population in GGA-CGT, as well as the mutation and crossover operators, were adapted to solve problem $R \parallel C_{max}$. Conversely, the remaining mechanisms and operators, as well as the parameter settings were not modified.

In the GGA group-based encoding scheme, the groups are the units or genes of the chromosomes. Hence, in a group-based solution for the $R \parallel C_{max}$ problem each gene represents a machine [277]. That is the solutions chromosomes have a length m (number of machines), and every gene i includes the groups of jobs assigned to it.

Fig. 4 contains an instance of $R \parallel C_{max}$ with five machines and eleven jobs, where each column represents a machine i from 1 to M, each row depicts a job j from 1 to N, and every cell contains the processing time p_{ij} required by machine i to process job j. In addition to this, Fig. 4 holds two solutions encoded using the representation scheme based on groups, one partial and one complete, for this instance of $R \parallel C_{max}$ problem. A complete solution for this problem, using the group-based encoding, is represented with a chromosome with five genes, one for each machine i from i_1 to i_5, where each job j from j_1 to j_{11} is assigned to one the five available machines exactly. Given the above, Fig. 4 contains a complete solution, where machine i_1 contains jobs j_4, j_6, and j_{10}, machine i_2 holds jobs j_3, and j_7, machine i_3 has jobs j_5, j_9, and j_{11}, machine i_4 includes the job j_2, and machine i_5 holds jobs j_1 and j_8. On the other hand, to take advantage of the structure of the solutions, genetic operators will work with the machines and the information about which jobs belong to which machines.

5.1.1. Population initialization

The population initialization strategy of GGA-CGT was modified since it used a heuristic that incorporates knowledge of the Bin Packing (BP) problem domain. In the new population initialization strategy, the jobs are first permuted at random and then scheduled, using the well-known $R \parallel C_{max}$ local heuristic *min*() [278]. In this heuristic, for each job j to be scheduled, the equation $C_i = C_i + p_{ij}$ is used to calculate the value of C_i that is generated for each machine i when assigning each job j, where C_i is the processing time required by machine i to process its assigned jobs, and p_{ij} is the time it needs to process job j. Next, the machine that generates the lowest C_i value is identified, and the job is assigned to it. Fig. 4 contains a potential solution for the $R \parallel C_{max}$ problem. For this example, an instance with five machines and eleven jobs is considered, as well as a list with the eleven jobs permuted randomly. Thus, the partial solution was made up scheduling the first ten permuted jobs with the heuristic *min*(); while the final solution indicates in which machine the last job in the permuted list (i.e., j_{11}) must be scheduled. Therefore, following the already mentioned *min*() procedure, the processing time C_i of each machine plus the time it requires to process job $p_{ij_{11}}$ results in the following way: $C_1 = 100 + 40, C_2 = 120 + 20, C_3 = 80 + 50, C_4 = 150 + 10$, and $C_5 = 100 + 50$. Hence, j_{11} is assigned to machine i_3 since it generated the lowest C_i value.

5.1.2. Gene-level crossover

The Gene-Level Crossover Operator proposed by Quiroz-Castellanos *et al.* to solve the bin packing problem [18] was slightly adapted to work with the $R \parallel C_{max}$ problem. Given two parent solutions (p_1 and p_2), two children solutions (c_1 and c_2) are generated as follows. The crossover process begins generating two ordered lists to determine the genes transmission sequence for each parent solution (l_1 for c_1 and l_2 for c_2). Such lists are ordered considering the C_i values of each machine in increasing order. That is, from the machine that finishes processing its jobs first to the machine that ends at last. Next, l_1 and l_2 are used to control the genes (machines) transmission. Consequently, the machines that process their jobs faster are transmitted first and the slowest ones until the end. Hence, c_1 starts receiving the faster machine from p_1, then the faster machine from p_2, and so on. Both parents can transmit every machine to each child; Nevertheless, children only receive the machine of the parent that appears first in l_1 or l_2, according to the case.

Frequently, the genes transmission process ends with infeasible children solutions. Some of the jobs appear twice in the solution (assigned to two machines), and must be eliminated from one of them. We eliminate the duplicated jobs from the machines with the higher C_i value. On the other hand, some jobs can be missing from the solution; these unassigned jobs are scheduled using the well-known scheduling heuristic *min*() [278]. Algorithm 1 contains the detail of the Gene-Level Crossover Operator.

Algorithm 1 Gene-Level Crossover Operator

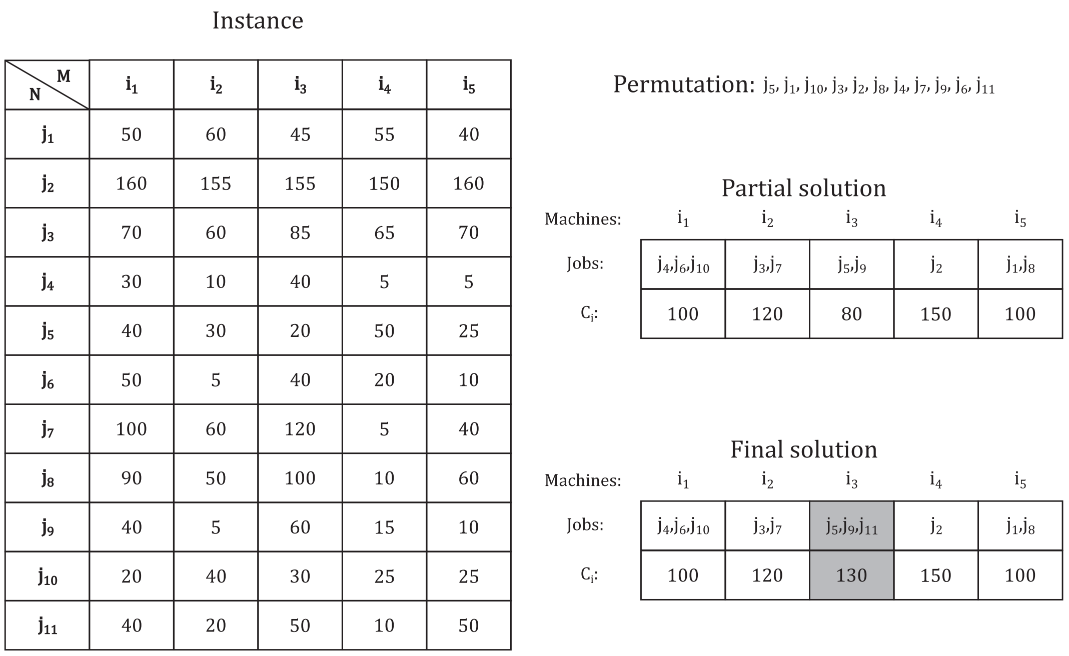
```
Input: Two parents $p_1$ and $p_2$.
Output: Two children $c_1$ and $c_2$.
1: Generate an increasing order list $l_1$ with the machines of $p_1$ based on $C_i$.
2: Generate an increasing order list $l_2$ with the machines of $p_2$ based on $C_i$.
3: for $i$ in range $(1, \ldots, m)$ do
4:    if machine $l_{1_i} \notin c_1$ then
5:       Transmit machine in position $i$ of $l_1$ to $c_1$
6:    end if
7:    if machine $l_{2_i} \notin c_1$ then
8:       Assign machine in position $i$ of $l_2$ to $c_1$
9:    end if
10: end for
11: for $i$ in range $(1, \ldots, m)$ do
12:    if machine $l_{2_i} \notin c_2$ then
13:       Transmit machine in position $i$ of $l_2$ to $c_2$
14:    end if
15:    if machine $l_{1_i} \notin c_2$ then
16:       Assign machine in position $i$ of $l_1$ to $c_2$
17:    end if
18: end for
19: Remove repeated jobs.
20: Schedule unassigned jobs with the heuristic min().
```

5.1.3. Download mutation operator

For this case study, the download mutation operator was used to replace the original GGA-CGT mutation operator. In general, this operator modifies two genes in a solution using two phases (download and reassign). First, genes are divided to get two sets (W and O), such that W contains the machines with $C_i = C_{max}$, and O includes the machines with $C_i < C_{max}$; then, one machine is selected at random from each set (W and O) and their jobs are downloaded. Second, a new solution is obtained by redistributing the unassigned jobs in these machines (W and O) with the heuristic *min*(). Algorithm 2 contains the mutation process of the download operator.

Algorithm 2 Download mutation operator

```
Input: A solution $X$ with $m$ machines $i_1, \ldots, i_m$.
Output: A mutated solution of $X$.
1: Calculate the processing time $S_i$ of each machine $i$.
2: Generate the set $W$ using the machines with
   $S_i = C_{max}$ and $O$ with machines where $S_i \neq C_{max}$.
3: Randomly select the $w$ machine from $W$ and $o$ from
   $O$, such that $h \neq r$.
4: Release all the jobs in the selected machines $h$ and $r$.
5: Create the list $RJ$ with a permutation of the released
   jobs.
6: for job $j$ in $RJ$ do
   7: if $S_w + p_{wj} < S_o + p_{oj}$ then
      8: schedule job $j$ in machine $w$
   9: else
      10: schedule job $j$ in machine $o$
   11: end if
12: end for
```



Instance

N \ M	i_1	i_2	i_3	i_4	i_5
j_1	50	60	45	55	40
j_2	160	155	155	150	160
j_3	70	60	85	65	70
j_4	30	10	40	5	5
j_5	40	30	20	50	25
j_6	50	5	40	20	10
j_7	100	60	120	5	40
j_8	90	50	100	10	60
j_9	40	5	60	15	10
j_{10}	20	40	30	25	25
j_{11}	40	20	50	10	50

Permutation: $j_5, j_1, j_{10}, j_3, j_2, j_8, j_4, j_7, j_9, j_6, j_{11}$

Partial solution

Machines:	i_1	i_2	i_3	i_4	i_5
Jobs:	j_4,j_6,j_{10}	j_3,j_7	j_5,j_9	j_2	j_1,j_8
C_i:	100	120	80	150	100

Final solution

Machines:	i_1	i_2	i_3	i_4	i_5
Jobs:	j_4,j_6,j_{10}	j_3,j_7	j_5,j_9,j_{11}	j_2	j_1,j_8
C_i:	100	120	130	150	100

Fig. 4. Potential solution for the $R \parallel C_{max}$ problem generated with the population initialization strategy of GGA-CGT.

5.2. *Genetic algorithm with extended permutation solution encoding*

One of the last proposals to address the $R \parallel C_{max}$ problem was introduced by Sels *et al.* in 2015 [204]. In such work, the authors introduced a Genetic Algorithm adopting an extended permutation representation with job symbols and partitioning symbols, a commonly used representation for Parallel-Machine Scheduling problems [247,249]. Solutions encoded with this representation scheme have a length of $n + m - 1$, where n and m indicate the number of jobs and machines, respectively. Thereby, each gene is used to store a symbol or a job, such that each symbol indicates where each machine i starts and ends. Given the above, Fig. 5 contains a solution encoded using this extended permutation representation for a $R \parallel C_{max}$ problem with five machines and eleven jobs, where machine i_1 contains jobs j_4, j_6, and j_{10}, machine i_2 holds jobs j_3, and j_7, machine i_3 has jobs j_5, j_9, and j_{11}, machine i_4 includes the job j_2, and machine i_5 holds jobs j_1 and j_8.

In order to develop a Genetic Algorithm that works as well as possible, Sels *et al.* analyzed the performance of three selection methods (i.e., tournament selection, roulette wheel selection, and ranking selection), five crossover operators (i.e., partially mapped crossover, order crossover, position based crossover, order based crossover, and cycle crossover) and three mutation operators (i.e., swap mutation, insertion mutation, and inversion mutation). In such study, they performed every possible combination using a randomly generated population and concluded that the GA configuration with the tournament selection, order crossover, and the swap mutation operators produced the best performance.

Fig. 6 details the genetic material transmission process of the order crossover operator. As Fig. 6 indicates, this operator uses two parent solutions (p_1 and p_2) to generate one child (c) as follows. First, two cutting points (cs_1 and cs_2) are selected randomly in the range (1, $n + m - 1$). Then, the jobs and symbols between the two selected cutting points (cs_1 and cs_2) in p_1 are copied to c, preserving their positions, and subsequently deleted from p_2. Finally, the remaining jobs and symbols are inserted into c, respecting the order they appear in p_2.

On the other hand, solutions generated with the order crossover operator are later mutated with the swap mutation, to ensure some diversity in the population. Fig. 7 describes the mutation process. As Fig. 7 shows, the mutation operator exchanges the positions of two randomly selected genes (g_1 and g_2) from every solution; these genes can be two jobs within different machines, two jobs within one machine, two symbols, or a job and a symbol.

As the last part of the variation process, all the new solutions are improved using a local search, based on the variable neighborhood descent method introduced by Fanjul-Peyro and Ruiz in 2010 [275]. Additionally, before ending each generation, the solutions are evaluated and inserted back into the population by using an incremental replacement approach. In this way, the solutions with the worst *makespan* are replaced by the new solutions, while the best ones remain intact to the next generation. The results reported by Sels *et al.* [275], over 1400 instances, are used to compare the performance of this GA with our Grouping Genetic Algorithm and a Particle Swarm Optimization of the state-of-the-art.

5.3. *Particle Swarm Optimization with machine-based encoding*

Recalling from the literature review, the Particle Swarm Optimization (PSO) is one of the most recurrent swarm intelligence algorithms used to solve NP-hard grouping problems. Nevertheless, according to the scope of this review, it has not been used to solve the $R \parallel C_{max}$ problem. The aforementioned motivates this study, where the PSO introduced by Niu *et al.* to solve the Parallel-Machine Scheduling problem with identical machines and total tardiness minimization [272] is applied to $R \parallel C_{max}$. The PSO proposed by Niu *et al.* works as follows. First, it uses a machine-based encoding. In this representation, chromosomes have length n (number of jobs), each gene represents a job, and every gene stores the machine processing each job. Given the above, Fig. 8 contains a potential solution represented using the machine-based encoding for a $R \parallel C_{max}$ problem with five machines and eleven jobs, where machine i_1 contains jobs j_4, j_6, and j_{10}, machine i_2 holds jobs j_3, and j_7, machine i_3 has jobs j_5, j_9, and j_{11}, machine i_4 includes the job j_2, and machine i_5 holds jobs j_1 and j_8. Considering this representation, PSO creates a solution as follows: for each job, a real random value i between 1 and $m + 1$ (where m is the number of machines) is generated, to later rounding it with the floor function, and store it in the corresponding gene. Thanks to this encoding, PSO can work with traditional PSO operator, presented in Eqs. (5) and (6).

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times rand() \times (x_{gd}^{t} - x_{id}^{t}) + c_2 \times rand() \times (x_{pd}^{t} - x_{id}^{t}) \quad (5)$$

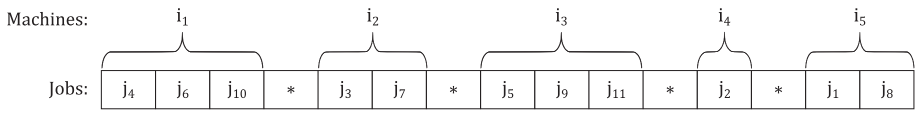


Fig. 5. Extended permutation encoding for $R \parallel C_{max}$ problem.

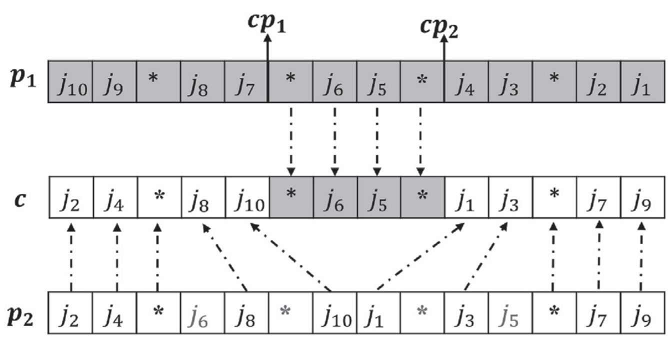

cp1
cp2
p1
j10 j9 * j8 j7 * j6 j5 * j4 j3 * j2 j1
c
j2 j4 * j8 j10 * j6 j5 * j1 j3 * j7 j9
p2
j2 j4 * j6 j8 * j10 j1 * j3 j5 * j7 j9

Fig. 6. Order crossover operator with extended permutation representation for $R \parallel C_{max}$ problem.

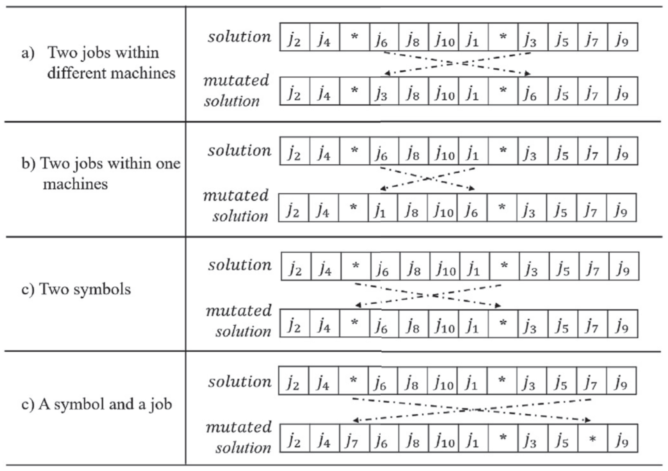


Fig. 7. Swap mutation with extended permutation representation for $R \parallel C_{max}$ problem.

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{6}$$

where v_{id}^{t+1} represents the velocity of the particle i, for next iteration $t+1$, and for vector dimension d; w is the inertia weight, used to balance the search space exploration and exploitation; rand() is a real value randomly generated using a uniform distribution over [0, 1]; and c_1 and c_2 are acceleration constants, which represent the weighting of stochastic acceleration terms that pull each particle toward the personal best x_{pd}^{t} (the best position of each particle during the whole search) and the global best x_{gd}^{t} (the position of the best particle in the current iteration); Finally, x_{id}^{t} and x_{id}^{t+1} depict the current position of particle i and its position for next iteration, respectively.

During the application of the flight operator, infeasible values can be generated. Therefore, a repairing mechanism is used to transform them into feasible ones. In this mechanism, if values are generated bellow 1, they are transformed into 1. On the other hand, if values are generated above M, they are transformed into M.

Niu *etal.* analyzed mainly three parameters of PSO: the inertia weight w and the acceleration constants c_1 and c_2. They test the following values: $w = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ and $c_1 = c_2 = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. They conclude that the parameters providing to PSO the best performance are $w = 0.2$ and $c_1 = c_2 = 2$; therefore, we use that configuration in this work.

5.4. Computational experiments

The performance of GGA-CGT, GA and PSO was measured using 1400 test instances employed by Fanjul-Peyro in 2010 [275]. This benchmark is divided into seven sets with 200 instances, such that each collection includes cases with 100, 200, 500 and 1000 jobs; as well as 10, 20, 30, 40 and 50 machines. These instances groups differ in the strategy used to generate the p_{ij} values. In this way, the benchmark includes the sets $U(1,100)$, $U(10,100)$, $U(100,120)$, $U(100,200)$, and $U(1000,1100)$ that were created using an uniform distribution in the ranges $(1,100)$, $(10,100)$, $(100,120)$, $(100,200)$, $(1000,1100)$. On the other hand, the sets *MacCorr* and *JobsCorr* were formed considering a correlation among the machines and the jobs, respectively. In addition to this, this benchmark contains the *makespan* found by 2 h of the commercial CPLEX algorithm, which is used to measure the Relative Percentage Deviation (*RPD*) from GGA-CGT and both GAs with the extended permutation representation to CPLEX.

Subsequently, GGA-CGT, GA, and PSO were compared using the *RPD* presented in Eq. (7), where $C_{max}(i)$ indicate the C_{max} value found by the assessed algorithm for instance i and $C^*_{max}(i)$ depicts the best C_{max} found using 2 h of the CPLEX algorithm for the same instance.

$$RPD = \frac{C_{max}(i) - C^*_{max}(i)}{C^*_{max}(i)} \tag{7}$$

Table 5 shows a comparison between the GGA-CGT, GA, and PSO based on *RPD*. Hence, each cell contains the *RPD* average from each algorithm to CPLEX for the 1400 instances. Furthermore, for a detailed study, instances were grouped using as criterion the number of jobs n, the number of machines m, and the distribution of the processing times p_{ij}.

From Table 5 can be observed that GGA-CGT had a lower average of *RPD* than GA and PSO solving the 1400 instances. Similar, it can be noted that GGA-CGT had a lower average of *RPD* when grouping the instances according to the number of jobs, the number of machines,

Table 5
GGA-CGT, GA, and PSO comparison using RPD.

	Instance set	GGA-CGT	GA	PSO
n	100	**0.03**	1.30	0.54
	200	**0.05**	1.22	0.53
	500	**0.06**	0.80	0.50
	1000	**0.06**	0.54	0.48
m	10	**0.05**	0.33	0.40
	20	**0.06**	0.76	0.50
	30	**0.05**	0.91	0.53
	40	**0.06**	1.33	0.57
	50	**0.06**	1.50	0.61
P_{ij}	$U(1,100)$	**0.07**	2.17	0.91
	$U(10,100)$	**0.10**	1.71	0.80
	$U(100,120)$	**0.02**	0.16	0.28
	$U(100,200)$	**0.07**	0.61	0.45
	$U(1000,1100)$	**0.01**	0.05	0.25
	MacsCorr	**0.08**	1.81	0.35
	JobsCorr	**0.05**	0.25	0.63
	Average	**0.06**	0.97	0.53

Jobs	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}
Machines	i_5	i_4	i_2	i_1	i_3	i_1	i_2	i_5	i_3	i_1	i_3

Fig. 8. Machine-based representation scheme for $R \parallel C_{max}$ problem.

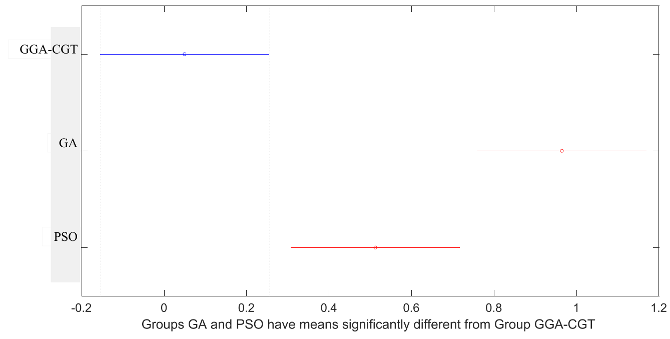


Fig. 9. Means plot and Tukey post hoc test for all tested algorithms considering the number of jobs.

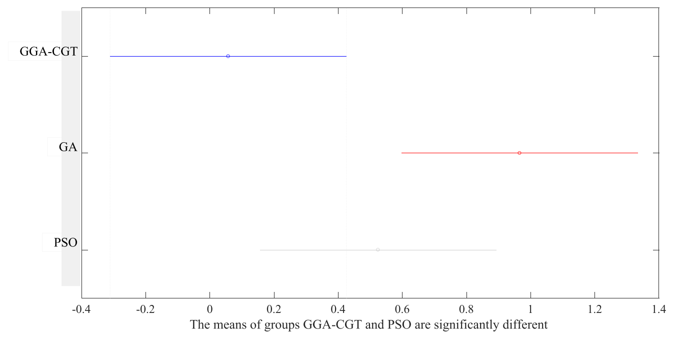


Fig. 11. Means plot and Tukey post hoc test for all tested algorithms considering the distribution of the processing times.

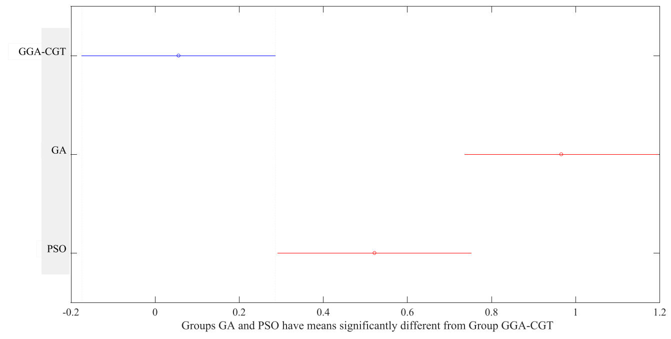


Fig. 10. Means plot and Tukey post hoc test for all tested algorithms considering the number of machines.

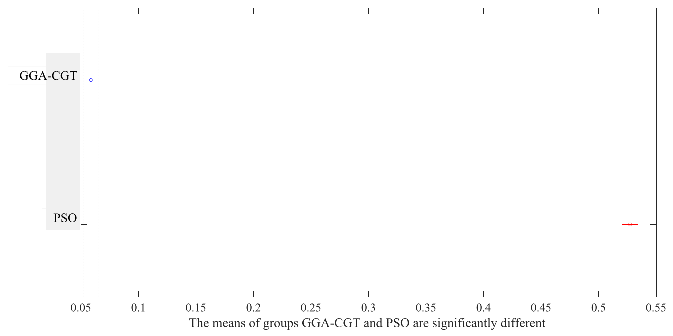


Fig. 12. Means plot and Tukey post hoc test for GGA-CGT and PSO for all test instances.

and the distribution of the jobs.

In addition, the single factor Analysis of Variance (ANOVA) and the post hoc Tukey test were applied to a further comparison of the performance of the solution methods. The Relative Percentage Deviations of the algorithms were analyzed considering four different approaches, looking for the existence of significant differences in the solutions. Figs. 9–11 show the means plot with Tukey intervals with a 95% confidence level for the first three comparisons, including the performance of the three algorithms in the 1400 instances grouped according to the number of jobs, the number of machines, and the distribution of the processing times, respectively. These studies were performed using the average RPD reached by each algorithm, presented in Table 5. Since, for GA, only those data related to its performance are known. On the other hand, in the last comparison, RPD values reached by GGA-CGT and PSO for each of the 1400 instances were contemplated. Fig. 12 shows the plot with Tukey intervals with a 95% confidence level for these algorithms.

As can be seen in Figs. 9 and 10, GGA-CGT is indeed statistically better than GA and PSO considering the average RPD they reached for the instances grouped according to the number of jobs and machines, respectively. On the other hand, Fig. 11 indicates that GGA-CGT is statistically better than GA, but not better than PSO, considering the average RPD they reached for the instances grouped according to the distribution of the processing time. Finally, Fig. 12 shows that GGA-CGT is statistically better than PSO, considering the DPR they reach for each of the 1400 instances.

5.5. *Conclusions of the case study*

As part of the case study, the GGA-CGT was adapted to solve the $R \parallel C_{max}$ Parallel-Machine Scheduling variant that considers unrelated machines, jobs with non-preemptions, and *makespan* reduction. Said metaheuristic was designed based on the trends identified during the literature review. That is, using a more natural or intuitive way to represent the potential solutions to the $R \parallel C_{max}$ problem (group-based) as well as variation operators designed to work together with the grouping encoding efficiently. Experimental results suggest that the Genetic Algorithm performance in the $R \parallel C_{max}$ problem is related to its solutions encoding as well as its variation operators. Hence, they keep the already found trends. That is, solvers performance to tackle grouping problems can be improved incorporating group-based representation schemes and suitable variation operators. On the other hand, the case study also revealed that PSO with machine-based encoding outperformed in most cases GA with the extended permutation. Future work involves the parameter tuning of GGA-CGT to study its reach in the $R \parallel C_{max}$, as well as the analysis of the effect of each of its components during the search process, to identify the best strategies that work together with the grouping encoding scheme and the features of the problem. On the other hand, regarding PSO, paths of work are related to design a PSO algorithm with the group-based representation and efficient variation operators to solve $R \parallel C_{max}$.

6. General conclusions

This paper presents a review of the metaheuristics developed in recent years for the solution of grouping problems, contributing to give some more structure and guidance to this line of research. An important contribution is the case study for the Parallel-Machine Scheduling problem, examining the most common representation schemes and the best state-of-the-art algorithms performance. We provide a survey dividing this growing research area into neighborhood searches, swarm intelligence techniques, and evolutionary algorithms. Besides, three representative metaheuristics were outlined in more detail, analyzing their performance. The literature review suggests that evolutionary algorithms have been the most helpful approaches, highlighting the use of Genetic Algorithms (GA). On the other hand, Tabu Search (TS) and Ant Colony Optimization (ACO) are the local search and swarm intelligence more studied, respectively. The state-of-the-art also suggests that Vehicle Routing (VR), Job Shop Scheduling (JSS), and Clustering Problem (CP) are the most studied grouping problems, whereas little attention has been given to problems like Modular Product Design (MPD) and Multivariate Microaggregation (MM). This review reveals that some metaheuristic like Grouping Evolution Strategy (GES) and Grouping Particle Swarm Optimization (GPSO) have been scarcely studied since they are recent, however, they have shown interesting results. An increasing tendency was identified to design metaheuristic algorithms with representation schemes based on groups, to date, the state-of-the-art contains grouping metaheuristics like Grouping Particle Swarm Optimization, Grouping Genetic Algorithms, and Grouping Evolution Strategies. Concerning the case study, three population heuristics were assessed solving the $R \parallel C_{max}$ Parallel-Machine Scheduling problem. The Particle Swarm Optimization (PSO) with the machine-based encoding, as well as two Genetic Algorithms (GAs) with different representation schemes: group-based and extended permutation. Experimental results suggest that the performances of GAs can be improved using a more natural way to encode solutions; that is, using the representation scheme based on groups and suitable variation operators. Furthermore, this study suggests that a GA with the group-based representation can reach even better results than PSO with the machine-based encoding. Finally, the literature review also allowed us to identify that there is a lack of a clear mapping between grouping problem features and best-suited techniques. That is, more studies should be devoted to analyzing the relationship between features of grouping problems and the performance of the algorithms used to solve them.

Acknowledgment

O. Ramos-Figueroa acknowledges support from CONACyT and the Universidad Veracruzana to pursue graduate studies at the Artificial Intelligence Research Center. O. Schütze acknowledges support from Conacyt Basic Science project No. 285599 and SEP Cinvestav project No. 231.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2019.100643.

References

[1] T. Stützle, Local Search Algorithms for Combinatorial Problems-Analysis, Algorithms and New Applications, DISKI-Dissertationen zur Künstlichen Intelligenz, Infix, Sankt Augustin, Germany.
[2] D. S. Johnson, M. R. Garey, A guide to the theory of np-completeness, Computers and Intractability.
[3] E.G. Talbi, Metaheuristics: from Design to Implementation, vol. 74, John Wiley & Sons, 2009.
[4] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[5] E. Falkenauer, A new representation and operators for genetic algorithms applied to grouping problems, Evol. Comput. 2 (2) (1994) 123–144.
[6] A. Martinez-Sykora, R. Alvarez-Valdés, J.A. Bennell, R. Ruiz, J.M. Tamarit, Matheuristics for the irregular bin packing problem with free rotations, Eur. J. Oper. Res. 258 (2) (2017) 440–455.
[7] J.L. Viegas, S.M. Vieira, E.M. Henriques, J.M. Sousa, A tabu search algorithm for the 3d bin packing problem in the steel industry, in: CONTROLO'2014–Proceedings of the 11th Portuguese Conference on Automatic Control, Springer, 2015, pp. 355–364.
[8] T. Kämpke, Simulated annealing: use of a new tool in bin packing, Ann. Oper. Res. 16 (1) (1988) 327–332.
[9] L.F.M. Santos, R.S. Iwayama, L.B. Cavalcanti, L.M. Turi, F.E. de Souza Morais, G. Mormilho, C.B. Cunha, A variable neighborhood search algorithm for the bin packing problem with compatible categories, Expert Syst. Appl. 124 (2019) 209–225.
[10] T.D. Lin, C.C. Hsu, L.F. Hsu, Optimization by ant colony hybrid local search for online class constrained bin packing problem, in: Applied Mechanics and Materials, vol. 311, Trans Tech Publ, 2013, pp. 123–128.
[11] T. Bayraktar, M.E. Aydin, M. Dugenci, A memory-integrated artificial bee algorithm for 1-d bin packing problems, in: Proc. CIE IMSS, 2014, pp. 1023–1034.
[12] A. Laurent, N. Klement, Bin Packing Problem with Priorities and Incompatibilities Using Pso: Application in a Health Care Community, 2019.
[13] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, S. Mirjalili, An improved nature inspired meta-heuristic algorithm for 1-d bin packing problems, Personal Ubiquitous Comput. 22 (5–6) (2018) 1117–1132.
[14] C. Zhao, L. Jiang, K.L. Teo, A hybrid chaos firefly algorithm for three-dimensional irregular packing problem, J. Ind. Manag. Optim. (2018) 147–157.
[15] K. Sim, E. Hart, Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, ACM, 2013, pp. 1549–1556.
[16] J.C. Gomez, H. Terashima-Marín, Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems, Genet. Program. Evolvable Mach. 19 (1–2) (2018) 151–181.
[17] S. Laabadi, M. Naimi, H. El Amri, B. Achchab, A crow search-based genetic algorithm for solving two-dimensional bin packing problem, in: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz), Springer, 2019, pp. 203–215.
[18] T. Kucukyilmaz, H.E. Kiziloz, Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem, Comput. Ind. Eng. 125 (2018) 157–170.
[19] A.H. Kashan, A.A. Akbari, B. Ostadi, Grouping evolution strategies: an effective approach for grouping problems, Appl. Math. Model. 39 (9) (2015) 2703–2720.
[20] B. Mondal, K. Dasgupta, P. Dutta, Load balancing in cloud computing using stochastic hill climbing-a soft computing approach, Procedia Technology 4 (2012) 783–789.
[21] I. Davydov, Y. Kochetov, Vns-based heuristic with an exponential neighborhood for the server load balancing problem, Electron. Notes Discrete Math. 47 (2015) 53–60.
[22] X. Yingzhuo, G.Q. Yang, Research on network load balancing method based on simulated annealing algorithm and genetic algorithm, in: Journal of Physics: Conference Series, vol. 1237, IOP Publishing, 2019, p. 022137.
[23] N. Téllez, M. Jimeno, A. Salazar, E. Nino-Ruiz, A tabu search method for load balancing in fog computing, Int. Artif. Intell 16 (2) (2018) 1–31.
[24] O. Homaee, A. Najafi, M. Dehghanian, M. Attar, H. Falaghi, A practical approach for distribution network load balancing by optimal re-phasing of single phase customers using discrete genetic algorithm, International Transactions on Electrical Energy Systems 29 (5) (2019) e2834.
[25] K. Ray, S. Bose, N. Mukherjee, A load balancing approach to resource provisioning in cloud infrastructure with a grouping genetic algorithm, in: 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), IEEE, 2018, pp. 1–6.
[26] A. Selvakumar, G. Gunasekaran, A novel approach of load balancing and task scheduling using ant colony optimization algorithm, Int. J. Softw. Innov. 7 (2) (2019) 9–20.
[27] N. Sethi, S. Singh, G. Singh, Improved mutation-based particle swarm optimization for load balancing in cloud data centers, in: Harmony Search and Nature Inspired Optimization Algorithms, Springer, 2019, pp. 939–947.
[28] D. Garg, P. Kumar, Evaluation and improvement of load balancing using proposed cuckoo search in cloudsim, in: International Conference on Advanced Informatics for Computing Research, Springer, 2019, pp. 343–358.
[29] L. Shen, J. Li, Y. Wu, Z. Tang, Y. Wang, Optimization of artificial bee colony algorithm based load balancing in smart grid cloud, in: 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia), IEEE, 2019, pp. 1131–1134.
[30] G. Kaur, K. Kaur, An adaptive firefly algorithm for load balancing in cloud computing, in: Proceedings of Sixth International Conference on Soft Computing for Problem Solving, Springer, 2017, pp. 63–72.
[31] M. Yang, L. Ba, Y. Liu, H. Zheng, J. Yan, X. Gao, J. Xiao, An improved genetic simulated annealing algorithm for stochastic two-sided assembly line balancing problem, Int simul model 18 (2019) 175–186.
[32] K. Buyukozkan, I. Kucukkoc, S.I. Satoglu, D.Z. Zhang, Lexicographic bottleneck mixed-model assembly line balancing problem: artificial bee colony and tabu search approaches with optimised parameters, Expert Syst. Appl. 50 (2016) 151–166.

[33] B. Yuan, C. Zhang, X. Shao, A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints, J. Intell. Manuf. 26 (1) (2015) 159–168.
[34] M. Fathi, A. Nourmohammadi, A. H. Ng, A. Syberfeldt, H. Eskandari, An improved genetic algorithm with variable neighborhood search to solve the assembly line balancing problem, Eng. Comput..
[35] A. Yoosefelahi, M. Aminnayeri, H. Mosadegh, H.D. Ardakani, Type ii robotic assembly line balancing problem: an evolution strategies algorithm for a multi-objective model, J. Manuf. Syst. 31 (2) (2012) 139–151.
[36] M.G. Nejad, A.H. Kashan, S.M. Shavarani, A novel competitive hybrid approach based on grouping evolution strategy algorithm for solving u-shaped assembly line balancing problems, Prod. Eng. 12 (5) (2018) 555–566.
[37] A. Baykasoglu, L. Ozbakir, Discovering task assignment rules for assembly line balancing via genetic programming, Int. J. Adv. Manuf. Technol. 76 (1–4) (2015) 417–434.
[38] P. Sresracoo, N. Kriengkorakot, P. Kriengkorakot, K. Chantarasamai, U-shaped assembly line balancing by using differential evolution algorithm, Math. Comput. Appl. 23 (4) (2018) 79.
[39] H.-y. Zhang, An immune genetic algorithm for simple assembly line balancing problem of type 1, Assemb. Autom. 39 (1) (2019) 113–123.
[40] M. Şahin, T. Kellegöz, An efficient grouping genetic algorithm for u-shaped assembly line balancing problems with maximizing production rate, Memetic Computing 9 (3) (2017) 213–229.
[41] Z. Li, N. Dey, A.S. Ashour, Q. Tang, Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem, Neural Comput. Appl. 30 (9) (2018) 2685–2696.
[42] X. Duan, B. Wu, Y. Hu, J. Liu, J. Xiong, An improved artificial bee colony algorithm with maxtf heuristic rule for two-sided assembly line balancing problem, Front. Mech. Eng. 14 (2) (2019) 241–253.
[43] J. Xiong, X. Duan, E. Wang, A hybrid artificial bee colony algorithm for balancing two-sided assembly line with assignment constraints, in: Journal of Physics: Conference Series, vol. 1303, IOP Publishing, 2019, p. 012145.
[44] E.K. Aydoğan, Y. Delice, U. Özcan, C. Gencer, Ö. Bali, Balancing stochastic u-lines using particle swarm optimization, J. Intell. Manuf. 30 (1) (2019) 97–111.
[45] L. Zhu, Z. Zhang, Y. Wang, A pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation, Int. J. Prod. Res. 56 (24) (2018) 7354–7374.
[46] K. Karagul, Y. Sahin, E. Aydemir, A. Oral, A simulated annealing algorithm based solution method for a green vehicle routing problem with fuel consumption, in: Lean and Green Supply Chain Management, Springer, 2019, pp. 161–187.
[47] F.Y. Vincent, S.-W. Lin, Multi-start simulated annealing heuristic for the location routing problem with simultaneous pickup and delivery, Appl. Soft Comput. 24 (2014) 284–290.
[48] J. Lu, L. Wang, A bi-strategy based optimization algorithm for the dynamic capacitated electric vehicle routing problem, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 646–653.
[49] D. Schermer, M. Moeini, O. Wendt, A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations, Comput. Oper. Res. 109 (2019) 134–158.
[50] A. Sadok, J. Teghem, H. Chabchoub, A hybrid grouping genetic algorithm for the inventory routing problem with multi-tours of the vehicle, International Journal of Combinatorial Optimization Problems and Informatics 1 (2) (2010) 42–61.
[51] A. Baniamerian, M. Bashiri, R. Tavakkoli-Moghaddam, Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking, Appl. Soft Comput. 75 (2019) 441–460.
[52] L. Feng, Y.-S. Ong, C. Chen, X. Chen, Conceptual modeling of evolvable local searches in memetic algorithms using linear genetic programming: a case study on capacitated vehicle routing problem, Soft Computing 20 (9) (2016) 3745–3769.
[53] D. Mester, An evolutionary strategies algorithm for large scale vehicle routing problem with capacitate and time windows restrictions, in: Proceedings of the Conference on Mathematical and Population Genetics, University of Haifa, Israel, 2002.
[54] S. Kunnapapdeelert, R. Klinsrisuk, Determination of green vehicle routing problem via differential evolution, Int. J. Logist. Syst. Manag. 34 (3) (2019) 395–410.
[55] Y. Marinakis, M. Marinaki, A. Migdalas, A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows, Inf. Sci. 481 (2019) 311–329.
[56] E.B. Tirkolaee, M. Alinaghian, A.A.R. Hosseinabadi, M.B. Sasi, A.K. Sangaiah, An improved ant colony optimization for the multi-trip capacitated arc routing problem, Comput. Electr. Eng. 77 (2019) 457–470.
[57] M. Davoodi, M.M. Golsefidi, M. Mesgari, A hybrid optimization method for vehicle routing problem using artificial bee colony and genetic algorithm, the International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 42 (2019) 293–297.
[58] J.H. Santillan, S. Tapucar, C. Manliguez, V. Calag, Cuckoo search via lévy flights for the capacitated vehicle routing problem, Journal of Industrial Engineering International 14 (2) (2018) 293–304.
[59] A.M. Altabeeb, A.M. Mohsen, A. Ghallab, An improved hybrid firefly algorithm for capacitated vehicle routing problem, Appl. Soft Comput. 84 (2019) 105728.
[60] R. Kamalakannan, R.S. Pandian, P. Sivakumar, A simulated annealing for the cell formation problem with ratio level data, Int. J. Enterp Netw. Manag. 10 (1) (2019) 78–90.
[61] F.G. Tari, K. Ahadi, Cellular layout design using tabu search, a case study, RAIRO Oper. Res. 53 (5) (2019) 1475–1488.
[62] I.C. Martins, R.G. Pinheiro, F. Protti, L.S. Ochi, A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem, Expert Syst. Appl. 42 (22) (2015) 8947–8955.
[63] A. Stawowy, Evolutionary strategy for manufacturing cell design, Omega 34 (1) (2006) 1–18.
[64] E. Vin, A. Delchambre, Generalized cell formation: iterative versus simultaneous resolution with grouping genetic algorithm, J. Intell. Manuf. 25 (5) (2014) 1113–1124.
[65] R. Branco, C. Rocha, Group technology: hybrid genetic algorithm with greedy formation and a local search cluster technique in the solution of manufacturing cell formation problems, in: Book of Abstracts of the 25th International Joint Conference on Industrial Engineering and Operations, 2019, p. 21.
[66] G.A. Suer, Evolutionary programming for designing manufacturing cells, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE, 1997, pp. 379–384.
[67] C. Dimopoulos, A genetic programming methodology for the solution of the multiobjective cell-formation problem, in: Proceedings of the 8th Joint Conference in Information Systems, JCIS 2005), 2005, pp. 1487–1494.
[68] V. Mahmoodian, A. Jabbarzadeh, H. Rezazadeh, F. Barzinpour, A novel intelligent particle swarm optimization algorithm for solving cell formation problem, Neural Comput. Appl. 31 (2) (2019) 801–815.
[69] R. Kamalakannan, R. Sudhakara Pandian, T. Sornakumar, S. Mahapatra, An ant colony optimization algorithm for cellular manufacturing system, in: Applied Mechanics and Materials, vol. 854, Trans Tech Publ, 2017, pp. 133–141.
[70] B. Karoum, Y.B. Elbenani, Discrete cuckoo search algorithm for solving the cell formation problem, Int. J. Manuf. Res. 14 (3) (2019) 245–264.
[71] A. Arunagiri, U. Marimuthu, P. Gopalakrishnan, A. Slota, J. Zajac, M.P. Paulraj, Sustainability formation of machine cells in group technology systems using modified artificial bee colony algorithm, Sustainability 10 (1) (2018) 42.
[72] S. Ingole, D. Singh, Unequal-area, fixed-shape facility layout problems using the firefly algorithm, Eng. Optim. 49 (7) (2017) 1097–1115.
[73] Y. Wang, Y. Chen, Y. Lin, Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem, Comput. Ind. Eng. 106 (2017) 105–122.
[74] C.H. Song, K. Lee, W.D. Lee, Extended simulated annealing for augmented tsp and multi-salesmen tsp, in: Proceedings of the International Joint Conference on Neural Networks, 2003, vol. 3, IEEE, 2003, pp. 2340–2343.
[75] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega 34 (3) (2006) 209–219.
[76] Y. Zhu, L. Wu, Structure study of multiple traveling salesman problem using genetic algorithm, in: 2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE, 2019, pp. 323–328.
[77] L. Kota, K. Jarmai, Mathematical modeling of multiple tour multiple traveling salesman problem using evolutionary programming, Appl. Math. Model. 39 (12) (2015) 3410–3433.
[78] J.K. Chong, X. Qiu, An opposition-based self-adaptive differential evolution with decomposition for solving the multiobjective multiple salesman problem, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 4096–4103.
[79] A. Singh, A.S. Baghel, A new grouping genetic algorithm approach to the multiple traveling salesperson problem, Soft Computing 13 (1) (2009) 95–101.
[80] X. Chen, P. Zhang, G. Du, F. Li, Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems, IEEE Access 6 (2018) 21745–21757.
[81] S. Pang, T. Li, F. Dai, M. Yu, Particle swarm optimization algorithm for multi-salesman problem with time and capacity constraints, Applied Mathematics & Information Sciences 7 (6) (2013) 2439.
[82] M.H. Xue, T.Z. Wang, S. Mao, Double evolutsional artificial bee colony algorithm for multiple traveling salesman problem, in: MATEC Web of Conferences, vol. 44, EDP Sciences, 2016 02025.
[83] M. Mohammadi, G. Rahmanifar, G. G. KAVEH, Optimization Multiple Traveling Salesman Problem by Considering the Learning Effect Function in Skill and Workload Balancing of Salesman with Using the Firefly Algorithm.
[84] M.R. Montoya, R.G. Velázquez, M.E. Analco, J.L.M. Flores, M.B.B. Loranca, Solution search for the capacitated p-median problem using tabu search, International Journal of Combinatorial Optimization Problems and Informatics 10 (2) (2019) 17–25.
[85] S. Sakamoto, E. Kulla, T. Oda, M. Ikeda, L. Barolli, F. Xhafa, A comparison study of hill climbing, simulated annealing and genetic algorithm for node placement problem in wmns, J. High Speed Netw. 20 (1) (2014) 55–66.
[86] I. Davydov, Y. Kochetov, S. Dempe, Local search approach for the competitive facility location problem in mobile networks, Int. J. Artif. Intell. 16 (1) (2018) 130–143.
[87] K.M. Ferreira, T.A. de Queiroz, Two effective simulated annealing algorithms for the location-routing problem, Appl. Soft Comput. 70 (2018) 389–422.
[88] F.L. Biajoli, A.A. Chaves, L.A.N. Lorena, A biased random-key genetic algorithm for the two-stage capacitated facility location problem, Expert Syst. Appl. 115 (2019) 418–426.
[89] A. Teran-Somohano, A.E. Smith, Locating multiple capacitated semi-obnoxious facilities using evolutionary strategies, Comput. Ind. Eng. 133 (2019) 303–316.
[90] L. Pitaksringkarn, M.A. Taylor, Grouping genetic alogirhtm in gis: a facility location modelling, Journal of the Eastern Asia Society for Transportation Studies 6 (2005) 2908–2920.
[91] R. Chi, Y. Su, Z. Qu, X. Chi, A hybridization of cuckoo search and differential evolution for the logistics distribution center location problem, Math. Probl. Eng. (2019).

[92] T. Levanova, A. Gnusarev, Development of ant colony optimization algorithm for competitive p-median facility location problem with elastic demand, in: International Conference on Mathematical Optimization Theory and Operations Research, Springer, 2019, pp. 68–78.

[93] I. Osinuga, A. Bolarinwa, L. Kazakovtsev, A modified particle swarm optimization algorithm for location problem, in: IOP Conference Series: Materials Science and Engineering, vol. 537, IOP Publishing, 2019, p. 042060.

[94] A. Rahmani, S. MirHassani, A hybrid firefly-genetic algorithm for the capacitated facility location problem, Inf. Sci. 283 (2014) 70–78.

[95] A.B. Pratiwi, N. Faiza, E.E. Winarko, Penerapan cuckoo search algorithm (csa) untuk menyelesaikan uncapacitated facility location problem (uflp), Contemporary Mathematics and Applications 1 (1) (2019) 34–45.

[96] S.S. Choong, L.-P. Wong, C.P. Lim, An artificial bee colony algorithm with a modified choice function for the traveling salesman problem, Swarm and evolutionary computation 44 (2019) 622–635.

[97] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, Q. Pan, A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems, IEEE/CAA Journal of Automatica Sinica 6 (4) (2019) 904–916.

[98] F. Garza-Santisteban, R. Sánchez-Pámanes, L.A. Puente-Rodríguez, I. Amaya, J.C. Ortiz-Bayliss, S. Conant-Pablos, H. Terashima-Marín, A simulated annealing hyper-heuristic for job shop scheduling problems, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 57–64.

[99] F. Zhao, S. Qin, Y. Zhang, W. Ma, C. Zhang, H. Song, A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem, Expert Syst. Appl. 126 (2019) 321–339.

[100] M.E. Aydin, T.C. Fogarty, A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application, J. Intell. Manuf. 15 (6) (2004) 805–814.

[101] J.-Q. Li, P. Duan, J. Cao, X.-P. Lin, Y.-Y. Han, A hybrid pareto-based tabu search for the distributed flexible job shop scheduling problem with e/t criteria, IEEE Access 6 (2018) 58883–58897.

[102] A.A.R. Hosseinabadi, J. Vahidi, B. Saemi, A.K. Sangaiah, M. Elhoseny, Extended genetic algorithm for solving open-shop scheduling problem, Soft computing 23 (13) (2019) 5099–5116.

[103] J.C. Chen, C.-C. Wu, C.-W. Chen, K.-H. Chen, Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm, Expert Syst. Appl. 39 (11) (2012) 10016–10021.

[104] R. Varela, A. Gomez, C.R. Vela, J. Puente, C. Alonso, Heuristic generation of the initial population in solving job shop problems by evolutionary strategies, in: International Work-Conference on Artificial Neural Networks, Springer, 1999, pp. 690–699.

[105] S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Genetic programming for job shop scheduling, in: Evolutionary and Swarm Intelligence Algorithms, Springer, 2019, pp. 143–167.

[106] L. Wang, D.-Z. Zheng, A modified evolutionary programming for flow shop scheduling, Int. J. Adv. Manuf. Technol. 22 (7–8) (2003) 522–527.

[107] X. Wu, X. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, Memetic Computing 11 (4) (2019) 335–355.

[108] Z. Zhuang, Z. Huang, Z. Lu, L. Guo, Q. Cao, W. Qin, An improved artificial bee colony algorithm for solving open shop scheduling problem with two sequence-dependent setup times, Procedia CIRP 83 (2019) 563–568.

[109] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, A.C. Ammari, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, J. Intell. Manuf. 29 (3) (2018) 603–615.

[110] M. Feng, X. Yi, G. Li, S. Tang, H. Jun, A grouping particle swarm optimization algorithm for flexible job shop scheduling problem, in: 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, vol. 1, IEEE, 2008, pp. 332–336.

[111] R.K. Phanden, Z. Palková, R. Sindhwani, A framework for flexible job shop scheduling problem using simulation-based cuckoo search, Advances in Industrial and Production Engineering: Select Proceedings of FLAME 2018 (2019) 247.

[112] Z. Zhuang, Z. Huang, Z. Lu, L. Guo, Q. Cao, W. Qin, An improved artificial bee colony algorithm for solving open shop scheduling problem with two sequence-dependent setup times, Procedia CIRP 83 (2019) 563–568.

[113] B. Fan, W. Yang, Z. Zhang, Solving the two-stage hybrid flow shop scheduling problem based on mutant firefly algorithm, Journal of Ambient Intelligence and Humanized Computing 10 (3) (2019) 979–990.

[114] A. Baykasoglu, T. Dereli, S. Das, Project team selection using fuzzy optimization approach, Cybern. Syst.: Int. J. 38 (2) (2007) 155–185.

[115] H. Wi, S. Oh, J. Mun, M. Jung, A team formation model based on knowledge and collaboration, Expert Syst. Appl. 36 (5) (2009) 9121–9134.

[116] H. Wang, Z. Ren, X. Li, H. Jiang, Solving team making problem for crowdsourcing with evolutionary strategy, in: 2018 5th International Conference on Dependable Systems and Their Applications (DSA), IEEE, 2018, pp. 65–74.

[117] L.E. Agustín-Blas, S. Salcedo-Sanz, E.G. Ortiz-García, A. Portilla-Figueras, Á.M. Pérez-Bellido, S. Jiménez-Fernández, Team formation based on group technology: a hybrid grouping genetic algorithm approach, Comput. Oper. Res. 38 (2) (2011) 484–495.

[118] J.A. Delgado-Osuna, M. Lozano, C. García-Martínez, An alternative artificial bee colony algorithm with destructive–constructive neighbourhood operator for the problem of composing medical crews, Inf. Sci. 326 (2016) 215–226.

[119] W.H. El-Ashmawi, A.F. Ali, M.A. Tawhid, An improved particle swarm optimization with a new swap operator for team formation problem, Journal of Industrial Engineering International 15 (1) (2019) 53–71.

[120] M. Bello, R. Bello, A. Nowé, M.M. García-Lorenzo, A method for the team selection problem between two decision-makers using the ant colony optimization, in: Soft Computing Applications for Group Decision-Making and Consensus Modeling, Springer, 2018, pp. 391–410.

[121] R. Sanaei, K. Otto, K. Wood, K. Hölttä-Otto, et al., A rapid algorithm for multi-objective pareto optimization of modular architecture, in: DS 87-4 Proceedings of the 21st International Conference on Engineering Design (ICED 17), vol. 4, Design Methods and Tools, Vancouver, Canada, 2017, pp. 169–178. 21-25.08. 2017.

[122] M. Mutingi, C. Mbohwa, Modeling modular design for sustainable manufacturing: a fuzzy grouping genetic algorithm approach, in: Grouping Genetic Algorithms, Springer, 2017, pp. 199–211.

[123] H.G. Wahdan, S.S. Kassem, H.M. Abdelsalam, Product modularization using cuckoo search algorithm, in: International Conference on Operations Research and Enterprise Systems, Springer, 2016, pp. 20–34.

[124] O. Durán, L. Pérez, A. Batocchio, Optimization of modular structures using particle swarm optimization, Expert Syst. Appl. 39 (3) (2012) 3507–3515.

[125] J. Balasch-Masoliver, V. Muntés-Mulero, J. Nin, Using genetic algorithms for attribute grouping in multivariate microaggregation, Intell. Data Anal. 18 (5) (2014) 819–836.

[126] E. Fayyoumi, O. Nofal, Applying genetic algorithms on multi-level micro-aggregation techniques for secure statistical databases, in: 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), IEEE, 2018, pp. 1–6.

[127] A.A. Aksut, Population-based Ant Colony Optimization for Multivariate Microaggregation, Nova Southeastern University, 2013.

[128] F. Roanne, A variable neighborhood search with integer programming for the zero-one multiple-choice knapsack problem with setup, Variable Neighborhood Search (2019) 152.

[129] A. Hiley, B.A. Julstrom, The quadratic multiple knapsack problem and three heuristic approaches to it, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, 2006, pp. 547–552.

[130] J. Qin, X. Xu, Q. Wu, T. Cheng, Hybridization of tabu search with feasible and infeasible local searches for the quadratic multiple knapsack problem, Comput. Oper. Res. 66 (2016) 199–214.

[131] A.S. Fukunaga, A new grouping genetic algorithm for the multiple knapsack problem, in: 2008 IEEE Congress on Evolutionary Computation, IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 2225–2232.

[132] A.N. Ünal, A genetic algorithm for the multiple knapsack problem in dynamic environment, in: Proceedings of the World Congress on Engineering and Computer Science, vol. 2, 2013.

[133] D. Libao, W. Sha, J. Chengyu, H. Cong, A hybrid mutation scheme-based discrete differential evolution algorithm for multidimensional knapsack problem, in: 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), IEEE, 2016, pp. 1009–1014.

[134] J.H. Drake, M. Hyde, K. Ibrahim, E. Ozcan, A genetic programming hyper-heuristic for the multidimensional knapsack problem, Kybernetes 43 (9/10) (2014) 1500–1511.

[135] M. Kong, P. Tian, Y. Kao, A new ant colony optimization algorithm for the multidimensional knapsack problem, Comput. Oper. Res. 35 (8) (2008) 2672–2683.

[136] X. Ma, Y. Zhang, A particle swarm optimization based on many-objective for multiple knapsack problem, in: 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2019, pp. 260–265.

[137] S. Sabet, M. Shokouhifar, F. Farokhi, A discrete artificial bee colony for multiple knapsack problem, Int. J. Reas. Based Intell. Syst. 5 (2) (2013) 88–95.

[138] A. Baykasoğlu, F.B. Ozsoydan, An improved firefly algorithm for solving dynamic multidimensional knapsack problems, Expert Syst. Appl. 41 (8) (2014) 3712–3725.

[139] M. Hifi, Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems, J. Comb. Optim. 8 (1) (2004) 65–84.

[140] F. Dusberger, G.R. Raidl, Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search, Electron. Notes Discrete Math. 47 (2015) 133–140.

[141] I.A.L. Sanchez, J.M. Vargas, C.A. Santos, M.G. Mendoza, C.J.M. Moctezuma, Solving binary cutting stock with matheuristics using particle swarm optimization and simulated annealing, Soft Computing 22 (18) (2018) 6111–6119.

[142] M.H. Jahromi, R. Tavakkoli-Moghaddam, A. Makui, A. Shamsi, Solving an one-dimensional cutting stock problem by simulated annealing and tabu search, Journal of Industrial Engineering International 8 (1) (2012) 24.

[143] S. Khebbache, C. PRINS, A. Yalaoui, Iterated Local Search Algorithm for the Constrained Two-Dimensional Non-guillotine Cutting Problem.

[144] A. Orlov, V. Kureichik, A. Glushchenko, Hybrid genetic algorithm for cutting stock and packaging problems, in: 2016 IEEE East-West Design & Test Symposium (EWDTS), IEEE, 2016, pp. 1–4.

[145] R. Chiong, Y.Y. Chang, P.C. Chai, A.L. Wong, A selective mutation based evolutionary programming for solving cutting stock problem without contiguity, in: 2008 IEEE Congress on Evolutionary Computation, IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1671–1677.

[146] P. Li, C. Wang, Y. Mao, A hybrid grouping genetic algorithm for one-dimensional cutting stock problem, Journal-Shanghai Jiaotong University-Chinese Edition- 40 (6) (2006) 1015.

[147] G. Evtimov, S. Fidanova, Ant colony optimization algorithm for 1d cutting stock problem, in: Advanced Computing in Industrial Mathematics, Springer, 2018, pp. 25–31.

[148] E.F. Shair, S. Khor, A. Abdullah, H. Jaafar, N. Saharuddin, A.Z. Abidin, Cuckoo search approach for cutting stock problem, International Journal of Information and Electronics Engineering 5 (2) (2015) 138.

[149] Y. Xu, A novel grouping particle swarm optimization approach for 2d irregular cutting stock problem, International Journal of Control and Automation 9 (8) (2016) 369–380.

[150] M. Yusoff, N. Roslan, Evaluation of genetic algorithm and hybrid genetic algorithm-hill climbing with elitist for lecturer university timetabling problem, in: International Conference on Swarm Intelligence, Springer, 2019, pp. 363–373.

[151] R.A. Aziz, M. Ayob, Z. Othman, Z. Ahmad, N.R. Sabar, An adaptive guided variable neighborhood search based on honey-bee mating optimization algorithm for the course timetabling problem, Soft Computing 21 (22) (2017) 6755–6765.

[152] N. Leite, F. Melício, A.C. Rosa, A fast simulated annealing algorithm for the examination timetabling problem, Expert Syst. Appl. 122 (2019) 137–151.

[153] P. Amaral, T.C. Pais, Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling, Comput. Oper. Res. 72 (2016) 160–174.

[154] S. Ribić, S. Konjicija, Evolution strategy to make an objective function in two-phase ilp timetabling, in: 2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers, IEEE, 2011, pp. 1486–1489.

[155] R. Raghavjee, N. Pillay, A comparison of genetic algorithms and genetic programming in solving the school timetabling problem, in: 2012 Fourth World Congress on Nature and Biologically Inspired Computing, NaBIC), IEEE, 2012, pp. 98–103.

[156] K. Shaker, S. Abdullah, A. Hatem, A differential evolution algorithm for the university course timetabling problem, in: 2012 4th Conference on Data Mining and Optimization (DMO), IEEE, 2012, pp. 99–102.

[157] L.E. Agustín-Blas, S. Salcedo-Sanz, E.G. Ortiz-García, A. Portilla-Figueras, Á.M. Pérez-Bellido, A hybrid grouping genetic algorithm for assigning students to preferred laboratory groups, Expert Syst. Appl. 36 (3) (2009) 7234–7241.

[158] M. Mazlan, M. Makhtar, A.F.K.A. Khairi, M.A. Mohamed, University course timetabling model using ant colony optimization algorithm approach, Indonesian Journal of Electrical Engineering and Computer Science 13 (1) (2019) 72–76.

[159] T. Thepphakorn, P. Pongcharoen, Variants and parameters investigations of particle swarm optimisation for solving course timetabling problems, in: International Conference on Swarm Intelligence, Springer, 2019, pp. 177–187.

[160] T. Thepphakorn, P. Pongcharoen, S. Vitayasak, A new multiple objective cuckoo search for university course timetabling problem, in: International Workshop on Multi-Disciplinary Trends in Artificial Intelligence, Springer, 2016, pp. 196–207.

[161] D. Ojha, R. K. Sahoo, S. Das, Automatic generation of timetable using firefly algorithm, Int. J. 6 (4).

[162] L.M. Abualigah, A.T. Khader, M.A. Al-Betar, Z.A.A. Alyasseri, O.A. Alomari, E.S. Hanandeh, Feature selection with β-hill climbing search for text clustering application, in: 2017 Palestinian International Conference on Information and Communication Technology (PICICT), IEEE, 2017, pp. 22–27.

[163] J. Brimberg, N. Mladenović, R. Todosijević, D. Urošević, Solving the capacitated clustering problem with variable neighborhood search, Ann. Oper. Res. 272 (1–2) (2019) 289–321.

[164] S. Seifollahi, A. Bagirov, E. Zare Borzeshi, M. Piccardi, A simulated annealing-based maximum-margin clustering algorithm, Comput. Intell. 35 (1) (2019) 23–41.

[165] A.F. Yaqoob, B. Al-Sarray, Finding best clustering for big networks with minimum objective function by using probabilistic tabu search, Iraqi J. Sci. 60 (8) (2019) 1837–1845.

[166] M. El-Shorbagy, A. Ayoub, A. Mousa, I. El-Desoky, An enhanced genetic algorithm with new mutation for cluster analysis, Comput. Stat. (2019) 1–38.

[167] C.-Y. Lee, E. Antonsson, Dynamic partitional clustering using evolution strategies, in: 2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies, vol. 4, IEEE, 2000, pp. 2716–2721.

[168] Q. Tan, Q. He, W. Zhao, Z. Shi, E.S. Lee, An improved fcmbp fuzzy clustering method based on evolutionary programming, Comput. Math. Appl. 61 (4) (2011) 1129–1144.

[169] A. Lensen, B. Xue, M. Zhang, Genetic Programming for Evolving Similarity Functions for Clustering: Representations and Analysis, Evolutionary computation, 2019, pp. 1–29.

[170] M. Alswaitti, M. Albughdadi, N.A.M. Isa, Variance-based differential evolution algorithm with an optional crossover for data clustering, Appl. Soft Comput. 80 (2019) 1–17.

[171] S.H. Razavi, E.O.M. Ebadati, S. Asadi, H. Kaur, An efficient grouping genetic algorithm for data clustering and big data analysis, in: Computational Intelligence for Big Data Analysis, Springer, 2015, pp. 119–142.

[172] A.H. Kashan, B. Rezaee, S. Karimiyan, An efficient approach for unsupervised fuzzy clustering based on grouping evolution strategies, Pattern Recognit. 46 (5) (2013) 1240–1254.

[173] R. Subekti, E. Sari, R. Kusumawati, Ant colony algorithm for clustering in portfolio optimization, in: Journal of Physics: Conference Series, vol. 983, IOP Publishing, 2018, p. 012096.

[174] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A new feature selection method to improve the document clustering using particle swarm optimization algorithm, Journal of Computational Science 25 (2018) 456–466.

[175] K.G. Dhal, A. Das, S. Ray, S. Das, A clustering based classification approach based on modified cuckoo search algorithm, Pattern Recognit. Image Anal. 29 (3) (2019) 344–359.

[176] C. Zhang, D. Ouyang, J. Ning, An artificial bee colony approach for clustering, Expert Syst. Appl. 37 (7) (2010) 4761–4767.

[177] A. Namdev, B. Tripathy, Scalable rough c-means clustering using firefly algorithm, International Journal of Computer Science and Business Informatics 16 (2) (2016) 1–14.

[178] A. Duarte, R. Martí, Tabu search and grasp for the maximum diversity problem, Eur. J. Oper. Res. 178 (1) (2007) 71–84.

[179] G. Palubeckis, E. Karčiauskas, A. Riškus, Comparative performance of three metaheuristic approaches for the maximally diverse grouping problem, Inf. Technol. Control 40 (4) (2011) 277–285.

[180] A.R.R. de Freitas, F.G. Guimarães, R.C.P. Silva, M.J.F. Souza, Memetic self-adaptive evolution strategies applied to the maximum diversity problem, Optimization Letters 8 (2) (2014) 705–714.

[181] K. Singh, S. Sundar, A new hybrid genetic algorithm for the maximally diverse grouping problem, International Journal of Machine Learning and Cybernetics (2019) 1–20.

[182] F.J. Rodriguez, M. Lozano, C. García-Martínez, J.D. GonzáLez-Barrera, An artificial bee colony algorithm for the maximally diverse grouping problem, Inf. Sci. 230 (2013) 183–196.

[183] J. Luan, Z. Yao, F. Zhao, X. Song, A novel method to solve supplier selection problem: hybrid algorithm of genetic algorithm and ant colony optimization, Math. Comput. Simulat. 156 (2019) 294–309.

[184] M. Mutingi, C. Mbohwa, Modeling supplier selection using multi-criterion fuzzy grouping genetic algorithm, in: Grouping Genetic Algorithms, Springer, 2017, pp. 213–228.

[185] S.K. Jauhar, M. Pant, Sustainable supplier selection: a new differential evolution strategy with automotive industry application, in: Recent Developments and New Direction in Soft-Computing Foundations and Applications, Springer, 2016, pp. 353–371.

[186] A. Fallahpour, E.U. Olugu, S.N. Musa, D. Khezrimotlagh, K.Y. Wong, An integrated model for green supplier selection under fuzzy environment: application of data envelopment analysis and genetic programming approach, Neural Comput. Appl. 27 (3) (2016) 707–725.

[187] B. Yuce, E. Mastrocinque, A hybrid approach using the bees algorithm and fuzzy-ahp for supplier selection, in: Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering, IGI Global, 2016, pp. 171–194.

[188] R. Kuo, S. Hong, Y. Huang, Integration of particle swarm optimization-based fuzzy neural network and artificial neural network for supplier selection, Appl. Math. Model. 34 (12) (2010) 3976–3990.

[189] G. Kanagaraj, S. Ponnambalam, N. Jawahar, Reliability-based total cost of ownership approach for supplier selection using cuckoo-inspired hybrid algorithm, Int. J. Adv. Manuf. Technol. 84 (5–8) (2016) 801–816.

[190] F. Wang, Z. Xu, Metaheuristics for robust graph coloring, J. Heuristics 19 (4) (2013) 529–548.

[191] A. Kose, B. A. Sonmez, M. Balaban, Simulated annealing algorithm for graph coloring, arXiv preprint arXiv:1712.00709.

[192] D. Matic, J. Kratica, V. Filipovic, Variable neighborhood search for solving bandwidth coloring problem, Comput. Sci. Inf. Syst. 14 (2) (2015) 309–327.

[193] S. Labed, K. Akram, S. Chikhi, Solving the graph b-coloring problem with hybrid genetic algorithm, in: 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), IEEE, 2018, pp. 1–7.

[194] P. Tolay, R. Kumar, Evolution of hyperheuristics for the biobjective graph coloring problem using multiobjective genetic programming, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, ACM, 2009, pp. 1939–1940.

[195] S. Khuri, T. Walters, Y. Sugono, A grouping genetic algorithm for coloring the edges of graphs, in: SAC, vol. 1, Citeseer, 2000, pp. 422–427.

[196] I. Fister, J. Brest, Using differential evolution for the graph coloring, in: 2011 IEEE Symposium on Differential Evolution (SDE), IEEE, 2011, pp. 1–7.

[197] L. Lv, C. Gao, J. Chen, L. Luo, Z. Zhang, Physarum-based ant colony optimization for graph coloring problem, in: International Conference on Swarm Intelligence, Springer, 2019, pp. 210–219.

[198] Z.-s. RAO, W.-y. ZHU, K. ZHANG, Solving Graph Coloring Problem Using Parallel Discrete Particle Swarm Optimization on Cuda, DEStech Transactions on Engineering and Technology Research (amsm).

[199] C. Aranha, K. Toda, H. Kanoh, Solving the graph coloring problem using cuckoo search, in: International Conference on Swarm Intelligence, Springer, 2017, pp. 552–560.

[200] K. Chen, H. Kanoh, A discrete artificial bee colony algorithm based on similarity for graph coloring problems, in: International Conference on Theory and Practice of Natural Computing, Springer, 2016, pp. 73–84.

[201] K. Chen, H. Kanoh, A discrete firefly algorithm based on similarity for graph coloring problems, in: 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2017, pp. 65–70.

[202] Y.-Y. Chen, C.-Y. Cheng, L.-C. Wang, T.-L. Chen, A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems—a case study for solar cell industry, Int. J. Prod. Econ. 141 (1) (2013) 66–78.

[203] S.-W. Lin, K.-C. Ying, A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems, Int. J. Prod. Res. 53 (4) (2015) 1065–1076.

[204] V. Sels, J. Coelho, A.M. Dias, M. Vanhoucke, Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem, Comput. Oper. Res. 53 (2015) 107–117.

[205] M. H. Danurhadi, D. D. Damayanti, W. Juliani, Identical parallel machine scheduling using genetic algorithm to minimize total tardiness for cnc 4 axis in pt dirgantara Indonesia (persero), eProceedings of Engineering 6 (2).

[206] C.-C. Chyu, W.-S. Chang, A competitive evolution strategy memetic algorithm for unrelated parallel machine scheduling to minimize total weighted tardiness and flow time, in: The 40th International Conference on Computers & Indutrial Engineering, IEEE, 2010, pp. 1–6.

[207] A.-Q. Yu, X.-S. Gu, Hybrid quantum-inspired evolutionary programming for identical parallel machines scheduling, Control Decis. 26 (10) (2011) 1473–1478.

[208] M. Durasević, D. Jakobović, K. Knežević, Adaptive scheduling on unrelated machines with genetic programming, Appl. Soft Comput. 48 (2016) 419–430.

[209] X. Wu, A. Che, A memetic differential evolution algorithm for energy-efficient parallel machine scheduling, Omega 82 (2019) 155–165.

[210] T.W. Liao, P. Su, Parallel machine scheduling in fuzzy environment with hybrid ant colony optimization including a comparison of fuzzy number ranking methods in consideration of spread of fuzziness, Appl. Soft Comput. 56 (2017) 65–81.

[211] S. Pakzad-Moghaddam, A lévy flight embedded particle swarm optimization for multi-objective parallel-machine scheduling with learning and adapting considerations, Comput. Ind. Eng. 91 (2016) 109–128.

[212] D. Laha, J.N. Gupta, An improved cuckoo search algorithm for scheduling jobs on identical parallel machines, Comput. Ind. Eng. 126 (2018) 348–360.

[213] E. Caniyilmaz, B. Benli, M.S. Ilkay, An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date, Int. J. Adv. Manuf. Technol. 77 (9–12) (2015) 2105–2115.

[214] A.E. Ezugwu, F. Akutsah, An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times, IEEE Access 6 (2018) 54459–54478.

[215] S. Henn, V. Schmid, Metaheuristics for order batching and sequencing in manual order picking systems, Comput. Ind. Eng. 66 (2) (2013) 338–351.

[216] Z. Pei, Z. Wang, Y. Yang, Research of order batching variable neighborhood search algorithm based on saving mileage, in: 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019), Atlantis Press, 2019.

[217] E.H. Grosse, C.H. Glock, R. Ballester-Ripoll, et al., A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions, Int. J. Oper. Quant. Manag. 20 (2) (2014) 65–83.

[218] I. Žulj, S. Kramer, M. Schneider, A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem, Eur. J. Oper. Res. 264 (2) (2018) 653–664.

[219] C.M. Hsu, K.Y. Chen, M.C. Chen, Batching orders in warehouses by minimizing travel distance with genetic algorithms, Comput. Ind. 56 (2) (2005) 169–178.

[220] C.-Y. Cheng, Y.-Y. Chen, T.-L. Chen, J.J.-W. Yoo, Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem, Int. J. Prod. Econ. 170 (2015) 805–814.

[221] C.-Y. Cheng, Y.-Y. Chen, T.-L. Chen, J.J.-W. Yoo, Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem, Int. J. Prod. Econ. 170 (2015) 805–814.

[222] Z. Li, Z. Zhou, An effective batching method based on the artificial bee colony algorithm for order picking, in: 2013 Ninth International Conference on Natural Computation (ICNC), IEEE, 2013, pp. 386–391.

[223] A.H. Kashan, M.H. Kashan, S. Karimiyan, A particle swarm optimizer for grouping problems, Inf. Sci. 252 (2013) 81–95.

[224] M. Cissé, S. Yalçındağ, Y. Kergosien, E. Şahin, C. Lenté, A. Matta, Or problems related to home health care: a review of relevant routing and scheduling problems, Operations Research for Health Care 13 (2017) 1–22.

[225] S. Frifita, M. Masmoudi, J. Euchi, General variable neighborhood search for home healthcare routing and scheduling problem with time windows and synchronized visits, Electron. Notes Discrete Math. 58 (2017) 63–70.

[226] A.M. Fathollahi-Fard, K. Govindan, M. Hajiaghaei-Keshteli, A. Ahmadi, A green home health care supply chain: new modified simulated annealing algorithms, J. Clean. Prod. 240 (2019) 118200.

[227] Z. Yihe, L. Ran, H. Yikang, Tabu search algorithm for periodic home health care problem, China Sciencepaper (14) (2015) 22.

[228] R. Borchani, M. Masmoudi, B. Jarboui, Hybrid genetic algorithm for home healthcare routing and scheduling problem, in: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, 2019, pp. 1900–1904.

[229] M. Mutingi, C. Mbohwa, Home health care staff scheduling: effective grouping approaches, in: IAENG Transactions on Engineering Sciences-Special Issue of the International Multi-Conference of Engineers and Computer Scientists, IMECS and World Congress on Engineering, CRC Press, Taylor & Francis Group, 2014, pp. 215–224.

[230] T. Zhang, X. Yang, Q. Chen, L. Bai, W. Chen, Modified aco for home health care scheduling and routing problem in Chinese communities, in: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2018, pp. 1–6.

[231] C. Akjiratikarl, P. Yenradee, P.R. Drake, Pso-based algorithm for home care worker scheduling in the UK, Comput. Ind. Eng. 53 (4) (2007) 559–583.

[232] L. Dekhici, R. Redjem, K. Belkadi, A. El Mhamedi, Discretization of the firefly algorithm for home care, Can. J. Electr. Comput. Eng. 42 (1) (2019) 20–26.

[233] C.d.C. Aranha, H. Iba, Using memetic algorithms to improve portfolio performance in static and dynamic trading scenarios, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, ACM, 2009, pp. 1427–1434.

[234] Y. Crama, M. Schyns, Simulated annealing for complex portfolio selection problems, Eur. J. Oper. Res. 150 (3) (2003) 546–571.

[235] M.M. Aldaihani, T.M. Al-Deehani, Mathematical models and a tabu search for the portfolio management problem in the Kuwait stock exchange, Int. J. Oper. Res. 7 (4) (2010) 445–462.

[236] R. Yusuf, B. Handari, G. Hertono, Implementation of agglomerative clustering and genetic algorithm on stock portfolio optimization with possibilistic constraints, in: AIP Conference Proceedings, vol. 2168, AIP Publishing, 2019, p. 020028.

[237] P. Lipinski, K. Winczura, J. Wojcik, Building risk-optimal portfolio using evolutionary strategies, in: Workshops on Applications of Evolutionary Computation, Springer, 2007, pp. 208–217.

[238] L. Wagman, Stock portfolio evaluation: an application of genetic-programming-based technical analysis, Genetic Algorithms and Genetic Programming at Stanford 2003 (2003) 213–220.

[239] A. Adebiyi, C. Ayo, Portfolio selection problem using generalized differential evolution 3, Appl. Math. Sci. 9 (42) (2015) 2069–2082.

[240] C.H. Chen, C.Y. Lu, C.B. Lin, An intelligence approach for group stock portfolio optimization with a trading mechanism, Knowl. Inf. Syst. (2019) 1–30.

[241] A. Steven, G.F. Hertono, B.D. Handari, Clustered stocks weighting with ant colony optimization in portfolio optimization, in: AIP Conference Proceedings, vol. 2023, AIP Publishing, 2018 020204.

[242] E. Bronshtein, O. Kondrateva, The decision support of the securities portfolio composition based on the particle swarm optimization, in: 7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019), Atlantis Press, 2019.

[243] E. Shadkam, R. Delavari, F. Memariani, M. Poursaleh, Portfolio selection by the means of cuckoo optimization algorithm, International Journal on Computational Science & Application.

[244] T. Maydina, G. Hertono, B. Handari, Implementation of agglomerative clustering and modified artificial bee colony algorithm on stock portfolio optimization with possibilistic constraints, in: AIP Conference Proceedings, vol. 2168, AIP Publishing, 2019 020030.

[245] H. Heidari, L. Neshatizadeh, Stock portfolio-optimization model by mean-semi-variance approach using of firefly algorithm and imperialist competitive algorithm, International Journal of Business and Development Studies 10 (1) (2018) 115–143.

[246] A.H. Kashan, A.A. Akbari, B. Ostadi, Grouping evolution strategies: an effective approach for grouping problems, Appl. Math. Model. 39 (9) (2015) 2703–2720.

[247] R. Cheng, M. Gen, Parallel machine scheduling problems using memetic algorithms, in: 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929), vol. 4, IEEE, 1996, pp. 2665–2670.

[248] M. Mutingi, C. Mbohwa, Grouping genetic algorithms: advances for real-world grouping problems, in: Grouping Genetic Algorithms, Springer, 2017, pp. 45–66.

[249] A.S. Mendes, F.M. Müller, P.M. França, P. Moscato, Comparing meta-heuristic approaches for parallel machine scheduling problems, Prod. Plan. Control 13 (2) (2002) 143–154.

[250] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.

[251] M. Buljubasic, Efficient Local Search for Several Combinatorial Optimization Problems, Theses. Université Montpellier, Nov. 2015https://tel.archives-ouvertes.fr/tel-01320380.

[252] M. Chiarandini, I. Dumitrescu, T. Stützle, Stochastic Local Search Algorithms for the Graph Colouring Problem, Computer & Information Science Series, Chapman & Hall, CRC.

[253] H.G. Santos, T.A. Toffolo, C.L. Silva, G. Vanden Berghe, Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem, Int. Trans. Oper. Res. 26 (2) (2019) 707–724.

[254] E.R.R. Kato, G.D. de Aguiar Aranha, R.H. Tsunaki, A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and random-restart hill climbing, Comput. Ind. Eng. 125 (2018) 178–189.

[255] R. Aringhieri, Composing medical crews with equity and efficiency, Cent. Eur. J. Oper. Res. 17 (3) (2009) 343–357.

[256] C. Peng, G. Wu, T. W. Liao, H. Wang, Research on multi-agent genetic algorithm based on tabu search for the job shop scheduling problem, PLoS One 14 (9).

[257] o. Stakic, A. Anokic, R. Jovanovic, Comparison of different grasp algorithms for the heterogeneous vector bin packing problem, in: 2019 China-Qatar International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing (AIAIM), IEEE, 2019, pp. 63–70.

[258] S. Octarina, S. Yahdin, B. Wardhani, Implementasi algoritma greedy randomized adaptive search procedure (grasp) dan formulasi model dotted board pada penyelesaian cutting stock problem bentuk irregular, in: Annual Research Seminar (ARS), vol. 4, 2019, pp. 228–233.

[259] T. Öncan, Milp formulations and an iterated local search algorithm with tabu thresholding for the order batching problem, Eur. J. Oper. Res. 243 (1) (2015) 142–155.

[260] M. de Siqueira Guersola, M.T.A. Steiner, Iterated local search adapted to clustering and routing problems, in: 2015 Latin America Congress on Computational Intelligence (LA-CCI), IEEE, 2015, pp. 1–6.

[261] E.R. Hruschka, R.J. Campello, A.A. Freitas, et al., A survey of evolutionary algorithms for clustering, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39 (2) (2009) 133–155.

[262] E. Falkenauer, The grouping genetic algorithms-widening the scope of the gas, Belg. J. Oper. Res. Stat. Comput. Sci. 33 (1) (1992) 2.

[263] M. Mutingi, C. Mbohwa, Optimizing order batching in order picking systems: hybrid grouping genetic algorithm, in: Grouping Genetic Algorithms, Springer, 2017, pp. 121–140.
[264] A.H. Kashan, M. Keshmiry, J.H. Dahooie, A. Abbasi-Pooya, A simple yet effective grouping evolutionary strategy (ges) algorithm for scheduling parallel machines, Neural Comput. Appl. 30 (6) (2018) 1925–1938.
[265] H.-F. Yan, C.-Y. Cai, D.-H. Liu, M.-X. Zhang, Water wave optimization for the multidimensional knapsack problem, in: International Conference on Intelligent Computing, Springer, 2019, pp. 688–699.
[266] F. Zhao, H. Liu, Y. Zhang, W. Ma, C. Zhang, A discrete water wave optimization algorithm for no-wait flow shop scheduling problem, Expert Syst. Appl. 91 (2018) 347–363.
[267] S.A. Ludwig, A. Moallem, Swarm intelligence approaches for grid load balancing, J. Grid Comput. 9 (3) (2011) 279–301.
[268] B. Jarboui, S. Ibrahim, P. Siarry, A. Rebai, A combinatorial particle swarm optimisation for solving permutation flowshop problems, Comput. Ind. Eng. 54 (3) (2008) 526–538.
[269] S. T. Milan, L. Rajabion, H. Ranjbar, N. J. Navimipoir, Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments, Comput. Oper. Res..
[270] W. Guo, M. Chen, L. Wang, Y. Mao, Q. Wu, A survey of biogeography-based optimization, Neural Comput. Appl. 28 (8) (2017) 1909–1926.
[271] F. Zhao, S. Qin, Y. Zhang, W. Ma, C. Zhang, H. Song, A two-stage differential biogeography-based optimization algorithm and its performance analysis, Expert Syst. Appl. 115 (2019) 329–345.
[272] Q. Niu, T. Zhou, L. Wang, A hybrid particle swarm optimization for parallel machine total tardiness scheduling, Int. J. Adv. Manuf. Technol. 49 (5–8) (2010) 723–739.
[273] E.L. Lawler, J.K. Lenstra, A.R. Kan, Recent developments in deterministic sequencing and scheduling: a survey, in: Deterministic and Stochastic Scheduling, Springer, 1982, pp. 35–73, https://doi.org/10.1007/978-94-009-7801-03.
[274] J.K. Lenstra, D.B. Shmoys, E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Math. Program. 46 (1–3) (1990) 259–271.
[275] L. Fanjul-Peyro, R. Ruiz, Iterated greedy local search methods for unrelated parallel machine scheduling, Eur. J. Oper. Res. 207 (1) (2010) 55–69.
[276] M. Quiroz-Castellanos, L. Cruz-Reyes, J. Torres-Jimenez, C. Gómez, H.J.F. Huacuja, A.C. Alvim, A grouping genetic algorithm with controlled gene transmission for the bin packing problem, Comput. Oper. Res. 55 (2015) 52–64.
[277] A. H. Kashan, M. H. Kashan, S. Karimiyan, Grouping Genetic Algorithm for Industrial Engineering Applications.
[278] O.H. Ibarra, C.E. Kim, Heuristic algorithms for scheduling independent tasks on nonidentical processors, J. Assoc. Comput. Mach. 24 (2) (1977) 280–289.