Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 545191, 12 pages
http://dx.doi.org/10.1155/2014/545191

*Research Article*

# Improving the Bin Packing Heuristic through Grammatical Evolution Based on Swarm Intelligence

**Marco Aurelio Sotelo-Figueroa,[1] Héctor José Puga Soberanes,[1] Juan Martín Carpio,[1] Héctor J. Fraire Huacuja,[2] Laura Cruz Reyes,[2] and Jorge Alberto Soria-Alcaraz[1]**

[1] *División de Estudios de Posgrado e Investigación, Instituto Tecnológico de León, León 37290, GTO, Mexico*
[2] *División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero, Ciudad Madero 89440, TAMPS, Mexico*

Correspondence should be addressed to Marco Aurelio Sotelo-Figueroa; marco.sotelo@itleon.edu.mx

Received 11 April 2014; Revised 22 June 2014; Accepted 23 June 2014; Published 24 July 2014

Academic Editor: Ker-Wei Yu

In recent years Grammatical Evolution (GE) has been used as a representation of Genetic Programming (GP) which has been applied to many optimization problems such as symbolic regression, classification, Boolean functions, constructed problems, and algorithmic problems. GE can use a diversity of searching strategies including Swarm Intelligence (SI). Particle Swarm Optimisation (PSO) is an algorithm of SI that has two main problems: premature convergence and poor diversity. Particle Evolutionary Swarm Optimization (PESO) is a recent and novel algorithm which is also part of SI. PESO uses two perturbations to avoid PSO's problems. In this paper we propose using PESO and PSO in the frame of GE as strategies to generate heuristics that solve the Bin Packing Problem (BPP); it is possible however to apply this methodology to other kinds of problems using another Grammar designed for that problem. A comparison between PESO, PSO, and BPP's heuristics is performed through the nonparametric Friedman test. The main contribution of this paper is proposing a Grammar to generate online and offline heuristics depending on the test instance trying to improve the heuristics generated by other grammars and humans; it also proposes a way to implement different algorithms as search strategies in GE like PESO to obtain better results than those obtained by PSO.

## 1. Introduction

The methodology development to solve a specific problem is a process that entails the problem study and the analysis instances from such problem. There are many problems [1] for which there are no methodologies that can provide the exact solution, because the size of the problem search space makes it intractable in time, and it makes it necessary to search and improve methodologies that can give a solution in a finite time. There are methodologies based on Artificial Intelligence which do not yield exact solutions; those methodologies, however, provide an approximation, and among those we can find the following methodologies.

*Heuristics* are defined as "a type of strategy that dramatically limits the search for solutions" [2, 3]. One important characteristic of heuristics is that they can obtain a result for an instance problem in polynomialtime [1], although heuristics are developed for a specific instance problem.

*Metaheuristics* are defined as "a master strategy that guides and modifies other heuristics to obtain solutions generally better that the ones obtained with a local search optimization" [4]. The metaheuristics can work over several instances of a given problem or various problems, but it is necessary to adapt the metaheuristics to work with each problem.

It has been shown that the metaheuristic Genetic Programming [5] can generate a heuristic that can be applied to an instance problem [6]. There also exist metaheuristics that are based on Genetic Programming's paradigm [7] such as *Grammatical Differential Evolution* [8], *Grammatical Swarm* [9], *Particle Swarm Programming* [10], and *Geometric Differential Evolution* [11].

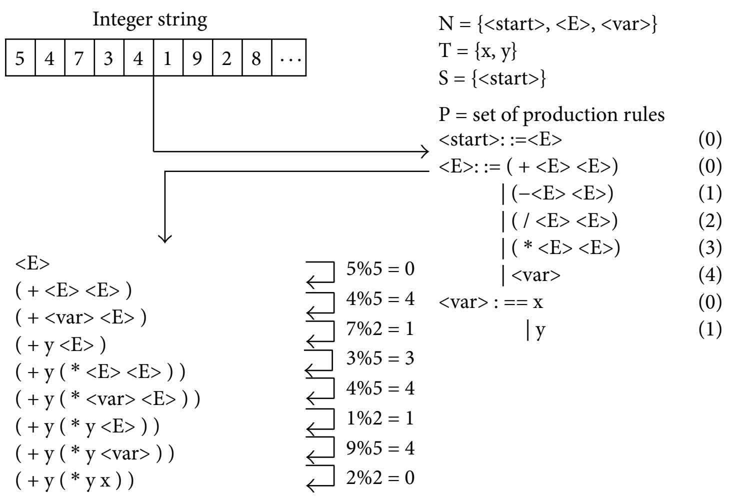

FIGURE 1: An example of a transformation from genotype to phenotype using a BNF Grammar. It begins with the start symbol, if the production rule for this symbol is only one rule, then the production rule replaces the start symbol, and the process begins choosing the production rules based on the current genotype. It takes each genotype and the nonterminal symbol from the left to perform the next production using (1) until all the genotypes are mapped or there are not more nonterminals in the phenotype.

The Bin Packing Problem (BPP) has been widely studied because of its many Industrial Applications, like wood and glass cutting, packing in transportation and warehousing [12], and job scheduling on uniform processors [13, 14]. This is an NP-Hard Problem [1] and due to its complexity many heuristics have been developed attempting to give an approximation [15–19]. Some metaheuristics have also been applied to try to obtain better results than those obtained by heuristics [20–22]. Some exact algorithms have been developed [23–25]; however, given the nature of the problem the time reported by these algorithms grows up and depending on the instance the time may grow up exponentially.

The contribution of this paper is to propose a generic methodology to generate heuristics using GE with search strategies. It has been shown that is possible to use this methodology to generate BPP heuristics by using PESO and PSO as search strategies; it was also shown that the heuristics generated with the proposed Grammar have better performance than the BPP's classical heuristics, which were designed by an expert in Operational Research. Those results were obtained by comparing the results obtained by the GE and the BPP heuristics by means of Friedman nonparametric test [26].

The GE is described in Section 2, including the PSO and PESO. Section 3 describes the Bin Packing Problem, the state-of-the-art heuristics, the instances used, and the fitness function. We describe the experiments performed in Section 4. Finally, general conclusions about the present work are presented in Section 5, including future perspectives of this work.

## 2. Grammatical Evolution

Grammatical Evolution (*GE*) [7] is a grammar-based form of Genetic Programming (*GP*) [27]. GE joins the principles of molecular biology, which are used by GP, and the power of formal grammars. Unlike GP, GE adopts a population of lineal genotypic integer strings, or binary strings, witch are transformed into functional phenotypic through a genotype-to-phenotype mapping process [28]; this process is also known as *Indirect Representation* [29]. The genotype strings evolve with no knowledge of their phenotypic equivalent, only using the fitness measure.

The transformation is governed through a Backus Naur Form grammar (*BNF*), which is made up of the tuple $N, T, P, S$, where $N$ is the set of all nonterminal symbols, $T$ is the set of terminals, $P$ is the set of production rules that map $N \rightarrow T$, and $S$ is the initial start symbol where $S \in N$. There are a number of production rules that can be applied to a nonterminal; an "|" (or) symbol separates the options.

Even though the GE uses the Genetic Algorithm (*GA*) [7, 28, 30] as a search strategy it is possible to use another search strategy like the Particle Swarm Optimization, called Grammatical Swarm (*GS*) [8].

In GE each individual is mapped into a program using the BNF, using (1) proposed in [28] to choose the next production based on the nonterminal symbol. An example of the mapping process employed by GE is shown in Figure 1. Consider

$$\text{Rule} = c\%r, \tag{1}$$

where $c$ is the codon value and $r$ is the number of production rules available for the current nonterminal.

The GE can use different search strategies; our proposed model is shown in Figure 2. This model includes the problem instance and the search strategy as an input. In [28] the search strategy is part of the process; however it can be seen as an additional element that can be chosen to work with GE. The GE will generate a solution through the search strategy selected and it will be evaluated in the objective function using the problem instance.

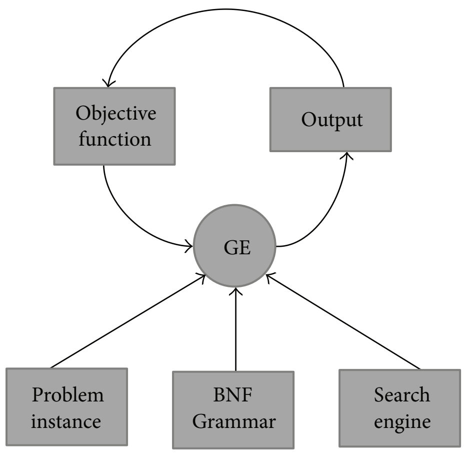

Figure 2: GE's methodology used in the present work: the presented methodology can be used with different search strategies.

*2.1. Particle Swarm Optimization.* Particle Swarm Optimization (PSO) [31–35] is a metaheuristic bioinspired in the flock of birds or school of fish. It was developed by Kennedy and Eberthart based on a concept called social metaphor. This metaheuristic simulates a society where all individuals contribute with their knowledge to obtain a better solution. There are three factors that influence the change of status or behavior of an individual.

(i) The knowledge of the environment or adaptation: it is related to the importance given to the experience of the individual.

(ii) His experience or local memory: it is related to the importance given to the best result found by the individual.

(iii) The experience of their neighbors or global memory: this is related to how important is the best result obtained by their neighbors or other individuals.

In this metaheuristic each individual is considered as a particle and moves through a multidimensional space that represents the social space; the search space depends on the dimension of space which in turn depends on the variables used to represent the problem.

For the update of each particle we use the velocity vector which tells how fast the particle will move in each of the dimensions; the method for updating the speed of PSO is given by (2), and its position is updated by (3). Algorithm 1 shows the complete PSO algorithm:

$$v_i = wv_i + \phi_1 \left(x_i - B_{\text{global}}\right) + \phi_2 \left(x_i - B_{\text{local}}\right), \tag{2}$$

$$x_i = x_i + v_i, \tag{3}$$

where

(i) $v_i$ is the velocity of the $i$th particle,

(ii) $w$ is adjustment factor to the environment,

(iii) $\phi_1$ is the memory coefficient in the neighborhood,

(iv) $\phi_2$ is the coefficient memory,

(v) $x_i$ is the position of the $i$th particle,

(vi) $B_{\text{global}}$ is the best position found so far by all particles,

(vii) $B_{\text{local}}$ is the best position found by the $i$th particle.

*2.2. Particle Evolutionary Swarm Optimization.* Particle Evolutionary Swarm Optimization (*PESO*) [36–38] is based on PSO but introduces two perturbations in order to avoid two problems observed in PSO [39]:

(i) premature convergence,

(ii) poor diversity.

Algorithm 2 shows the PESO Algorithm with two perturbations, Algorithms 3 and 4. The C-Perturbation has the advantage of keeping the self-organization potential of the flock as no separate probability distribution needs to be computed; meanwhile the M-Perturbation helps keeping diversity into the population.

## 3. Bin Packing Problem

The Bin Packing Problem (*BPP*) [40] can be described as follows: given $n$ items that need to be packed in the lowest possible number of bins, each item has a weight $w_j$, where $j$ is the element; the max capacity of the bins $c$ is also available. The objective is to minimize the bins used to pack all the items, given that each item is assigned only to one bin, and the sum of all the items in the bin can not exceed the bin's size.

This problem has been widely studied, including the following:

(i) proposing new theorems [41, 42],

(ii) developing new heuristic algorithms based on Operational Research concepts [18, 43],

(iii) characterizing the problem instances [44–46],

(iv) implementing metaheuristics [20, 47–49].

This problem has been shown to be an NP-Hard optimization problem [1]. A mathematical definition of the BPP is as follows:

Minimize

$$z = \sum_{i=1}^{n} y_i, \tag{4}$$

subject to the following constraints and conditions:

$$\begin{gathered}
\sum_{j=1}^{n} w_j x_{ij} \le c y_i, \quad i \in N = \{1, \dots, n\}, \\
\sum_{i=1}^{n} x_{ij} = 1, \quad j \in N, \\
y_i \in \{0, 1\}, \quad i \in N, \\
x_{ij} \in \{0, 1\}, \quad i \in N, \ j \in N,
\end{gathered} \tag{5}$$

```
Require: $w$ adaptation to environment coefficient, $\phi_1$ neighborhood memory
      coefficient, $\phi_2$ memory coefficient, $n$ swarm size.
(1)  Start the swarm particles.
(2)  Start the velocity vector for each particle in the swarm.
(3)  while stopping criterion not met do
(4)    for $i = 1$ to $n$ do
(5)      If the $i$-particle's fitness is better than the local best then replace the
         local best with the $i$-particle.
(6)      If the $i$-particle's fitness is better than the global best then replace the
         global best with the $i$-particle.
(7)      Update the velocity vector by (2).
(8)      Update the particle's position with the velocity vector by (3).
(9)    end for
(10) end while
```

Algorithm 1: PSO Algorithm.

```
Require: $w$ adaptation to environment coefficient, $\phi_1$ neighborhood memory
      coefficient, $\phi_2$ memory coefficient, $n$ swarm size.
(1)  Start the swarm particles.
(2)  Start the velocity vector for each particle in the swarm.
(3)  while stopping criterion not met do
(4)    for $i = 1$ to $n$ do
(5)      If the $i$-particle's fitness is better than the local best then replace the
         local best with the $i$-particle.
(6)      If the $i$-particle's fitness is better than the global best then replace the
         global best with the $i$-particle.
(7)      Update the velocity vector by (2).
(8)      Update the particle's position with the velocity vector by (3).
(9)      Apply the C-Perturbation
(10)     Apply the M-Perturbation
(11)   end for
(12) end while
```

Algorithm 2: PESO Algorithm.

```
(1) for all Particles do
(2)   Generate $r$ uniformly between 0 and 1.
(3)   Generate $p1$, $p2$ and $p3$ as random numbers between 1 and the number
      of particles.
(4)   Generate the $i$-new particle using the following equation and applying it
      to each particle dimension: $new_i = p1 + r(p2 - p3)$.
(5) end for
(6) for all Particles do
(7)   If the $i$-new particle is better that the $i$-particle then replace the $i$-particle
      with the $i$-new particle.
(8) end for
```

Algorithm 3: C-Perturbation.

where

(i) $w_j$ is weight of the $j$ item,

(ii) $y_i$ is binary variable that shows if the bin $i$ has items,

(iii) $x_{ij}$ indicates whether the $j$ item is into the $i$ bin,

(iv) $n$ is number of available bins,

(v) $c$ is capacity of each bin.

The algorithms for the BPP instances can be classified as *online* or *offline* [46]. We have algorithms considered *online* if

```
(1)  for all Particles do
(2)      for all Dimension do
(3)          Generate $r$ uniformly between 0 and 1.
(4)          if $r \leq 1/dimension$ then
(5)              $new_{id} = random(LowerBound, UpperBound)$
(6)          else
(7)              $new_{id} = Particle_d$
(8)          end if
(9)      end for
(10) end for
(11) for all Particles do
(12)     If the $i$-new particle is better that the $i$-particle then replace the $i$-particle
         with the $i$-new particle.
(13) end for
```

Algorithm 4: M-Perturbation.

we do not know the items before starting the packing process and *offline* if we know all the items before starting. In this research we worked with both algorithms.

*3.1. Tests Instances.* Beasley [50] proposed a collection of test data sets, known as *OR-Library* and maintained by the Beasley University, which were studied by Falkenauer [21]. This collection contains a variety of test data sets for a variety of Operational Research problems, including the BPP in several dimensions. For the one-dimensional BPP case the collection contains eight data sets that can be classified in two classes.

(i) *Unifor.* The data sets from binpack1 to binpack4 consist of items of sizes uniformly distributed in $(20, 100)$ to be packed into bins of size 150. The number of bins in the current known solution was found by [21].

(ii) *Triplets.* The data sets from binpack5 to binpack8 consist of items from $(24, 50)$ to be packed into bins of size 100. The number of bins can be obtained dividing the size of the data set by three.

Scholl et al. [23] proposed another collection of data sets; only 1184 problems were solved optimally. Alvim et al. [51] reported the optimal solutions for the remaining 26 problems. The collection contains three data sets.

(i) *Set 1.* It has 720 instances with items drawn from a uniform distribution on three intervals $[1, 100]$, $[20, 100]$, and $[30, 100]$. The bin capacity is $C = 100$, 120, and 150 and $n = 50, 100, 200$, and 500.

(ii) *Set 2.* It has 480 instances with $C = 1000$ and $n = 50$, 100, 200, and 500. Each bin has an average of 3–9 items.

(iii) *Set 3.* It has 10 instances with $C = 100{,}000$, $n = 200$, and items are drawn from a uniform distribution on $[20000, 35000]$. Set 3 is considered the most difficult of the three sets.

*3.2. Classic Heuristics.* Heuristics have been used to solve the BPP, obtaining good results. Reference [18] shows the following heuristics as Classical Heuristics; these heuristics can be used as *online heuristics* if the items need to be packed as they come in or *offline heuristics* if the items can be sorted before starting the packing process.

(i) *Best Fit* [17] puts the piece in the fullest bin that has room for it and opens a new bin if the piece does not fit in any existing bin.

(ii) *Worst Fit* [18] puts the piece in the emptiest bin that has room for it and opens a new bin if the piece does not fit in any existing bin.

(iii) *Almost Worst Fit* [18] puts the piece in the second emptiest bin if that bin has room for it and opens a new bin if the piece does not fit in any open bin.

(iv) *Next Fit* [15] puts the piece in the right-most bin and opens a new bin if there is not enough room for it.

(v) *First Fit* [15] puts the piece in the left-most bin that has room for it and opens a new bin if it does not fit in any open bin.

Even though there are some heuristics having better performance than the heuristics shown in the present section [16, 19, 42, 52, 53], such heuristics have been the result of research of lower and upper bounds to determine the minimal number of bins.

*3.3. Fitness Measure.* There are many *Fitness Measures* used to discern the results obtained by heuristics and metaheuristics algorithms. In [54] two fitness measures are shown: the first measure (see (6)) tries to find the difference between the used bins and the theorical upper bound on the bins needed; the second (see (7)) was proposed in [47] and rewards full or

```
Require: SS search strategy, FF Fitness Function, BNF-G Grammar, IS
      Instances Set.
(1)  for all Instance Set into Instances Set do
(2)     Select randomly an Instance from the Instance Set
(3)     Start an initial population.
(4)     while stopping criterion not met do
(5)        Apply the mapping process using the Grammar BNF-G, as seen in
           the Figure 1, to obtain an heuristic by each element into the population.
(6)        Calculate the fitness value, using FF, for each element into the population
           applying the heuristic generated to the instance selected.
(7)        Apply the search strategy to optimize the elements into the population.
(8)     end while
(9)     Apply the found heuristic to all instances from the Instance Set.
(10) end for
(11) return Heuristic for each instance set.
```

Algorithm 5: Proposed approach.

almost full bins; the objective is to fill each bin, minimizing the free space:

$$\text{Fitness} = B - \frac{\sum_{i=1}^{n} w_i}{C}, \tag{6}$$

$$\text{Fitness} = 1 - \left( \frac{\sum_{i=1}^{n} \left( \left( \sum_{j=1}^{m} w_j x_{ij} \right) / C \right)^2}{n} \right), \tag{7}$$

where

(i) $B$ is number of bins used,

(ii) $n$ is number of containers,

(iii) $m$ is number of pieces,

(iv) $w_j$ is $j$th's piece size,

(v) consider

$$x_{ij} = \begin{cases} 1 & \text{if the piece } j \text{ is in the container } i \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

(vi) $C$ is bin capacity.

## 4. Grammar Design and Testing

Algorithm 5 shows the proposed approach; this approach allows the use of different fitness functions and search strategies to generate heuristics automatically.

To improve the Bin Packing Heuristics it was necessary to design a grammar that represents the Bin Packing Problem. In [55] Grammar 1 is shown to be based on heuristic elements taken by [6]; however the results obtained in [1] give 10% of solutions that can not be applied to the instance and for this reason this approach does not need to be included to be compared against the results obtained.

That Grammar has been improved in the Grammar 2 [56] to obtain similar results to those obtained by the BestFit heuristic. However this grammar cannot be applied to Bin Packing offline Problems because it does not sort pieces. Grammar 3 is proposed to improve the results obtained by Grammar 2, given that it can generate heuristics online and offline:

Table 1: PESO and PSO parameters.

| Parameter | Value |
|---|---|
| Population size | 50 |
| $w$ | 1.0 |
| $\phi_1$ | 0.8 |
| $\phi_2$ | 0.5 |
| Function calls | 1500 |

$$\begin{gathered} \langle\text{inicio}\rangle \vDash (\langle\text{expr}\rangle) <= (\langle\text{expr}\rangle) \\ \langle\text{expr}\rangle \vDash (\langle\text{expr}\rangle \langle\text{op}\rangle \langle\text{expr}\rangle) \mid \langle\text{var}\rangle \mid \text{abs}(\langle\text{expr2}\rangle) \\ \langle\text{expr2}\rangle \vDash (\langle\text{expr2}\rangle \langle\text{op}\rangle \langle\text{expr2}\rangle) \mid \langle\text{var}\rangle \\ \langle\text{var}\rangle \vDash F \mid C \mid S \\ \langle\text{op}\rangle \vDash + \mid * \mid - \mid /. \end{gathered} \tag{9}$$

*Grammar 1.* Grammar based on FirstFit Heuristic was proposed in [55] and we use the Heuristic Components shown in [57]

$$\begin{gathered} \langle\text{inicio}\rangle \vDash \langle\text{exprs}\rangle \cdot (\langle\text{expr}\rangle) <= (\langle\text{expr}\rangle) \\ \langle\text{exprs}\rangle \vDash \text{Sort}(\langle\text{exprk}\rangle, \langle\text{order}\rangle) \mid \lambda \\ \langle\text{exprk}\rangle \vDash \text{Bin} \mid \text{Content} \\ \langle\text{order}\rangle \vDash \text{Asc} \mid \text{Des} \end{gathered}$$

Table 2: Results obtained by each heuristic using (2).

| Instance | Fitness function | MTP algorithm | Online heuristics | | | | | Offline heuristics | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | BestFit | FirstFit | NextFit | WorstFit | Almost WorstFit | BestFit | FirstFit | NextFit | Almost WorstFit | WorstFit |
| bin1data | Equation (7) | 64.307365 | 316.106050 | 316.106020 | 316.106020 | 316.106000 | 321.880250 | 68.090770 | 68.167305 | 314.939820 | 86.119570 | 76.842740 |
| | Bins used | 78378 | 101097 | 101097 | 101097 | 101097 | 101705 | 78660 | 78661 | 101097 | 79314 | 78843 |
| | Leftover bins | — | 22719 | 22719 | 22719 | 22719 | 23327 | 282 | 283 | 22719 | 465 | 936 |
| bin2data | Equation (7) | 24.725332 | 93.025230 | 93.388504 | 109.447590 | 97.613075 | 115.927666 | 44.560112 | 44.561455 | 110.053760 | 67.597020 | 47.655037 |
| | Bins used | 20246 | 23085 | 23094 | 23518 | 23158 | 23580 | 20994 | 20994 | 23615 | 21446 | 21030 |
| | Leftover bins | — | 2839 | 2848 | 3272 | 2912 | 3334 | 748 | 748 | 3369 | 784 | 1200 |
| bin3data | Equation (7) | 0.390077 | 1.885825 | 1.885825 | 2.586886 | 2.094760 | 2.253316 | 1.390289 | 1.390289 | 2.699653 | 1.518808 | 1.397016 |
| | Bins used | 562 | 613 | 613 | 642 | 622 | 631 | 596 | 596 | 650 | 603 | 596 |
| | Leftover bins | — | 51 | 51 | 80 | 60 | 69 | 34 | 34 | 88 | 34 | 41 |
| binpack1 | Equation (7) | 0.421557 | 2.425894 | 2.604965 | 7.941016 | 5.089604 | 5.365787 | 0.913949 | 0.914034 | 9.504668 | 1.479180 | 1.233465 |
| | Bins used | 981 | 1038 | 1044 | 1279 | 1131 | 1147 | 995 | 995 | 1372 | 1016 | 1003 |
| | Leftover bins | — | 57 | 63 | 298 | 150 | 166 | 14 | 14 | 391 | 22 | 35 |
| binpack2 | Equation (7) | 0.180042 | 2.259154 | 2.396851 | 7.989118 | 4.916397 | 4.992059 | 0.705970 | 0.706004 | 9.459408 | 1.043219 | 0.846457 |
| | Bins used | 2032 | 2154 | 2162 | 2669 | 2342 | 2353 | 2062 | 2062 | 2851 | 2087 | 2068 |
| | Leftover bins | — | 122 | 130 | 637 | 310 | 321 | 30 | 30 | 819 | 36 | 55 |
| binpack3 | Equation (7) | 0.097509 | 2.014334 | 2.133326 | 7.994689 | 4.725924 | 4.769628 | 0.591541 | 0.591543 | 9.412333 | 0.746746 | 0.666438 |
| | Bins used | 4024 | 4240 | 4255 | 5300 | 4614 | 4626 | 4078 | 4078 | 5647 | 4100 | 4085 |
| | Leftover bins | — | 216 | 231 | 1276 | 590 | 602 | 54 | 54 | 1623 | 61 | 76 |
| binpack4 | Equation (7) | 0.047260 | 1.838669 | 1.935710 | 7.961214 | 4.597339 | 4.622161 | 0.495512 | 0.495522 | 9.404397 | 0.622414 | 0.572282 |
| | Bins used | 8011 | 8407 | 8430 | 10548 | 9154 | 9167 | 8108 | 8108 | 11253 | 8141 | 8123 |
| | Leftover bins | — | 396 | 419 | 2537 | 1143 | 1156 | 97 | 97 | 3242 | 112 | 130 |
| binpack5 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.324353 | 4.830872 | 4.830872 | 6.420693 | 5.453772 | 4.848756 |
| | Bins used | 400 | 400 | 400 | 400 | 400 | 420 | 464 | 464 | 491 | 479 | 464 |
| | Leftover bins | — | 0 | 0 | 0 | 0 | 20 | 64 | 64 | 91 | 64 | 79 |
| binpack6 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.682040 | 4.553619 | 4.553619 | 6.197923 | 4.994128 | 4.557339 |
| | Bins used | 800 | 800 | 800 | 800 | 800 | 820 | 916 | 916 | 971 | 936 | 916 |
| | Leftover bins | — | 0 | 0 | 0 | 0 | 20 | 116 | 116 | 171 | 116 | 136 |
| binpack7 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.334901 | 4.556952 | 4.556953 | 6.078939 | 4.776067 | 4.558652 |
| | Bins used | 1660 | 1660 | 1660 | 1660 | 1660 | 1680 | 1900 | 1900 | 2002 | 1919 | 1900 |
| | Leftover bins | — | 0 | 0 | 0 | 0 | 20 | 240 | 240 | 342 | 240 | 259 |
| binpack8 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.167147 | 4.400591 | 4.400588 | 6.059548 | 4.527734 | 4.400708 |
| | Bins used | 3340 | 3340 | 3340 | 3340 | 3340 | 3360 | 3801 | 3801 | 4024 | 3823 | 3801 |
| | Leftover bins | — | 0 | 0 | 0 | 0 | 20 | 461 | 461 | 684 | 461 | 483 |
| hard28 | Equation (7) | 0.167276 | 0.655480 | 0.655350 | 13.133352 | 1.547310 | 1.927801 | 0.655480 | 0.655350 | 13.133352 | 1.927801 | 1.547310 |
| | Bins used | 1972 | 1995 | 1995 | 2755 | 2024 | 2050 | 1995 | 1995 | 2755 | 2050 | 2024 |
| | Leftover bins | — | 23 | 23 | 783 | 52 | 78 | 23 | 23 | 783 | 52 | 78 |
| | Remaining bins | | 26423 | 26484 | 31602 | 27936 | 29133 | 2163 | 2164 | 34322 | 2447 | 3508 |

Table 3: Example of heuristics obtained for each instance set using Grammar 3.

| Instance | Heuristic generated |
|---|---|
| dasaset1 | Sort(Elements, Des) · Sort(Bin, Des) · $((F + S)) \leq (\text{abs}(C))$ |
| dataset2 | Sort(Elements, Des) · Sort(Cont, Des) · $(\text{abs}(S)) \leq (\text{abs}((C - F)))$ |
| dataset3 | Sort(Elements, Des) · Sort(Cont, Des) · $(S) \leq ((C - F))$ |
| binpack1 | Sort(Content, Des) · $(\text{abs}(F)) \leq ((C- \text{abs}(S)))$ |
| binpack2 | Sort(Content, Des) · $((F + S)) \leq (C)$ |
| binpack3 | Sort(Content, Des) · $(F) \leq (\text{abs}((C - S)))$ |
| binpack4 | Sort(Content, Asc) · $(S) \leq ((C - F))$ |
| binpack5 | $((S + F)) \leq (C)$ |
| binpack6 | $(F) \leq ((\text{abs}(C) -S))$ |
| binpack7 | $(\text{abs}(F)) \leq (\text{abs}((S - C)))$ |
| binpack8 | $(\text{abs}((S + F))) \leq (C)$ |
| hard28 | Sort(Cont, Des) · $(F) \leq (\text{abs}((C - S)))$ |

$$\begin{gathered}\langle\text{expr}\rangle \vDash (\langle\text{expr}\rangle\ \langle\text{op}\rangle\ \langle\text{expr}\rangle) \mid \langle\text{var}\rangle \mid \text{abs}\,(\langle\text{expr2}\rangle)\\ \langle\text{expr2}\rangle \vDash (\langle\text{expr2}\rangle\ \langle\text{op}\rangle\ \langle\text{expr2}\rangle) \mid \langle\text{var}\rangle\\ \langle\text{var}\rangle \vDash F \mid C \mid S\\ \langle\text{op}\rangle \vDash + \mid * \mid - \mid /.\end{gathered} \tag{10}$$

*Grammar 2.* Grammar proposed in [56] was based on Best-Fist Heuristic:

$$\begin{gathered}\langle\text{begin}\rangle \vDash \langle\text{exproff}\rangle\ \langle\text{exprsort}\rangle\ (\langle\text{expr}\rangle) <= (\langle\text{expr}\rangle)\\ \langle\text{Exproff}\rangle \vDash \text{Sort}\,(\text{Elements}, \langle\text{order}\rangle) \mid \lambda\\ \langle\text{exprsort}\rangle \vDash \text{Sort}\,(\langle\text{exprkind}\rangle, \langle\text{order}\rangle) \mid \lambda\\ \langle\text{exprkind}\rangle \vDash \text{Bins} \mid \text{SumElements}\\ \langle\text{order}\rangle \vDash \text{Asc} \mid \text{Des} \qquad (11)\\ \langle\text{expr}\rangle \vDash (\langle\text{expr}\rangle\ \langle\text{op}\rangle\ \langle\text{expr}\rangle) \mid \langle\text{var}\rangle \mid \text{abs}\,(\langle\text{expr2}\rangle)\\ \langle\text{expr2}\rangle \vDash (\langle\text{expr2}\rangle\ \langle\text{op}\rangle\ \langle\text{expr2}\rangle) \mid \langle\text{var}\rangle\\ \langle\text{var}\rangle \vDash F \mid C \mid S\\ \langle\text{op}\rangle \vDash + \mid * \mid - \mid /.\end{gathered}$$

*Grammar 3.* Grammar proposal to generate heuristics online and offline is based on Grammar 2 , where

(i) $S$ is size of the current piece,

(ii) $C$ is bin capacity,

(iii) $F$ is sum of the pieces already in the bin,

(iv) Elements sorts the elements,

(v) Bin sorts the bins based on the bin number,

(vi) Cont sorts the bins based on the bin contents,

(vii) Asc sorts in ascending order,

(viii) Des sorts in descending order.

In order to generate the heuristics Grammar 3 was used. The search strategies applied to the GE were PESO and PSO. The number of function calls was taken from [56], where it was explained that this number is only 10% from the number of function calls used by [6]. To obtain the parameters shown in Table 1 a fine-tuning process was applied based on Covering Arrays (CA) [58]; in this case the CA was generated using the Covering Array Library (CAS) [59] from The National Institute of Standards and Technology (NIST) (http://csrc.nist.gov/groups/SNS/acts/index.html).

In order to generate the heuristics, one instance from each set was used. Once the heuristic was obtained for each instance set, it was applied to all the sets to obtain the heuristic's fitness. The instance sets used were detailed in Section 3.1. 33 experiments were performed independently and the median was used to compare the results against those obtained with the heuristics described in Section 3. The comparison was implemented through the nonparametric test of Friedman [26, 60]; this nonparametric test used a post hoc analysis to discern the performance between the experiments and gives a ranking of them.

The method to apply the heuristics generated by Grammar 3 for an instance set is described below.

(i) For each instance in the instance set the generated heuristic will be applied.

(ii) The generated heuristic has the option to sort the items before starting the packing process, to treat the instances like offline instances.

(iii) The next part of the generated heuristic says how to sort the bins; many heuristics require sorting the bins before packing an item.

(iv) The last part, the inequality, determines the rule to pack an item.

Sometimes the generated heuristic does not have items ordered and it makes the heuristic work like an online heuristic. If it does not have the bins ordered all the items will be packed into the bins in the order they were created.

Table 4: Results obtained by each heuristic over the instance set.

| Instance | Fitness function | PSO |  |  | PESO |  |  |
|---|---|---|---|---|---|---|---|
|  |  | Grammar 1 | Grammar 2 | Grammar 3 | Grammar 1 | Grammar 2 | Grammar 3 |
| bin1data | Equation (7) | 316.106020 | 316.106050 | 68.167305 | 316.106020 | 316.106050 | 68.090770 |
|  | Bins used | 101097 | 101097 | 101097 | 101097 | 101097 | 78660 |
|  | Leftover bins | 22719 | 22719 | 22719 | 22719 | 22719 | 282 |
| bin2data | Equation (7) | 93.388510 | 93.025230 | 44.561455 | 93.388510 | 93.007030 | 44.560112 |
|  | Bins used | 23094 | 23156 | 23583 | 23097 | 23156 | 20994 |
|  | Leftover bins | 2848 | 2910 | 3337 | 2851 | 2910 | 748 |
| bin3data | Equation (7) | 1.885825 | 1.885825 | 1.390289 | 1.885825 | 1.885825 | 1.390289 |
|  | Bins used | 613 | 622 | 650 | 613 | 622 | 596 |
|  | Leftover bins | 51 | 60 | 88 | 51 | 60 | 34 |
| binpack1 | Equation (7) | 2.604965 | 2.425894 | 0.914034 | 2.604965 | 2.425894 | 0.913949 |
|  | Bins used | 1044 | 1131 | 1372 | 1044 | 1131 | 995 |
|  | Leftover bins | 63 | 150 | 391 | 63 | 150 | 14 |
| binpack2 | Equation (7) | 2.396851 | 2.259154 | 0.706004 | 2.396851 | 2.259154 | 0.705970 |
|  | Bins used | 2162 | 2342 | 2851 | 2162 | 2342 | 2062 |
|  | Leftover bins | 130 | 310 | 819 | 130 | 310 | 30 |
| binpack3 | Equation (7) | 2.133326 | 2.014334 | 0.591543 | 2.133326 | 2.014334 | 0.591541 |
|  | Bins used | 4255 | 4614 | 5647 | 4255 | 4614 | 4078 |
|  | Leftover bins | 231 | 590 | 1623 | 231 | 590 | 54 |
| binpack4 | Equation (7) | 1.935710 | 1.838669 | 0.495522 | 1.935710 | 1.838669 | 0.495512 |
|  | Bins used | 8430 | 9154 | 11253 | 8430 | 9154 | 8108 |
|  | Leftover bins | 419 | 1143 | 3242 | 419 | 1143 | 97 |
| binpack5 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
|  | Bins used | 400 | 400 | 400 | 400 | 400 | 400 |
|  | Leftover bins | 0 | 0 | 0 | 0 | 0 | 0 |
| binpack6 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
|  | Bins used | 800 | 800 | 800 | 800 | 800 | 800 |
|  | Leftover bins | 0 | 0 | 0 | 0 | 0 | 0 |
| binpack7 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
|  | Bins used | 1660 | 1660 | 1660 | 1660 | 1660 | 1660 |
|  | Leftover bins | 0 | 0 | 0 | 0 | 0 | 0 |
| binpack8 | Equation (7) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
|  | Bins used | 3340 | 3340 | 3340 | 3340 | 3340 | 3340 |
|  | Leftover bins | 0 | 0 | 0 | 0 | 0 | 0 |
| hard28 | Equation (7) | 0.655350 | 0.655480 | 0.655350 | 0.655350 | 0.655480 | 0.655480 |
|  | Bins used | 1995 | 2024 | 2755 | 1995 | 2024 | 1995 |
|  | Leftover bins | 23 | 52 | 783 | 23 | 52 | 23 |
| Remaining bins |  | 26484 | 27934 | 33002 | 26487 | 27934 | 1282 |

## 5. Results

In Table 2 the results obtained with online and offline heuristics (described in Section 3.2) are shown. Results obtained by an exact algorithm were included, the MTP algorithm [40], and results from the fitness function from Section 3.3 are shown as well with the number of bins used. A row was added where the difference of containers regarding the optimal is shown. These results were obtained by applying the heuristics to each instance; all the results from an instance set were added.

Table 3 shows examples of heuristics generated using the proposed Grammar with GE for each instance set; some heuristics can be reduced but this is not part of the present work.

The results obtained by the PSO and PESO with the Grammars are shown in Table 4; these results are the median from 33 individual experiments. Using the results obtained by the heuristics and the GE with PESO and PSO the Friedman nonparametric test was performed to discern the results. The value obtained by the Friedman nonparametric test is 85.789215 and the $P$ value 6.763090E-11; this means that the

Table 5: Rankings of the algorithms.

| Algorithm | Ranking Friedman |
|---|---|
| Exact | 2.666667 |
| PESO-Grammar 3 | 4.375000 |
| PSO-Grammar 3 | 4.791667 |
| BestFit-Offline | 6.916667 |
| FirstFit-Offline | 7.250000 |
| PESO-Grammar 2 | 8.666667 |
| BestFit | 8.791667 |
| PSO-Grammar 2 | 8.791667 |
| FirstFit | 8.958333 |
| PESO-Grammar 1 | 9.083333 |
| PSO-Grammar 1 | 9.083333 |
| WorstFit-Offline | 9.541667 |
| AlmostWorstFit-Offline | 10.625000 |
| WorstFit | 10.791667 |
| NextFit | 12.250000 |
| AlmostWorstFit | 14.291667 |
| NextFit-Offline | 16.125000 |

tested heuristics have different performance. Due to this it was necessary to apply a post hoc procedure to obtain the Heuristics Ranking shown in Table 5.

Both Tables 2 and 4 have an extra row at the bottom with the total remaining bins. The results obtained by PESO using Grammar 3 show that this heuristic which has been deployed automatically has less bins than the other classic heuristics.

## 6. Conclusions and Future Works

In the present work a Grammar was proposed to generate online and offline heuristics in order to improve heuristics generated by other grammars and by humans. It also was proposed using PESO as a search strategy based on Swarm Intelligence to avoid the problems observed in PSO.

Through the results obtained in Section 5, it was concluded that it is possible to generate good heuristics with the proposed Grammar. Additionally it can be seen that the quality of these heuristics strongly depends on the grammar used to evolve.

The grammar proposed in the present work shows that is possible to generate heuristics with better performance that the well-known BestFit, FirstFit, NextFit, WorstFit, and Almost WorstFit heuristics from Section 3.2 regardless of heuristics being online or offline. While the heuristics are designed to work with all the instances sets, the GE adjusts heuristics automatically to work with one instance set and it makes it possible for GE to generate offline or online heuristics. The GE can generate as many heuristics as instances sets that have been working and try to adapt the best heuristic that can be generated with the used Grammar.

The results obtained by PESO are better than those obtained by PSO by using Grammars 2 and 3, but with Grammar 1 PESO and PSO have the same performance.

The current investigation is based on the one-dimensional bin packing problem but this methodology can be used to solve other problems, due to the generality of the approach. It is necessary to apply heuristic generation to other problems and investigate if the GE with PESO as search strategy gives better results than the GP or GE with other search strategies.

It will be necessary to find a methodology to choose the instance or instances for the training process as well as to determine if the instances are the same or to classify the instances in groups with the same features to generate only one heuristic by group.

It will also be necessary to research other metaheuristics that do not need the parameter tuning because the metaheuristics shown in the present paper were tuned using Covering Arrays.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors want to thank to the Instituto Tecnológico de León (ITL) for the support provided for this research. Additionally the authors want to aknowledge the generous support from the Consejo Nacional de Ciencia y Tecnológia (CONACyT) from Mexico for this research project.

## References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.

[2] E. A. Feigenbaum and J. Feldman, *Computers and Thought*, AAAI Press, 1963.

[3] M. H. J. Romanycia and F. J. Pelletier, "What is a heuristic?" *Computa tional Intelligence*, vol. 1, no. 1, pp. 47–58, 1985.

[4] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[5] J. R. Koza, " Hierarchical genetic algorithms operating on populations of computer programs," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 768–774, San Mateo, Calif, USA, 1989.

[6] E. K. Burke, M. Hyde, and G. Kendall, "Evolving bin packing heuristics with genetic programming," in *Parallel Problem Solving from Nature-PPSN IX*, T. Runarsson, H.-G. Beyer, J. Merelo-Guervós, L. Whitley, and X. Yao, Eds., vol. 4193 of *Lecture Notes in Comput er Science*, pp. 860–869, Springer, Berlin, Germany, 2006.

[7] C. Ryan, J. Collins, and M. O'Neill, "Grammatical evolution: evolving programs for an arbitrary language," in *Proceedings of the 1st European Workshop on Genetic Programming*, vol. 1391 of *Lecture Notes in Com puter Science*, pp. 83–95, Springer, 1998.

[8] M. O'Neill and A. Brabazon, "Grammatical differential evolution," in *Proceedings of the International Conference on Artificial Intelligence (ICAI '06)*, CSEA Press, Las Vegas, Nev, USA, 2006.

[9] M. O'Neill and A. Brabazon, "Grammatical swarm: the generation of programs by social programming," *Natural Computing*, vol. 5, no. 4, pp. 443–462, 2006.

[10] J. Togelius, R. de Nardi, and A. Moraglio, "Geometric PSO + GP = particle swarm programming," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 3594–3600, Hong Kong, June 2008.

[11] A. Moraglio and S. Silva, "Geometric di ff ere ntial evolution on the space of genetic programs," in *Genetic Programming*, A. Esparcia-Alcazar, A. Ekart, S. Silva, S. Dignum, and A. Uyar, Eds., vol. 6021 of *Lecture Notes in Computer Science*, pp. 171–183, Springer, Berlin, Germany, 2010.

[12] A. Lodi, S. Martello, and D. Vigo, "Recent advances on two-dimensional bin packing problems," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 379–396, 2002.

[13] H. van de Vel and S. Shijie, "Application of the bin-packing technique to job scheduling on uniform processors," *The Journal of the Operational Research Society*, vol. 42, no. 2, pp. 169–172, 1991.

[14] B. T. Han, G. Diehr, and J. S. Cook, "Multiple-type, two-dimensional bin packing problems: applications and algorithms," *Annals of Operations Research*, vol. 50, no. 1, pp. 239–261, 1994.

[15] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, pp. 299–325, 1974.

[16] A. C. C. Yao, "New algorithms for bin packing," *Journal of the Association for Computing Machinery*, vol. 27, no. 2, pp. 207–227, 1980.

[17] W. T. Rhee and M. Talagrand, "On-line bin packing with items of random size," *Mathematics of Operations Research*, vol. 18, no. 2, pp. 438–445, 1993.

[18] E. Coffman Jr., G. Galambos, S. Martello, and D. Vigo, *Bin Packing Approximation Algorithms: Combinatorial Analysis*, Kluwer Academic Publishers, 1998.

[19] K. Fleszar and K. S. Hindi, "New heuristics for one-dimensional bin-packing," *Computers and Operations Research*, vol. 29, no. 7, pp. 821–839, 2002.

[20] T. Kämpke, "Simulated annealing: use of a new tool in bin packing," *Annals of Operations Research*, vol. 16, no. 1, pp. 327–332, 1988.

[21] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, pp. 5–30, 1996.

[22] A. Ponee-Pérez, A. Pérez-Garcia, and V. Ayala-Ramirez, "Bin-packing using genetic algorithms," in *Proceedings of the 15th International Conference on Electronics, Communications and Computers (CONIELECOMP '05)*, pp. 311–314, IEEE Computer Society, Los Alamitos, Calif, USA, March 2005.

[23] A. Scholl, R. Klein, and C. Jürgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers and Operations Research*, vol. 24, no. 7, pp. 627–645, 1997.

[24] J. M. V. de Carvalho, "Exact solution of bin-packing problems using column generation and branch-and-bound," *Annals of Operations Research*, vol. 86, pp. 629–659, 1999.

[25] J. Puchinger and G. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification," in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, J. Mira and J. Alvarez, Eds., vol. 3562 of *Lecture Notes in Computer Science*, pp. 41–53, Springer, Berlin, Germany, 2005.

[26] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[27] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke and and G. Kendall, Eds., pp. 127–164, Kluwer, Boston, Mass, USA, 2005.

[28] I. Dempsey, M. O'Neill, and A. Brabazon, "Foundations in grammatical," in *Foundations in Grammatical Evolution for Dynamic Environments*, vol. 194, Springer, New York, NY, USA, 2009.

[29] H. lan Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job -shop scheduling, rescheduling, and open-shop scheduling problems," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 375–382, Morgan Kaufmann, Burlington, Mass, USA, 1993.

[30] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

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

[32] C. Maurice, *Particle Swarm Optimization*, Wiley, Estados Unidos, USA, 2006.

[33] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[34] M. F. Tasgetiren, P. N. Suganthan, and Q. Pan, "A discrete particle swarm optimization algorithm for the generalized traveling salesman problem," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 158–167, New York, NY, USA, July 2007.

[35] T. Gong and A. L. Tuson, "Binary particle swarm optimization: a forma analysis approach," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07)*, p. 172, ACM, New York, NY, USA, July 2007.

[36] A. E. M. Zavala, A. H. Aguirre, and E. R. Villa Diharce, "Constrained optimization via Particle Evolutionary Swarm Optimization algorithm (PESO)," in *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '05)*, pp. 209–216, New York, NY, USA, June 2005.

[37] A. E. M. Zavala, A. H. Aguirre, and E. R. Villa Diharce, "Particle evolutionary swarm optimization algorithm (PESO)," in *Proceedings of the 6th Mexican International Conference on Computer Science (ENC '05)*, pp. 282–289, Puebla, Mexico, September 2005.

[38] A. E. Muñoz-Zavala, A. Hernández-Aguirre, E. R. Villa-Diharce, and S. Botello-Rionda, "PESO+ for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 231–238, July 2006.

[39] G. T. Pulido and C. A. C. Coello, "A constraint-handling mechanism for particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 2, pp. 1396–1403, Portland, Ore, USA, June 2004.

[40] S. Martello and P. Toth, *Knapsack Problems. Algorithms and Computer Implementations*, John Wiley & Sons, New York, NY, USA, 1990.

[41] E. G. Coffman Jr., C. Courcoubetis, M. R. Garey, P. W. Shor, and R. R. Weber, "Bin packing with discrete item sizes. I. Perfect packing theorems and the average case behavior of optimal packings," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 384–402, 2000.

[42] T. G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei, "New bin packing fast lower bounds," *Computers and Operations Research*, vol. 34, no. 11, pp. 3439–3457, 2007.

[43] J. Coffman, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: combinatorial analysis," in *Handbook of Combinatorial Optimization*, pp. 151–207, Kluwer Academic, 1999.

[44] S. P. Fekete and J. Schepers, "New classes of fast lower bounds for bin packing problems," *Mathematical Programming*, vol. 91, no. 1, pp. 11–31, 2001.

[45] S. S. Seiden, R. van Stee, and L. Epstein, "New bounds for variable-sized online bin packing," *SIAM Journal on Computing*, vol. 32, no. 2, pp. 455–469, 2003.

[46] E. G. Coffman Jr. and J. Csirik, "A classification scheme for bin packing theory," *Acta Cybernetica*, vol. 18, no. 1, pp. 47–60, 2007.

[47] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1186–1192, May 1992.

[48] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 345–357, 1999.

[49] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2D strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257–300, 2001.

[50] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[51] A. C. F. Alvim, C. C. Ribeiro, F. Glover, and D. J. Aloise, "A hybrid improvement heuristic for the one-dimensional bin packing problem," *Journal of Heuristics*, vol. 10, no. 2, pp. 205–229, 2004.

[52] C. D. T. Suárez, E. P. Gonzlez, and M. V. Rendón, "A heuristic algorithm for the offline one-dimensional bin packing problem inspired by the point Jacobi matrix iterative method," in *Proceedings of the 5th Mexican International Conference on Artificial Intelligence (MICAI '06)*, pp. 281–286, Mexico City, Mexico, November 2006.

[53] S. Tam, H. Tam, L. Tam, and T. Zhang, "A new optimization method, the algorithm of changes, for Bin Packing Problem," in *Proceedings of the IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA '10)*, pp. 994–999, September 2010.

[54] M. Hyde, *A genetic programming hyper-heuristic approach to automated packing [Ph.D. thesis]*, University of Nottingham, 2010.

[55] M. A. Sotelo-Figueroa, H. J. Puga Soberanes, J. Martin Carpio et al., "Evolving bin packing heuristic using micro-differential evolution with indirect representation," in *Recent Advances on Hybrid Intelligent Systems*, vol. 451 of *Studies in Computational Intelligence*, pp. 349–359, Springer, Berlin, Germany, 2013.

[56] M. Sotelo-Figueroa, H. Puga Soberanes, J. Martin Carpio, H. Fraire Huacuja, L. Cruz Reyes, and J. Soria-Alcaraz, "Evolving and reusing bin packing heuristic through grammatical differential evolution," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC '13)*, pp. 92–98, Fargo, ND, USA, August 2013.

[57] E. K. Burke and G. Kendall, *Search Method ologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer, New York, NY, USA, 2006.

[58] A. Rodriguez-Cristerna, J. Torres-Jimenez, I. Rivera-Islas, C. Hernandez-Morales, H. Romero-Monsivais, and A. Jose-Garcia, "A mutation-selection algorithm for the problem of minimum brauer chains," in *Advances in Soft Computing*, I. Batyrshin and G. Sidorov, Eds., vol. 7095 of *Lecture Notes in Computer Science*, pp. 107–118, Springer, Berlin, Germany, 2011.

[59] R. N. Kacker, D. Richard Kuhn, Y. Lei, and J. F. Lawrence, "Combinatorial testing for software: an adaptation of design of experiments," *Measurement*, vol. 46, no. 9, pp. 3745–3752, 2013.

[60] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, Boca Raton, Fla, USA, 2nd edition, 2000.