mathematics

MDPI

Article

A Reactive Population Approach on the Dolphin Echolocation Algorithm for Solving Cell Manufacturing Systems

Ricardo Soto [1], Broderick Crawford [1], Rodrigo Olivares [2,*], César Carrasco [1], Eduardo A. Rodriguez Tello [3], Carlos Castro [4], Fernando Paredes [5] and Hanns de la Fuente-Mella [6]

1 Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile; ricardo.soto@pucv.cl (R.S.); broderick.crawford@pucv.cl (B.C.); ccarrascocarre@gmail.com (C.C.)
2 Escuela de Ingeniería Informática, Universidad de Valparaíso, Valparaíso 2362905, Chile
3 Cinvestav Tamaulipas, Km. 5.5 Carretera Victoria - Soto La Marina, Victoria Tamps. 87130, Mexico; ertello@cinvestav.mx
4 Departamento de Informática, Universidad Técnica Federico Santa Maria, Valparaíso 2390123, Chile; carlos.castro@inf.utfsm.cl
5 Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago 8370109, Chile; fernando.paredes@udp.cl
6 Escuela de Comercio, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile; hanns.delafuente@pucv.cl
* Correspondence: rodrigo.olivares@uv.cl

Received: 9 July 2020; Accepted: 17 August 2020; Published: 19 August 2020

Abstract: In this paper, we integrate the autonomous search paradigm on a swarm intelligence algorithm in order to incorporate the auto-adjust capability on parameter values during the run. We propose an independent procedure that begins to work when it detects a stagnation in a local optimum, and it can be applied to any population-based algorithms. For that, we employ the autonomous search technique which allows solvers to automatically re-configure its solving parameters for enhancing the process when poor performances are detected. This feature is dramatically crucial when swarm intelligence methods are developed and tested. Finding the best parameter values that generate the best results is known as an optimization problem itself. For that, we evaluate the behavior of the population size to autonomously be adapted and controlled during the solving time according to the requirements of the problem. The proposal is testing on the dolphin echolocation algorithm which is a recent swarm intelligence algorithm based on the dolphin feature to navigate underwater and identify prey. As an optimization problem to solve, we test a machine-part cell formation problem which is a widely used technique for improving production flexibility, efficiency, and cost reduction in the manufacturing industry decomposing a manufacturing plant in a set of clusters called cells. The goal is to design a cell layout in such a way that the need for moving parts from one cell to another is minimized. Using statistical non-parametric tests, we demonstrate that the proposed approach efficiently solves 160 well-known cell manufacturing instances outperforming the classic optimization algorithm as well as other approaches reported in the literature, while keeping excellent robustness levels.

Keywords: autonomous search; swarm intelligence; auto-adjust parameter values; cell manufacturing systems

1. Introduction

Modern engineering tries to solve classical problems by combining new trends on technologies with traditional models. For example [1–3], mathematical models formulated by widely-known approaches are applied to improve industrial processes. In this context, the flexibility, efficiency, and productiveness appear as outstanding features in the current competitive manufacturing industries [4,5]. One of the most studied topics in this line of research is the manufacturing system. Developing cellular manufacturing systems is known to be an efficient way for handling those complex industry requirements. Competitive manufacturing industries have modeled their processes for managing the key resources in order to maximize their profits [6–8]. A crucial step for building these competitive systems is the machine-part cell formation (MPCF), which involves the decomposition of a manufacturing industry in completely independent clusters called cells. Those cells contain machines that process parts that share some features or belong to the same family. In an ideal decomposition, each cell is fully dedicated to a family of parts; however, often machines are required by two or more families. This leads to the generation of exceptional elements, that is, parts that need to visit different cells. Unfortunately, such a part flow among cells negatively impacts production times, inventory, and final costs among others [9]. As a consequence, the goal of the MPCF is to smartly design an industry decomposition into a set of cells in such a way the amount of exceptional elements is minimized.

Preliminary experiments were carried out by using classic exact methods such as linear programming [10,11]. Complete techniques are powerful techniques that employ an analysis of the entire search space to verify the global optimum value, and they often require a high amount of computational time and memory. For example, goal programming [12,13] is proposed as an exact method for solving this problem. In [14,15], the constraint programming paradigm is applied to the manufacturing cell design problem, and Ref. [16] includes the boolean satisfiability method for cell formation in group technology. As can be seen, these approaches are competitive when small and medium instances are solved. However, they take too long to solve the hardest instances [17]. For this reason, we believe that metaheuristics successfully work when a balance between the solving-time and good solutions is needed. In this context, metaheuristics as tabu search [18,19], simulated annealing [9], and particle swarm optimization [20] have become significant.

In this research, we solve the MPCF problem testing a swarm intelligence algorithm improved by a self-tuning component for improving the search process, increasing or decreasing the area of potential solutions. We propose a new procedure able to properly identify possible stagnation in a local optimal using information provided by the algorithm performance itself while it moves in the search space. This procedure is implemented by using modular architecture, and it is independent from the optimization algorithm. Therefore, it can be included in any population-based technique. To evaluate our procedure, we firstly employ the dolphin echolocation algorithm as a case study because parameter tuning has been not included in this algorithm yet. This algorithm is a method based on swarm intelligence that mimics the capabilities of dolphins, which employ their genetic sonar for direction-finding and hunting [21]. Such a sonar can emit high-frequency sounds called clicks to find prey. The algorithm represents the solutions as the dolphin positions, while the emission of clicks' balances the exploration and exploitation capabilities of the metaheuristic. To reach the parameter setting paradigm on the dolphin echolocation algorithm, we use the promising capability belonging to Autonomous Search (AS) systems, which promotes the self-configuration of complex optimization solvers [22,23]. The main idea is to let the solver self-adjust some of its parameters in order to alleviate the parameter-setting work of programmers, which is known to be a tedious task, problem-dependent, and it is considered as an optimization problem itself. According to the No Free Lunch theorem [24], there is no general optimal metaheuristic parameter setting. It is not obvious to define a priori which parameter setting should be used. The optimal values for the parameters depend mainly on the problem and even the instance to deal with and on the search time that the user wants

to spend on solving the problem. A universally optimal parameter value set for a given metaheuristic does not exist [25].

Based on this approach, the improved dolphin echolocation algorithm is able to autonomously adjust its population at runtime, according to its exhibited performance. We perform a set of experiments on 160 well-known MPCF instances where the proposed algorithm outperforms the classic algorithm as well as other metaheuristics reported in the literature while keeping excellent robustness levels.

The remainder of this paper is organized as follows: First, in Section 2, we present a literature review of the problem and several techniques to resolve it. Then, the machine-part cell formation with its mathematical model is explained in Section 3. Later, in Section 4, the dolphin echolocation algorithm is detailed. Next, our proposed variation is exposed in Section 5 explaining each one of its steps. Finally, the experimental results, statistical comparisons of results, and conclusions of this work are shown in Sections 6 and 7, respectively.

2. Literature Review

The design of productive cells has had a large impact in the last two decades, due to the importance of establishing a strategy when organizing the location of each machine present in a manufactured industry. However, the interdependence of each productive cell makes it difficult for a product not to carry out transfers between cells. Therefore, the objective of the problem is to minimize the number of inter-cell movements to be able to finalize a product. In this context, several approaches have already been proposed, for instance, exact methods from the mathematical programming domain [10–13,26–28]. We may also find more modern exact methods such as constraint programming [15] and SAT [16]. In addition, exact methods combined with metaheuristics can also be found [29].

Now, considering metaheuristics for resolving the MPCF, we also find an extensive list of works devoted to manufacturing systems, principally using genetic algorithms. For instance, using a traditional approach [30], for the minimization of exceptional elements that conduces to a multi-objective optimization standpoint [31], mixed with discrete event simulation [32] to solve an MPCF mathematical model from the automobile industry, and combined with local search procedures [33].

Predator–prey as a variant of the classic genetic algorithm has also been reported [34], which proposes using as a fitness function a new grouping efficacy computation of cells. The scatter search method has been employed as well for solving manufacturing systems problems, but mostly oriented to multi-criteria group scheduling [35]. Classic simulated annealing [9], tabu search [18,19,36–38], and differential evolution [39] algorithms also present in the literature. These approaches are oriented to the grouping efficacy of cells in machine-part cell formation. A combination of particle swarm optimization and data mining techniques has also been used [20] to solve a well-known set of MPCF problems. More recent metaheuristics have also been employed to solve the this problem, for instance, a migrating birds algorithm is used to solve 90 classic MPCF problems efficiently [40]. An extended version of this work, where the several sorting processes of the metaheuristic are paralleled, has also been published [41]. Finally, another group of modern metaheuristics is proposed to solve this same set of problems efficiently, some examples are artificial fish swarm algorithms [42], shuffled frog leaping algorithms [43], bat algorithms [44], and flower and pollination algorithms [45].

Despite the long list of reported manuscripts being varied and extensive, currently few works are designed to solve this problem using a self-adjusting approach. This work is focused on improving the classic dolphin echolocation algorithm [46] by introducing autonomous search capabilities to the core algorithm, which is able to balance the amount of population required to different parts of the search space. We select this swarm intelligence method because a self-adaptive approach has not been proposed for this algorithm yet. In this context, we investigate a recent proposal about a simplification of the dolphin echolocation algorithm that it needs no empirical parameter to work [47]. In [48],

a modified dolphin echolocation algorithm based on the creation of the basic collapse mechanisms, and it is proposed to optimize analysis of plastic structures. This work verifies the efficiency of the algorithm by comparing results with exact solutions. Similar work can be seen in [49] where again an improved dolphin echolocation algorithm is developed to solve a complex optimization problem.

Using statistical non-parametric tests, we compare the results with the classic algorithm and several other metaheuristics, illustrating the efficiency of our AS algorithm. We additionally analyze a larger and more complex set of instances composed of 160 benchmarks, providing new bounds for a subset of instances whose global optimum is currently unknown.

3. Problem Statement

As previously explained, the MPCF consists of organizing a factory plant in productive cells, where each cell contains machines that process product parts. The idea is to minimize the so-called exceptional elements, which are indeed parts that move from one cell to another to satisfy the production workflow [27]. The mathematical model representing this problem is described as follows:

- Indices.
  - i: machine type, $i \in \{1, \ldots, M\}$.
  - j: part type, $j =\in \{1, \ldots, P\}$.
  - k: cell type, $k =\in \{1, \ldots, C\}$.
- Parameters.
  - M: the number of machines.
  - P: the number of parts.
  - C: the number of cells.
  - M_{max}: the maximum number of machines per cell.
- Variables.
  - $X = [x_{ij}]$: the binary machine-part incidence matrix, where:

$$x_{ij} = \begin{cases} 1 & \text{if machine } i \text{ process the part } j \\ 0 & \text{otherwise} \end{cases}$$

  - $Y = [y_{ik}]$: the binary machine-cell incidence matrix, where:

$$y_{ik} = \begin{cases} 1 & \text{if machine } i \text{ belongs to cell } k \\ 0 & \text{otherwise} \end{cases}$$

  - $Z = [z_{jk}]$: the binary part-cell incidence matrix, where:

$$z_{jk} = \begin{cases} 1 & \text{if part } j \text{ belongs to cell } k \\ 0 & \text{otherwise} \end{cases}$$

The objective function is given by:

$$\texttt{minimize} \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{i_j} z_{j_k} (1 - y_{i_k}) \tag{1}$$

subject to:

$$\sum_{k=1}^{C} y_{ik} = 1, \ \forall \, i \tag{2}$$

$$\sum_{k=1}^{C} z_{jk} = 1, \ \forall \, j \tag{3}$$

$$\sum_{i=1}^{M} y_{ik} \leq M_{max}, \ \forall \ k \tag{4}$$

An example of binary machine-part incidence matrix is depicted in Figure 1 together with its corresponding ideal configuration, where no exceptional elements are present.

	Part									
Machine	1	2	3	4	5	6	7	8	9	10
A			**1**				**1**			**1**
B	**1**	**1**				**1**				
C		**1**				**1**			**1**	
D				**1**	**1**					
E			**1**				**1**			
F			**1**							**1**
G					**1**			**1**		
H			**1**				**1**			
I	**1**	**1**							**1**	
J				**1**	**1**			**1**		

	Part									
Machine	3	7	10	1	2	6	9	4	5	8
A	**1**	**1**	**1**							
E	**1**	**1**								
F	**1**		**1**							
H	**1**	**1**								
B				**1**	**1**	**1**				
C					**1**	**1**	**1**			
I				**1**	**1**		**1**			
D								**1**	**1**	
G									**1**	**1**
J								**1**	**1**	**1**

Figure 1. Initial and ideal incidence machine-part matrices.

This organization is built using the solution of the problem, which is given by the binary machine-cell incidence matrix depicted on the left side of Figure 2. Machines B, C, and I are stated in cell 1; the machines D, G, and J in cell 2; the machines A, E, F, and H in cell 3. An analogous representation of this matrix is depicted in Figure 2, which is employed in this paper as it allows for avoiding the generation of unfeasible solutions when the same machine is assigned to more than one cell by the metaheuristic.

	Machine									
Cell	A	B	C	D	E	F	G	H	I	J
1		1	1						1	
2				1			1			1
3	1				1	1		1		

	Machine									
	A	B	C	D	E	F	G	H	I	J
Cell	3	1	1	2	3	3	2	3	1	2

Figure 2. Two solution representations for the machine-cell matrix.

4. Dolphin Echolocalization Algorithm

Echolocation is an interesting exploration system present in some species of the animal kingdom. One of them is the dolphin, which smartly uses this feature for locating objects over the sea, particularly

its prey. The dolphin echolocation operates by the emission of sounds, called clicks, and echoes. Once the dolphin emits a click, an echo is returned when an object is hit by the sound. The time interval between the click emission and echo reception is used by the dolphin to evaluate the distance from the prey. Using this capability, the dolphin is able to track an object by continuously emitting clicks and receiving its corresponding echo. The dolphin echolocation metaheuristic mimics those behaviors and a general procedure of the algorithm is depicted in Algorithm 1.

- Parameters of the algorithm.
  - $NumberLoops$: Maximum number of iterations (stop criteria).
  - NL: The number of locations. Each location will correspond to a potential solution of the problem.
  - R_e: The effective radius of search. This parameter is used to calculate the accumulated fitness of the problem.
  - $Power$: The degree of the convergence curve.

Algorithm 1: Dolphin echolocation algorithm.

```
1 initialPopulation();
2 while not isStopCriterionReached() do
3 |   calculatePP();
4 |   calculateFitness();
5 |   findBestLocation();
6 |   allocateProbabilities();
7 |   selectNext();
8 |   loopLocations();
9 end
```

The first phase of the algorithm is to sort in ascending or descending order the search space of the problem. To this end, a matrix of alternatives is created. Then, for each decision variable j, an associated vector A_j is created that holds the possible alternatives for j. Vectors for each decision variable are grouped forming the $Alternatives_{MA:NV}$ matrix, where MA is $Max(A_j)_{j=1:NV}$, and NV is the number of variables of the problem. Then, NL random dolphin locations (solutions) are produced and then stored in a $L_{NL \times NV}$ matrix. For the present problem, each solution is composed of a set of machines assigned to cells as depicted in Figure 2.

On line 3, a while loop encloses a set of actions to be performed until the stop criterion is met. An interesting feature of the dolphin echolocation algorithm is the capability of balancing exploration and exploitation capabilities by a user-defined convergence factor. Indeed, the convergence curve on which the process should operate is defined by Equation (5):

$$PP(It_i) = PP_1 + (1 - PP_1)\frac{It_i^{Power} - 1}{(ItNumber)^{Power} - 1} \tag{5}$$

where PP is the predefined probability, PP_1 is the convergence factor of the first iteration, It_i is the iteration i, and $ItNumber$ is the number of iterations. Once PP is computed, the fitness for each location must be calculated. On line 6, the accumulative fitness is computed according to Algorithm 2.

Algorithm 2: Calculate accumulative fitness.

```
for i = {1, ..., numLocation} do
    for j = {1, ..., NV} do
        autonomousSearch() and calculateFitness();
        findPosition L_ij();
        for k = {−R_e, ..., R_e} do
            AF_(A+k)j ← (1/R_e)(R_e − |k|) fitness(i) + AF_(A+k)j
        end
    end
end
```

where $AF_{(A+k)j}$ defines the accumulative fitness of the $(A+k)$th alternative for variable j. The autonomous search method will be detailed in the next section.

Once the accumulative fitness is computed, the best location must be encountered, and the alternatives allocated to the variables of the best location in the AF matrix are set to zero, as shown in Algorithm 3.

Algorithm 3: Restart AF values.

```
for i = {1, ..., alternatives} do
    for j = {1, ..., NV} do
        if i = The best location(j) then
            AF_ij ← 0;
        end
    end
end
```

5. Autonomous Search

The basic version of the dolphin echolocation algorithm does not have capability for controlling its parameters. If the behavior of NL (numbers of location) is studied, we can see that it is valuated before the run of the metaheuristic. When the loop statement of the algorithm is running, the potential solutions are modified updating their location. However, the total number of solutions remains unchanged.

On the one hand, we can observe that the second part of the metaheuristic uses a reset process for the accumulate function. It is a static procedure and it is influenced by the best solution, resting always according to the best location (solution). On the other hand, in the third part of the algorithm, the assignment of probabilities is controlled by condition if the alternative is equal to the best location. Again, the best solution dramatically impacts the performance of the procedure. For both cases, that thought is not necessarily successful, since often a bad solution, sometimes, can lead the search to the global best solution.

These concerns are very important for the efficiently of the algorithm, the reason why we have decided to create an autonomous system approach for evaluating and calculating the value of the NL in an autonomous way. For that, we use autonomous search, which is a particular case of adaptive systems that improve their solving performance by modifying and adjusting themselves to the problem at hand, either by adaptation or supervised adaptation [22,50]. This approach has successfully been applied in constraint programming using bio-inspired algorithms [51] for controlling the process resolution of solver tools [52]. The objective of autonomous search is to allow the metaheuristic to self-adapt the value of the parameter NL during the run, according to the algorithm convergence.

Recent works illustrating how to improve evolutionary algorithms by dynamically controlling parameters during solving time can be seen in [53–58]. Analyzing these works, we have considered to modify the original dolphin echolocation algorithm taking the autonomous search principles for creating a procedure that adaptively vary the number of potential solutions (NL parameter), according to the exhibited efficiency during the search process. The dolphin echolocation algorithm involves at least four parameters, but we vary only the NL parameter because, in a previous sample phase, we observe that this component dramatically impacts the performance of the algorithm. On the other hand, we are convinced that measuring the quality of an autonomous multi-proposal is complicated, due to the fact that there would be no clarity on which parameter gives the best result. As a consequence, we will have a much more straightforward implementation.

We believe this approach also provides know-how for future experiments involving adaptive population for population-based algorithms. The procedure is described in Algorithm 4. Inputs of the procedure are: the number of location NL, the number of variables NV, current iteration t, and number of stagnation ls that represents the number of iterations where the best solution does not improve, commonly called "local search".

Algorithm 4: Adaptive approach for the NL parameter.

Data: NL, NV, t, and ls
Result: New value for the parameter NL and the locations updated

```
1  if t achieves ls then
2      α ← |bestFitness − worstFitness| / Σ_{i=1}^{NL} fitness(i);
3      if Random[0,1) > α then
4          ns ← roundup(NL × α);
5          rand ← Random[0,1];
6          NL ← NL + ns;
7          if Random[0,1) > α then
8              for i = {ns, ..., NL} do
9                  L_i ← best(L_i);
10             end
11         end
12         else
13             for i = {ns, ..., NL} do
14                 for j = {1, ..., NV} do
15                     L_ij ← Random{0,1};
16                 end
17             end
18         end
19     end
20     else
21         rs ← roundup(NL × α);
22         if NL ≥ rs then
23             NL ← NL − rs;
24         end
25     end
26 end
```

Now, we calculate the difference between best and worst solutions, according to the cost of each one and then it is divided by the sum of all costs (Line 2). This value describes the percentage ($Random[0,1) > \alpha$) of maximum separation between solutions. A low percentage value indicates that

solutions are homogeneous skewed to the best solution, in which case the procedure creates new solutions (Lines 4–18). To create these solutions, we reuse the percentage to determinate if they are cloned from best solution (Lines 8–10) or they are randomly generated (Lines 13–17). On the other hand, if $\alpha > Random[0,1)$ means that solutions are heterogeneous, thus the procedure removes the worst solutions (Lines 21–24). In both cases, the percentages calculated can also be seen as a mechanism to support the exploration and exploitation phases. If solutions are homogeneous, this procedure allows for exploring towards new solutions, while, if solutions are heterogeneous, the procedure converges towards a set of similar solutions itself.

Finally, we implement Algorithm 4 for parameter tuning in a modular manner. If we closely analyze this procedure, we can note that it operates independently of the main process of the metaheuristic. When the dolphin echolocation algorithm is caught in a local optimal, the adaptive approach is invoked to modify the population. In this context, we guarantee that this module can be included as a plug and play plugin in more than 80 population-based metaheuristics [59] without an over-effort. In this sense, the proposed contribution is framed to a more broad approach.

6. Experimental Results

In order to evaluate the performance of the autonomous algorithm, we have used two sets of the machine-part cell formation, the first consisting of 90 instances proposed by Boctor [27], while the second consists of 70 new instances which have a higher level of complexity. Parameter setting is known to be a complex task, itself being recognized as an optimization problem. To select the parameters of the algorithm, we performed a sampling test. Best results were obtained using the following configuration taken from [60]:

- Number locations (NL): 10.
- Number loops ($Iterations$): 2000.
- Effective radius (R_e): 1.
- Convergence factor (PP_0): 0.1.
- Degree of the curve ($Power$): 1.
- Local search (ls): 20.

We performed variations only on the number of locations, due to its high level of impact on the efficiency of the algorithm. Autonomous search indicates the changes that must be made will depend on the performance of the algorithm in measurable periods. Thus, we can establish as a premise: "a better quality of solutions leads to an increase in the number of solutions, while a lower quality of solutions means a decrease in the number of solutions".

Our approach has been implemented in `Java` 1.8 and run in a computer with an Intel Core i5 6600 k 3500 MHz processor, 8 GB RAM DDR4 2133 MHz with Windows 10 64 bits as an operating system.

6.1. Boctor Problems

For testing, Boctor refers to 10 basic problems which come out in 90 different instances, where there are instances with two and three numbers of cells. In the first one, the maximum number of machines per cell (M_{max}) varies between 8 and 12. The second one presents variations between 6 and 9. Each of these instances will be described in Table 1 where it is assigned an identifier number for each of them.

In this experiment, population variations were performed every 31 iterations, making increases and decreases randomly. In order to measure the performance of the modified algorithm, comparisons were made with the original algorithm, establishing fixed populations of $NL = \{5, 10, 15, 20, 25, \text{and } 30\}$, which seeks to establish similar conditions to those presented by the modified Dolphin Echolocation Algorithm. The obtained results can be seen in Table 2.

Table 1. Boctor instances.

Problem	Instance	M_{max}	C	OPT	Problem	Instance	M_{max}	C	OPT
1	Boctor 01	8	2	11	10	Boctor 02	8	2	7
2	Boctor 01	9	2	11	11	Boctor 02	9	2	6
3	Boctor 01	10	2	11	12	Boctor 02	10	2	4
4	Boctor 01	11	2	11	13	Boctor 02	11	2	3
5	Boctor 01	12	2	11	14	Boctor 02	12	2	3
6	Boctor 01	6	3	27	15	Boctor 02	6	3	7
7	Boctor 01	7	3	18	16	Boctor 02	7	3	6
8	Boctor 01	8	3	11	17	Boctor 02	8	3	6
9	Boctor 01	9	3	11	18	Boctor 02	9	3	6
19	Boctor 03	8	2	4	28	Boctor 04	8	2	14
20	Boctor 03	9	2	4	29	Boctor 04	9	2	13
21	Boctor 03	10	2	4	30	Boctor 04	10	2	13
22	Boctor 03	11	2	3	31	Boctor 04	11	2	13
23	Boctor 03	12	2	1	32	Boctor 04	12	2	13
24	Boctor 03	6	3	9	33	Boctor 04	6	3	27
25	Boctor 03	7	3	4	34	Boctor 04	7	3	18
26	Boctor 03	8	3	4	35	Boctor 04	8	3	14
27	Boctor 03	9	3	4	36	Boctor 04	9	3	13
37	Boctor 05	8	2	9	46	Boctor 06	8	2	5
38	Boctor 05	9	2	6	47	Boctor 06	9	2	3
39	Boctor 05	10	2	6	48	Boctor 06	10	2	3
40	Boctor 05	11	2	5	49	Boctor 06	11	2	3
41	Boctor 05	12	2	4	50	Boctor 06	12	2	2
42	Boctor 05	6	3	11	51	Boctor 06	6	3	6
43	Boctor 05	7	3	8	52	Boctor 06	7	3	4
44	Boctor 05	8	3	8	53	Boctor 06	8	3	4
45	Boctor 05	9	3	6	54	Boctor 06	9	3	3
55	Boctor 07	8	2	7	64	Boctor 08	8	2	13
56	Boctor 07	9	2	4	65	Boctor 08	9	2	10
57	Boctor 07	10	2	4	66	Boctor 08	10	2	8
58	Boctor 07	11	2	4	67	Boctor 08	11	2	5
59	Boctor 07	12	2	4	68	Boctor 08	12	2	5
60	Boctor 07	6	3	11	69	Boctor 08	6	3	14
61	Boctor 07	7	3	5	70	Boctor 08	7	3	11
62	Boctor 07	8	3	5	71	Boctor 08	8	3	11
63	Boctor 07	9	3	4	72	Boctor 08	9	3	10
73	Boctor 09	8	2	8	82	Boctor 10	8	2	8
74	Boctor 09	9	2	8	83	Boctor 10	9	2	5
75	Boctor 09	10	2	8	84	Boctor 10	10	2	5
76	Boctor 09	11	2	5	85	Boctor 10	11	2	5
77	Boctor 09	12	2	5	86	Boctor 10	12	2	5
78	Boctor 09	6	3	12	87	Boctor 10	6	3	10
79	Boctor 09	7	3	12	88	Boctor 10	7	3	8
80	Boctor 09	8	3	8	89	Boctor 10	8	3	8
81	Boctor 09	9	3	8	90	Boctor 10	9	3	5

Table 2. Boctor and AS results. For space reasons, we renamed *I* instead of *Iterations*.

ID	AS			NL = 5			NL = 10			NL = 15			NL = 20			NL = 25			NL = 30		
	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$
1	5	10	11	6	30	11.6	9	20	11.5	6	20	11.5	10	50	11.4	13	60	11.1	18	60	11
2	4	10	11	4	20	11.5	4	20	11.6	5	20	11.5	5	20	11.3	6	30	11.2	7	20	11
3	4	10	11	4	20	11.4	5	20	11.4	5	30	11.3	7	30	11.2	6	20	11.2	7	30	11
4	5	20	11	7	40	11.6	6	60	11.5	7	50	11.5	6	30	11.3	8	30	11.2	15	90	11.1
5	8	60	11	20	210	11.5	13	150	11.3	10	130	11.3	17	180	11.2	9	80	11.1	12	110	11
6	35	230	27	42	240	27.7	44	240	27.5	49	230	27.4	39	220	27.3	37	250	27.3	55	280	27.2
7	18	80	18	35	330	18.6	24	220	18.5	29	250	18.3	21	190	18.3	25	230	18.2	30	240	18.2
8	10	70	11	14	120	11.5	11	100	11.4	13	120	11.2	14	110	11.2	16	100	11.1	15	90	11.1
9	10	50	11	13	140	11.4	13	140	11.4	14	130	11.3	12	90	11.4	11	120	11.2	16	150	11.1
10	4	30	7	16	120	7.5	9	70	7.7	10	70	7.6	7	50	7.5	8	60	7.4	11	90	7.2
11	6	70	6	8	90	6.6	9	110	6.5	8	80	6.5	9	90	6.5	9	90	6.4	10	100	6.4
12	8	80	4.1	10	100	4.5	10	100	4.5	11	110	4.4	12	130	4.3	13	120	4.3	14	160	4.3
13	3	20	3	3	30	3.6	3	40	3.6	4	40	3.5	4	50	3.4	5	40	3.4	5	50	3.2
14	6	20	3	6	40	3.7	6	30	3.6	8	40	3.5	9	50	3.3	11	50	3.3	12	60	3.3
15	7	40	7	7	50	7.5	8	50	7.4	8	50	7.4	12	70	7.4	13	80	7.3	13	90	7.2
16	5	40	6	6	40	6.6	7	40	6.5	9	70	6.3	14	100	6.3	17	110	6.2	18	130	6.1
17	11	100	6	15	130	6.7	19	180	6.6	22	190	6.5	27	230	6.4	31	290	6.3	38	310	6.3
18	38	200	6	41	270	6.8	44	320	6.7	45	340	6.6	47	390	6.5	49	460	6.5	50	530	6.5
19	7	50	4	7	60	4.4	8	60	4.4	8	70	4.4	10	90	4.3	12	100	4.2	15	90	4.1
20	3	20	4	4	40	4.6	4	50	4.5	7	80	4.5	8	80	4.4	11	90	4.3	12	110	4.2
21	8	50	4.2	9	70	4.7	9	60	4.7	11	90	4.6	12	110	4.5	14	150	4.5	15	160	4.4
22	3	20	3	3	30	3.5	4	40	3.4	6	50	3.4	9	80	3.3	10	110	3.2	11	110	3.2
23	4	20	1.2	4	40	1.7	6	50	1.7	7	80	1.6	7	70	1.6	9	100	1.5	10	100	1.4
24	12	70	9	13	80	9.6	13	90	9.5	15	110	9.4	18	200	9.4	20	240	9.3	21	270	9.3
25	8	60	4	8	90	4.5	9	80	4.6	9	90	4.4	11	190	4.3	12	130	4.3	13	130	4.2
26	5	40	4	5	40	4.5	4	60	4.5	5	60	4.4	8	80	4.3	9	80	4.3	9	90	4.2
27	4	30	4	4	50	4.6	5	60	4.5	7	80	4.5	9	70	4.4	9	100	4.5	10	90	4.4
28	4	20	14	5	30	14.7	6	40	14.5	5	50	14.4	7	60	14.4	8	70	14.3	9	100	14.3
29	6	50	13	7	60	13.5	7	70	13.4	8	70	13.3	9	100	13.3	11	130	13.2	14	170	13.2
30	6	60	13	6	60	13.6	6	80	13.5	7	60	13.5	8	70	13.4	10	90	13.4	11	120	13.3
31	4	30	13	4	40	13.5	5	40	13.5	5	60	13.4	6	80	13.3	7	80	13.2	9	90	13.2
32	3	20	13	3	30	13.6	4	50	13.3	6	50	13.3	8	70	13.2	9	80	13.2	11	110	13.1
33	4	20	27	4	40	27.5	6	50	27.4	9	70	27.4	11	120	27.5	12	130	27.3	14	160	27.3
34	14	120	18	17	180	18.5	15	130	18.3	18	190	18.2	19	210	18.3	22	290	18.2	26	320	18.1
35	11	100	14	13	120	14.6	14	150	14.6	17	190	14.5	20	230	14.3	22	270	14.3	23	300	14.3
36	11	100	13	12	130	13.4	13	120	13.3	14	150	13.3	13	170	13.2	18	200	13.2	19	220	13.1
37	5	30	9.1	5	40	9.7	6	50	9.6	5	40	9.5	6	30	9.5	9	40	9.4	10	60	9.4
38	6	30	6.1	7	40	6.8	8	60	6.7	7	50	6.6	9	50	6.4	0	70	6.4	11	80	6.4
39	8	50	6.1	9	70	6.7	9	60	6.5	11	90	6.5	12	80	6.3	14	90	6.3	15	110	6.3
40	2	10	5	3	30	5.5	4	30	5.4	6	70	5.4	8	40	5.3	9	80	5.2	10	70	5.2
41	2	20	6.1	2	40	6.8	3	40	6.5	5	60	6.4	7	60	6.4	8	50	6.3	10	90	6.3
42	17	100	11	18	200	11.5	18	210	11.5	20	230	11.4	21	220	11.3	18	170	11.3	25	240	11.1
43	5	40	8	5	40	8.6	6	60	8.7	7	60	8.5	8	80	8.4	9	80	8.3	9	100	8.2
44	8	60	8	9	70	8.5	10	80	8.4	11	90	8.3	12	110	8.3	12	100	8.2	13	160	8.1
45	5	30	6	6	40	6.4	7	60	6.3	7	50	6.3	8	70	6.2	9	50	6.2	15	90	6.1

Table 2. *Cont.*

ID	AS			NL = 5			NL = 10			NL = 15			NL = 20			NL = 25			NL = 30		
	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$	t	I	$\bar{x}$
46	2	10	5	3	50	5.6	4	30	5.5	5	70	5.4	9	80	5.4	7	70	5.3	10	110	5.2
47	2	20	3	3	30	3.5	4	20	3.5	4	50	3.4	7	80	3.3	8	100	3.2	10	130	3.2
48	4	60	3.1	6	100	3.7	6	70	3.6	9	130	3.6	11	120	3.5	14	120	3.4	17	180	3.4
49	2	10	3	3	40	3.6	4	60	3.5	5	80	3.5	10	90	3.4	11	110	3.3	14	150	3.2
50	2	10	2.1	3	20	2.8	5	30	2.7	7	50	2.6	8	60	2.3	9	80	2.3	10	110	2.3
51	4	30	6	4	40	6.5	7	50	6.4	9	70	6.3	12	130	6.3	15	130	6.2	17	160	6.2
52	16	150	4	19	180	4.7	18	160	4.5	19	210	4.3	20	240	4.3	17	150	4.2	19	170	4.1
53	5	10	4	6	70	4.5	7	90	4.4	9	100	4.4	10	90	4.2	13	120	4.2	14	140	4.2
54	10	100	3	12	110	3.4	13	100	3.4	14	130	3.3	16	180	3.2	18	210	3.2	22	250	3.1
55	4	20	7	5	50	7.6	4	30	7.5	6	80	7.4	5	60	7.3	8	70	7.2	11	120	7.2
56	4	30	4	4	40	4.5	6	50	4.2	7	60	4.1	8	70	4.1	9	100	4.1	9	110	4
57	3	10	4	3	30	4.4	4	60	4.4	4	50	4.3	6	90	4.3	7	80	4.3	12	130	4.2
58	2	10	4	3	20	4.7	3	40	4.6	4	40	4.5	5	50	4.4	6	80	4.4	7	80	4.3
59	4	20	4	4	30	4.3	4	50	4.3	5	60	4.2	6	70	4.2	8	110	4.1	9	100	4.1
60	22	140	11	28	210	11.5	24	180	11.5	29	190	11.4	30	260	11.3	33	290	11.3	34	300	11.3
61	5	40	5	5	50	5.5	6	50	5.5	7	40	5.3	6	30	5.3	9	40	5.2	10	80	5.2
62	3	30	5.1	4	30	5.6	4	50	5.6	6	70	5.5	8	60	5.4	8	70	5.4	11	140	5.3
63	8	70	4	8	70	4.4	8	90	4.4	9	80	4.4	11	110	4.3	13	140	4.2	15	160	4.2
64	4	30	13	4	40	13.6	4	40	13.5	5	70	13.4	6	80	13.4	7	90	13.2	10	100	13.1
65	4	30	10	4	30	10.7	5	60	10.5	6	50	10.3	7	80	10.3	8	80	10.3	11	90	10.2
66	8	70	8	8	90	8.6	9	80	8.5	9	80	8.5	10	110	8.4	13	140	8.3	17	200	8.2
67	3	10	5	3	20	5.5	3	40	5.4	4	60	5.4	5	70	5.3	7	90	5.3	12	130	5.2
68	2	20	5	2	40	5.4	3	30	5.4	4	70	5.3	8	120	5.2	9	110	5.2	10	130	5.1
69	8	60	14	8	60	14.6	8	70	14.5	9	90	14.5	10	110	14.4	12	120	14.2	13	140	14.1
70	4	30	11	4	30	11.5	4	50	11.5	5	70	11.4	7	80	11.3	7	90	11.2	10	120	11.1
71	5	50	11	5	50	11.5	5	60	11.4	5	70	11.3	6	80	11.2	7	90	11.1	8	100	11
72	15	170	10	16	160	10.6	17	180	10.7	19	200	10.5	21	230	10.3	16	170	10.2	18	190	10.2
73	16	100	8	16	110	8.5	19	210	8.4	17	120	8.5	18	140	8.4	21	230	8.2	22	240	8.1
74	4	10	8	4	30	8.6	4	40	8.6	6	50	8.6	7	60	8.5	7	90	8.4	10	110	8.3
75	2	10	8	3	20	8.4	4	40	8.4	4	40	8.4	4	50	8.3	5	70	8.2	7	80	8.1
76	3	10	5	3	10	5.7	3	30	5.6	3	40	5.4	4	50	5.4	6	80	5.2	9	100	5.1
77	4	30	5	4	50	5.5	5	60	5.5	7	80	5.5	8	80	5.3	9	110	5.1	12	130	5.1
78	7	70	12	7	80	12.3	8	90	12.3	9	90	12.2	10	100	12.2	9	110	12.1	15	210	12.1
79	5	40	12	5	40	12.5	5	50	12.5	6	70	12.4	8	90	12.3	8	100	12.2	11	110	12.2
80	6	50	8	7	60	8.6	7	70	8.6	7	80	8.5	9	70	8.4	10	80	8.3	12	90	8.1
81	15	70	8	16	70	8.5	20	110	8.4	18	90	8.3	17	80	8.2	22	160	8.1	26	190	8
82	5	30	8	5	40	8.7	5	60	8.7	6	70	8.6	8	90	8.4	9	80	8.2	10	90	8.1
83	6	50	5	6	50	5.5	6	80	5.3	8	100	5.3	9	70	5.2	11	120	5.2	12	120	5.1
84	4	20	5	4	30	5.3	4	50	5.3	5	60	5.2	6	70	5.2	8	110	5.1	9	100	5
85	2	20	5	2	30	5.4	3	30	5.5	4	50	5.4	7	80	5.3	8	90	5.3	11	130	5.1
86	2	20	5	3	40	5.6	4	60	5.6	5	60	5.5	9	90	5.4	9	100	5.4	10	110	5.3
87	12	70	10	12	80	10.5	15	110	10.5	18	170	10.4	19	200	10.3	21	230	10.2	24	260	10.1
88	7	90	8	7	90	8.4	7	80	8.4	8	90	8.3	9	100	8.2	10	120	8.2	12	140	8.1
89	8	60	8	8	60	8.6	8	90	8.5	9	80	8.4	10	90	8.3	11	110	8.3	15	170	8.2
90	4	30	5	4	40	5.5	4	60	5.4	5	50	5.3	6	80	5.3	8	90	5.2	9	100	5.1
Avg	6.9	49	7.9	8.2	71.1	8.5	8.6	77.6	8.4	9.7	89.1	8.3	11	103.6	8.2	12.1	116.9	8.2	14.6	140.3	8.1

We already have used metrics to successfully evidence the performance of algorithms [52,61–64]. In this case study, we employ a similar formulation to measure the potential of the adaptive approach:

1. The time was established in milliseconds (t) to reach the global optimum.

2. We used the number of iterations (*Iterations*) necessary to obtain the global optimum.
3. We will use the robustness of the algorithm which is represented by the following formula:

$$robustness = \bar{x} = \frac{\sum Z_i}{31} \tag{6}$$

Within the obtained results, it can be seen that the conjunction of autonomous search with dolphin echolocation algorithm has achieved in 91.1% of the cases to reach a 100% robustness. The experiments that were executed by the original algorithm obtained the following results, $NL = 5$, only 10 instances managed to obtain a robustness greater than 50%; $NL = 10$, 32 instances reached a percentage equal to the previous one; $NL = 15$, 50 instances obtained a robustness greater than 60%; $NL = 20$, 52 instances reached a percentage higher than 70%; $NL = 25$, 48 instances managed to overcome a robustness of 80%; $NL = 30$, 8 instances achieved 100% robustness, which means 8.89% of the experiments performed. However, this is achieved with a significant amount of time and *Iterations* needed to reach the global optimum. On the other hand, in 100% of the cases, the autonomous approach managed to obtain smaller execution times and *Iterations* quantity than the original algorithm. However, with $NL = 5$, similar times and *Iterations* were obtained due to the small amount of population. However, its robustness is largely affected.

With the obtained results, it can be established that, when using the technique of autonomous search, the optimal solution is reached in a smaller amount of time and *Iterations*, which can be observed in Figures 3 and 4. On the other hand, Figure 5 shows the robustness of the algorithm, which is increased when using autonomous search. However, after updating the population with the same robustness increments, but with a higher number of *Iterations* and execution time.

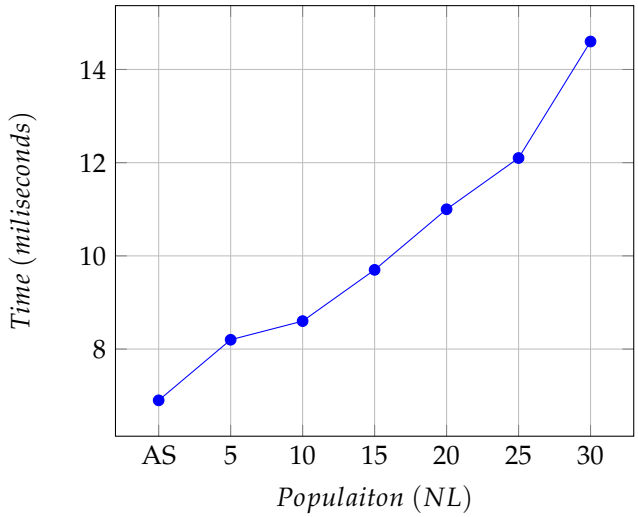

Figure 3. Average of Runtime.

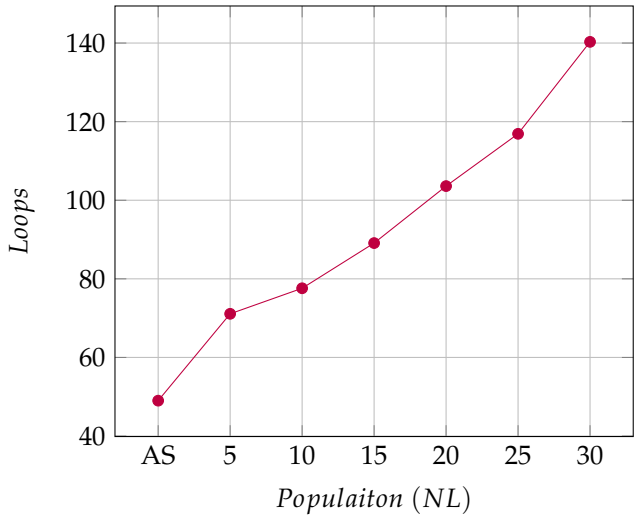

Figure 4. Average of *Iterations*.

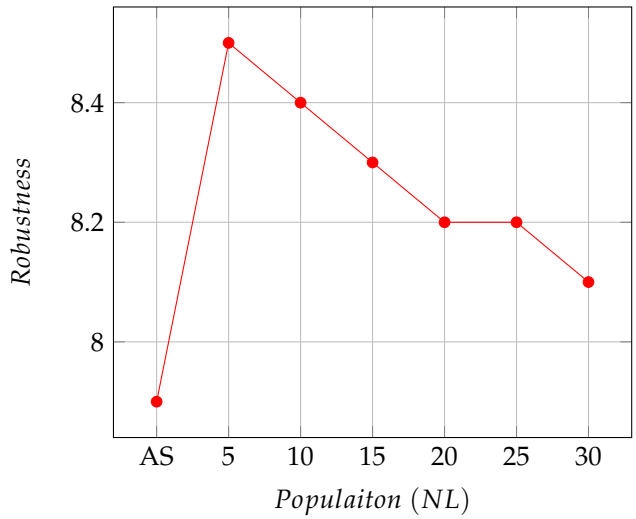

Figure 5. Average of Robustness.

6.2. Other Instances

To reaffirm the performance of the autonomous dolphin echolocation algorithm, proceed to a new phase of experimentation with problems of a greater degree of difficulty, which consists of 70 new instances depicted in Table 3.

As in previous experiments, population changes are performed every 20 iterations (local search number ls, defined at the beginning of this section), which consist of increases or decreases in the number of locations, according to Algorithm 4. To validate the performance of the technique used, it will be compared with the original algorithm, with fixed populations of $NL = \{5, 10, 15, 20, 25$ and $30\}$, in order to establish similar conditions. The obtained results are presented in Table 4.

Table 3. Hardest instances.

Instance	M	P	Problem	M_{max}	C	Problem	M_{max}	C
King–Nakornchai Problem [65]	5	7	1	3	2	2	2	3
Waghodekar–Sah Problem [66]	5	7	3	3	2	4	2	3
Seifoddini Problem [67]	5	18	5	3	2	6	2	3
Kusiak–Chow Problem [26]	6	8	7	3	2	8	2	3
Kusiak–Chow Problem [26]	7	11	9	4	2	10	3	3
Boctor Problem [27]	7	11	11	4	2	12	3	3
Seifoddini–Wolfe Problem [67]	8	12	13	4	2	14	3	3
Chandrasekharan–Rajagopalan Problem [68]	8	20	15	4	2	16	3	3
Chandrasekharan–Rajagopalan Problem [68]	8	20	17	4	2	18	3	3
Mosier–Taube Problem [69]	10	10	19	5	2	20	4	3
Chan and Milner Problem [70]	10	15	21	5	2	22	4	3
Askin–Subramanian Problem [71]	14	24	23	7	2	24	5	3
Stanfel Problem [72]	14	24	25	7	2	26	5	3
McCormick Problem [73]	16	24	27	8	2	28	6	3
Srinivasan Problem [74]	16	30	29	8	2	30	6	3
King Problem [65]	16	43	31	8	2	32	6	3
Carrie Problem [75]	18	24	33	9	2	34	6	3
Mosier–Taube Problem [69]	20	20	35	10	2	36	7	3
Kumar–Vannelli Problem [76]	20	23	37	10	2	38	7	3
Carrie Problem [75]	20	35	39	10	2	40	7	3
Boe–Cheng Problem [77]	20	35	41	10	2	42	7	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	43	12	2	44	8	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	45	12	2	46	8	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	47	12	2	48	8	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	49	12	2	50	8	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	51	12	2	52	8	3
Chandrasekharan–Rajagopalan Problem [68]	24	40	53	12	2	54	8	3
McCormick Problem [73]	27	27	55	14	2	56	9	3
Carrie Problem [75]	28	46	57	14	2	58	10	3
Kumar–Vannelli Problem [76]	30	41	59	15	2	60	10	3
Stanfel Problem [72]	30	50	61	15	2	62	10	3
Stanfel Problem [72]	30	50	63	15	2	64	10	3
King–Nakornchai Problem [65]	30	90	65	18	2	66	12	3
McCormick Problem [73]	37	53	67	19	2	68	13	3
Chandrasekharan–Rajagopalan Problem [68]	40	100	69	20	2	70	14s	3

To measure the performance of the solution, we use the following metrics:

1. Determining the execution time necessary to reach the global optimum, which is expressed in milliseconds.
2. The best optimal found (x).
3. Using Equation (6) to show the robustness ($\bar{x}$) of the obtained results.

Through the obtained results, it is possible to affirm that an autonomous approach obtained in 90% of the cases proposed 100% of the robustness. On the other hand, the experiments that were executed by the original algorithm obtained the following results: when $NL = 5$, only 32% achieved 100%; if $NL = 10$, then 37%; for $NL = 15$, 40% cases are successful; in the case of $NL = 20$, the percentage with good results achieves a maximum of 46% while $NL = 25$ outperforms 58%. Finally, the best results are given by $NL = 30$, where it is able to resolve 140 instances, equivalent to 87% of total. However, these results need a longer execution time compared to the modified algorithm. Although our proposal is better than all configurations of the original version, due to it finding almost 90% of the known optimal, its solving time is overcome by the original version because of the fact that it includes an extra computational-time to the self-adjustment of the NL.

Results allow asserting that using the proposed autonomous approach for the dolphin echolocation algorithm to solve the machine-part cell formation problem is an efficient alternative. We can see that this improvement reaches global optima in a shorter amount of time (see Figure 6). More evidence of the performance of the algorithm is illustrated in Figure 7.

Finally, it can be stated that the use of the autonomous search technique on dolphin echolocation algorithm causes positive impacts on it. Decreasing time and loops, with a considerable increase in robustness, which can be verified in Tables 2 and 4 that were presented in this investigation.

Table 4. Hardest instances and AS results. For space reasons, we renamed *I* instead of *Iterations*.

ID	AS			NL = 5			NL = 10			NL = 15			NL = 20			NL = 25			NL = 30		
	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$
1	1	2	2	1	2	2	2	2	2	3	2	2	2	2	2	4	2	2	4	2	2
2	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	2	0	0	3	0	0
3	1	5	5	1	5	5	1	5	5	2	5	5	2	5	5	3	5	5	4	5	5
4	1	8	8	1	8	8	2	8	8	2	8	8	3	8	8	3	8	8	3	8	8
5	1	5	5	1	5	5	1	5	5	1	5	5	2	5	5	3	5	5	5	5	5
6	1	11	11	1	11	11	1	11	11	2	11	11	3	11	11	3	11	11	4	11	11
7	1	2	2	1	2	2	2	2	2	1	2	2	3	2	2	3	2	2	4	2	2
8	1	7	7	1	7	7	1	7	7	3	7	7	3	7	7	3	7	7	4	7	7
9	1	3	3	1	3	3	1	3	3	1	3	3	2	3	3	3	3	3	3	3	3
10	1	5	5	2	5	5	2	5	5	3	5	5	3	5	5	3	5	5	5	5	5
11	1	2	2	2	2	2	1	2	2	3	2	2	3	2	2	4	2	2	4	2	2
12	1	2	2	1	2	2	2	2	2	2	2	2	2	2	2	3	2	2	3	2	2
13	1	6	6	1	6	6	1	6	6	1	6	6	3	6	6	2	6	6	4	6	6
14	1	7	7	2	7	7	2	7	7	1	7	7	2	7	7	3	7	7	5	7	7
15	1	7	7	1	7	7	1	7	7	2	7	7	4	7	7	3	7	7	4	7	7
16	1	14	14	1	14	14	2	14	14	3	14	14	4	14	14	5	14	14	5	14	14
17	1	28	28	1	28	28	2	28	28	4	28	28	5	28	28	6	28	28	5	28	28
18	1	39	39	1	39	39	4	39	39	4	39	39	6	39	39	6	39	39	7	39	39
19	1	1	1	1	1	1	2	1	1	2	1	1	3	1	1	4	1	1	5	1	1
20	1	0	0	2	0	0	2	0	0	3	0	0	4	0	0	4	0	0	4	0	0
21	1	4	4	1	4	4	2	4	4	2	4	4	4	4	4	5	4	4	6	4	4
22	1	0	0	2	0	0	3	0	0	4	0	0	4	0	0	5	0	0	6	0	0
23	3	1	1	4	1	1.1	4	1	1	5	1	1	5	1	1	6	1	1	7	1	1
24	2	2	2	2	2	2	3	2	2	4	2	2	5	2	2	5	2	2	6	2	2
25	2	2	2	4	2	2	3	2	2	3	2	2	4	2	2	4	2	2	5	2	2
26	3	2	2	3	2	2.1	4	2	2	5	2	2	7	2	2	7	2	2	8	2	2
27	4	16	16	5	16	16.2	7	16	16.1	7	16	16.1	8	16	16	8	16	16	9	16	16
28	3	22	22	3	22	22	4	22	22	4	22	22	5	22	22	7	22	22	8	22	22
29	2	12	12	3	12	12	3	12	12	4	12	12	4	12	12	5	12	12	6	12	12
30	4	17	17	5	17	17.1	5	17	17	5	17	17	5	17	17	6	17	17	7	17	17
31	4	15	15	5	15	15.1	6	15	15.1	6	15	15	7	15	15	7	15	15	8	15	15
32	6	21	21.3	7	21	21.8	8	21	21.6	10	21	21.5	11	21	21.5	13	21	21.4	15	21	21.4
33	3	13	13	4	13	13	4	13	13	5	13	13	6	13	13	7	13	13	7	13	13
34	4	18	18	5	18	18.1	5	18	18	6	18	18	7	18	18	8	18	18	8	18	18
35	5	27	27	6	27	27.1	6	27	27	7	27	27	8	27	27	8	27	27	8	27	27

Table 4. *Cont.*

ID	AS			NL = 5			NL = 10			NL = 15			NL = 20			NL = 25			NL = 30		
	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$	t	x	$\bar{x}$
36	7	41	41	9	41	41.3	8	41	41.2	8	41	41.2	10	41	41.1	12	41	41	13	41	41
37	4	25	25	4	25	25	5	25	25	5	25	25	6	25	25.1	7	25	25	9	25	25
38	9	34	34.2	9	34	34.8	10	34	34.5	13	34	34.4	14	34	34.3	17	34	34.2	20	34	34.2
39	10	1	1	7	1	1.1	9	1	1.1	10	1	1.1	12	1	1.1	14	1	1	16	1	1
40	12	13	13	13	13	13.4	13	13	13.3	15	13	13.3	16	13	13.1	17	13	13.2	20	13	13
41	11	21	21.3	11	21	21.9	12	21	21.8	13	21	21.8	15	21	21.7	18	21	21.8	23	21	21.7
42	13	34	34	13	34	34.2	15	34	34.2	15	34	34.1	17	34	34.1	16	34	34	17	34	34
43	14	0	0.1	13	0	0.4	15	0	0.3	18	0	0.2	19	0	0.2	21	0	0.1	23	0	0
44	9	3	3	13	3	3.9	14	3	3.5	17	3	3.4	20	3	3.2	23	3	3.1	31	3	3
45	16	3	3.3	18	3	3.9	19	3	3.7	21	3	3.6	24	3	3.5	25	3	3.6	33	3	3.4
46	15	4	4.1	15	4	4.8	16	4	4.8	18	4	4.7	21	4	4.4	25	4	4.4	26	4	4.3
47	12	9	9	13	9	9.5	15	9	9.5	16	9	9.4	19	9	9.3	21	9	9.3	26	9	9.1
48	7	13	13	8	13	13.3	9	13	13.2	11	13	13.2	14	13	13.1	15	13	13	20	13	13
49	10	19	19	11	19	19.1	12	19	19.1	13	19	19.1	12	19	19	14	19	19	16	19	19
50	14	28	28	15	28	28.4	15	28	28.3	16	28	28.2	17	28	28.1	18	28	28	18	28	28
51	18	22	22	19	22	22.3	24	22	22.2	24	22	22.2	25	22	22.1	27	22	22	28	22	22
52	14	32	32	15	32	32.3	15	32	32.2	17	32	32.1	17	32	32.1	18	32	32	20	32	32
53	13	22	22	16	22	22.1	17	22	22.1	19	22	22	20	22	22	21	22	22	25	22	22
54	16	31	31.2	17	31	31.9	18	31	31.7	19	31	31.7	20	31	31.4	23	31	31.3	25	31	31.2
55	17	32	32	18	32	32	18	32	32	19	32	32	20	32	32	21	32	32	20	32	32
56	19	66	66.1	20	66	66.5	22	66	66.4	23	66	66.3	26	66	66.2	29	66	66.1	34	66	66.1
57	9	36	36	9	36	36.3	10	36	36.2	13	36	36.1	14	36	36	17	36	36	20	36	36
58	17	49	49	19	49	49.2	18	49	49.2	18	49	49.2	20	49	49.1	22	49	49	23	49	49
59	12	5	5	12	5	5.2	13	5	5.1	16	5	5	17	5	5	18	5	5	19	5	5
60	11	7	7	11	7	7.4	13	7	7.3	12	7	7.3	14	7	7.2	16	7	7.1	20	7	7
61	20	5	5.1	23	5	5.7	25	5	5.6	24	5	5.4	27	5	5.3	28	5	5.3	29	5	5.2
62	18	9	9	19	9	9.3	20	9	9.2	22	9	9.1	24	9	9.1	27	9	9	30	9	9.1
63	12	23	23	13	23	23.3	13	23	23.4	15	23	23.3	16	23	23.2	17	23	23.2	20	23	23.1
64	11	33	33	13	33	33.9	14	33	33.5	17	33	33.4	20	33	33.2	23	33	33.1	31	33	33
65	21	28	28	25	28	28.3	25	28	28.2	26	28	28.1	27	28	28.1	28	28	28.1	33	28	28
66	15	47	47	16	47	47.2	16	47	47.2	17	47	47.1	18	47	47	21	47	47.1	28	47	47
67	31	212	212	33	212	212.3	36	212	212.2	37	212	212	39	212	212.1	43	212	212	47	212	212
68	28	304	304	29	304	304.1	31	304	304.1	32	304	304.2	35	304	304.1	40	304	304.1	44	304	304
69	14	13	13	15	13	13.3	16	13	13.2	17	13	13.1	18	13	13	19	13	13	21	13	13
70	15	19	19	17	19	19.3	18	19	19.3	20	19	19.2	22	19	19.1	24	19	19	26	19	19
Avg	8	22.4	22.4	8.8	22.4	22.6	9.6	22.4	22.6	10.4	22.4	22.5	11.8	22.4	22.5	13	22.4	22.5	15	22.4	22.5

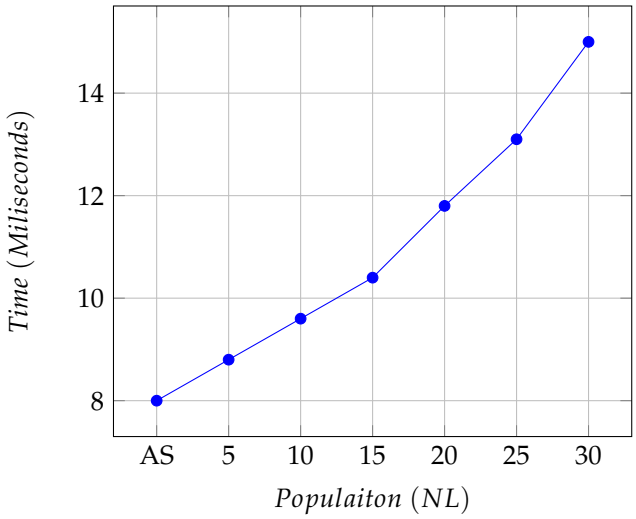

Figure 6. Average of Runtime.

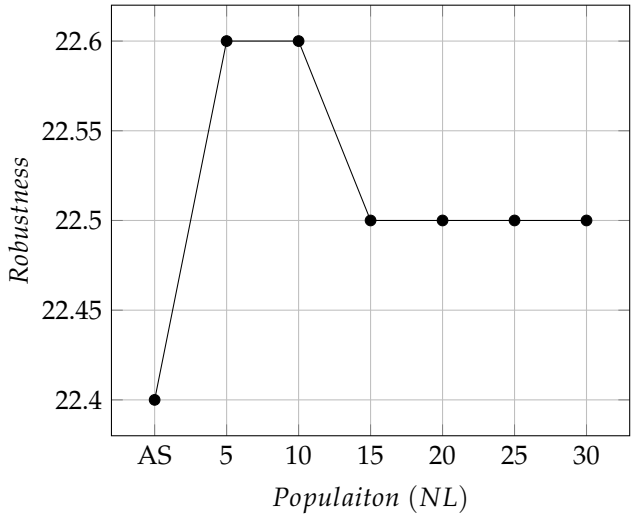

Figure 7. Average of Robustness.

6.3. Statistical Analysis

To test a significant difference between the original version of the Dolphin Echolocation Algorithm and our autonomous proposal, we tested with a statistical test for each instance through the Kolmogorov–Smirnov–Lilliefors. This test allows us to determine if samples are independent [78], and then we use Wilcoxon's signed rank [79] for statistically comparing the results. For this evaluation, the basic dolphin echolocation algorithm with $NL = 30$ was used, due to it reporting better results.

For the Kolmogorov–Smirnov–Lilliefors test, we consider a hypothesis evaluation (p-value ≤ 0.05), smaller values that 0.05 determine that the null hypothesis will be reject. For the Wilcoxon's signed rank, we observe the averages of each instance and discover that there is not a significant difference between the results of both approaches. This leads us to decrease the levels of significance (p-value ≤ 0.01), and thus it has a higher precision. Both tests were conducted using GNU Octave [80].

The Kolmogorov–Smirnov–Lilliefors test is necessary to study the independence of samples by determining if the $\bar{x}$ reaches from the 31 executions of each instance describe a Gaussian distribution. For that, we use H_0 as null hypotheses which states that $\bar{x}$ follows a normal distribution. The alternative hypotheses H_1 state the opposite. We perform the test to demonstrate the alternative hypothesis. The obtained result for the p-value was $0.071 \geq 0.05$; then, H_1 must be rejected. This evidence implies

that applying the central limit theorem is not viable. Then, as the samples are not distributed normally, we have chosen a non-parametric statistical hypothesis test, *Wilcoxon's signed rank* to compare these results. For the *Wilcoxon's signed rank* test, H_0 describes the null hypotheses and it states that $\bar{x}$ achieved by our autonomous approach is better than $\bar{x}$ achieved by the dolphin echolocation algorithm.

Tables 5 and 6 compare the original algorithm versus the autonomous approach, for all tested instances via the *Wilcoxon's signed rank* test. As the significance level is also established to 0.05, smaller values that 0.05 defines that H_0 cannot be assumed. To know the winner, it is enough to observe the row of the instance and identify the column where the p-value exists, for example in the problem 4 (Boctor 01), the autonomous approach is better than the basic DEA because its value is lower than 0.05, then H_1 cannot be assumed. In the other cases, there is no information.

Table 5. Statistical test. DEA vs. Autonomous approach solving the Boctor instance.

Problem	Instance	DEA (NL = 30)	Autonomous DEA	Problem	Instance	DEA (NL = 30)	Autonomous DEA
1	Boctor 01	–	–	10	Boctor 02	–	0.0016
2	Boctor 01	–	–	11	Boctor 02	–	0.0029
3	Boctor 01	–	–	12	Boctor 02	–	–
4	Boctor 01	–	0.0017	13	Boctor 02	–	0.0015
5	Boctor 01	–	–	14	Boctor 02	–	0.0012
6	Boctor 01	–	0.0013	15	Boctor 02	–	0.0012
7	Boctor 01	–	0.0022	16	Boctor 02	–	0.0029
8	Boctor 01	–	–	17	Boctor 02	–	0.0011
9	Boctor 01	–	0.0181	18	Boctor 02	–	0.0022
19	Boctor 03	–	–	28	Boctor 04	–	0.0013
20	Boctor 03	–	0.0023	29	Boctor 04	–	0.0018
21	Boctor 03	–	–	30	Boctor 04	–	0.0016
22	Boctor 03	–	0.0025	31	Boctor 04	–	0.0018
23	Boctor 03	–	–	32	Boctor 04	–	0.0025
24	Boctor 03	–	0.0021	33	Boctor 04	–	0.0018
25	Boctor 03	–	0.0018	34	Boctor 04	–	0.0015
26	Boctor 03	–	0.0012	35	Boctor 04	–	0.0021
27	Boctor 03	–	0.0019	36	Boctor 04	–	0.0012
37	Boctor 03	–	0.0021	46	Boctor 06	–	0.0022
38	Boctor 05	–	0.0017	47	Boctor 06	–	0.0018
39	Boctor 05	–	0.0028	48	Boctor 06	–	–
40	Boctor 05	–	0.0017	49	Boctor 06	–	0.0014
41	Boctor 05	–	0.0014	50	Boctor 06	–	–
42	Boctor 05	–	0.0018	51	Boctor 06	–	0.0025
43	Boctor 05	–	0.0021	52	Boctor 06	–	0.0028
44	Boctor 05	–	0.0016	53	Boctor 06	–	0.0017
45	Boctor 05	–	0.0017	54	Boctor 06	–	0.0022
55	Boctor 07	–	0.0018	64	Boctor 08	–	0.0019
56	Boctor 07	–	0.0029	65	Boctor 08	–	0.0018
57	Boctor 07	–	0.0019	66	Boctor 08	–	0.0015
58	Boctor 07	–	0.0017	67	Boctor 08	–	0.0021
59	Boctor 07	–	0.0024	68	Boctor 08	–	0.0018
60	Boctor 07	–	0.0025	69	Boctor 08	–	0.0021
61	Boctor 07	–	0.0025	70	Boctor 08	–	0.0029
62	Boctor 07	–	–	71	Boctor 08	–	0.0022
63	Boctor 07	–	0.0023	72	Boctor 08	–	0.0028
73	Boctor 09	–	0.0021	82	Boctor 10	–	0.0025
74	Boctor 09	–	0.0021	83	Boctor 10	–	0.0022
75	Boctor 09	–	0.0014	84	Boctor 10	–	0.0018
76	Boctor 09	–	0.0029	85	Boctor 10	–	0.0018
77	Boctor 09	–	0.0024	86	Boctor 10	–	0.0018
78	Boctor 09	–	0.0017	87	Boctor 10	–	0.0017
79	Boctor 09	–	0.0011	88	Boctor 10	–	0.0015
80	Boctor 09	–	0.0016	89	Boctor 10	–	0.0015
81	Boctor 09	–	0.0019	90	Boctor 10	–	0.0017

Table 6. Statistical test. DEA vs. Autonomous approach solving the hardest instance.

Problem	DEA (NL = 30)	Autonomous DEA	Problem	DEA (NL = 30)	Autonomous DEA
–	36	–	–		
2	–	–	37	–	–
3	–	–	38	–	0.0008
4	–	–	39	–	–
5	–	–	40	–	–
6	–	–	41	–	0.0012
7	–	–	42	–	–
8	–	–	43	0.0011	–
9	–	–	44	–	–
10	–	–	45	–	0.0011
11	–	–	46	–	0.0015
12	–	–	47	–	0.0013
13	–	–	48	–	–
14	–	–	49	–	–
15	–	–	50	–	–
16	–	–	51	–	–
17	–	–	52	–	–
18	–	–	53	–	–
19	–	–	54	–	0.0008
20	–	–	55	–	–
21	–	–	56	–	0.0011
22	–	–	57	–	–
23	–	–	58	–	–
24	–	–	59	–	–
25	–	–	60	–	–
26	–	–	61	–	0.0014
27	–	–	62	–	0.0008
28	–	–	63	–	0.0014
29	–	–	64	–	–
30	–	–	65	–	–
31	–	–	66	–	–
32	–	0.0025	67	–	–
33	–	–	68	–	–
34	–	–	69	–	–
35	–	–	70	–	–

According to results in the 90 first instances, for p-value lower than 0.01 (p-value $\leq$ 0.01) for the original dolphin echolocation algorithm are 0, for the autonomous approach are 78. The rest of tests do not provide significant information, due to p-values being greater than 0.01, but less than 0.99, or samples are equal. Now, considering the 70 hard instances, the original dolphin echolocation algorithm achieves 1 p-value lower than 0.01, while our autonomous approach reaches 1 p-value below 0.01. For the rest of the tests, p-values are not significant or, again, samples are equal.

Summarizing, we can note that the improved algorithm exhibits a better performance concerning its basic version only in the smaller Doctor's instances and a few of the hardest instances. We believe this behavior is due to the native algorithm operating appropriately when it faces complex optimization problems. However, as previously mentioned, our proposed self-adaptive algorithm can be considered a contribution itself and it can be applied to a wide set of population-based methods.

7. Conclusions

In this research, we have presented an adaptive procedure that allows for identifying a stagnation in a local optimal. This feature appears when the variability of solutions is required to explore different feasible regions of an optimization problem. The technique is implemented via modular architecture and it is independent of the optimization algorithm, and therefore it can be included in any population-based method. To test this approach, we employ the dolphin echolocation algorithm to solve different instances of the machine-part cell formation. This approach is inspired by the online control for number location parameter which is valuated before the run of the metaheuristic.

To this end, we use an autonomous search that is a particular case of adaptive systems that improve their solving performance by modifying and adjusting themselves to the problem at hand, either by adaptation or supervised adaptation. We have tested two sets of the machine-part cell formation, the first one consisted of 90 instances proposed by Boctor, while the second one included 70 new hard instances, where several global optimum values that were not reached using the basic dolphin echolocation algorithm were achieved via the auto-adaptive approach. We have also compared the proposed adaptive approach by using a nonparametric statistical tests and the results are conclusive. As a conclusion, and as a way to compare the methods, the use of the autonomous search technique on the dolphin echolocation algorithm causes positive impacts on it, decreasing time and loops, with a considerable increase in robustness.

As future work, we plan to experiment auto-adaptive approaches in recent bio-inspired algorithms and to provide a larger comparison of techniques for solving the machine-part cell formation. The integration of autonomous search can lead the research toward new study lines, such as dynamically selecting the best binarization strategy during solving according to performance indicators as analogously studied in [81].

Author Contributions: Formal analysis, R.O. and C.C. (Cécar Carrasco); investigation, R.S., B.C., R.O., H.d.l.F.-M., and E.A.R.T.; methodology, R.S. and B.C.; resources, R.S. and B.C.; software, R.O. and C.C. (Cécar Carrasco); validation, H.d.l.F.-M. and E.A.R.T., F.P., C.C. (Carlos Castro), and R.O.; writing—original draft, C.C. (Cécar Carrasco); writing—review and editing, R.S., B.C., E.A.R.T., and H.d.l.F.-M. All the authors of this paper hold responsibility for every part of this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1171243. Hanns de la Fuente-Mella, Ricardo Soto, and Broderick Crawford are supported by Grant Nucleo de Investigacion en Data Analytics/VRIEA/PUCV/039.432/2020.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Božek, P.; Ivandić, Ž.; Lozhkin, A.; Lyalin, V.; Tarasov, V. Solutions to the characteristic equation for industrial robot's elliptic trajectories. *Teh. Vjesn. Tech. Gaz.* **2016**, *23*. [CrossRef]
2. Božek, P.; Al Akkad, M.A.; Blištan, P.; Ibrahim, N.I. Navigation control and stability investigation of a mobile robot based on a hexacopter equipped with an integrated manipulator. *Int. J. Adv. Robot. Syst.* **2017**, *14*. [CrossRef]
3. Kilin, A.; Božek, P.; Karavaev, Y.; Klekovkin, A.; Shestakov, V. Experimental investigations of a highly maneuverable mobile omniwheel robot. *Int. J. Adv. Robot. Syst.* **2017**, *14*. [CrossRef]
4. Zhang, Z. Modeling complexity of cellular manufacturing systems. *Appl. Math. Model.* **2011**, *35*, 4189–4195. [CrossRef]
5. Gu, P.; Monid, A. Design of cellular manufacturing systems. An industrial case study. *Robot. Comput. Integr. Manuf.* **1993**, *10*, 147–151. [CrossRef]
6. Ah kioon, S.; Bulgak, A.A.; Bektas, T. Integrated cellular manufacturing systems design with production planning and dynamic system reconfiguration. *Eur. J. Oper. Res.* **2009**, *192*, 414–428. [CrossRef]
7. Saxena, L.K.; Jain, P.K. Dynamic cellular manufacturing systems design—A comprehensive model. *Int. J. Adv. Manuf. Technol.* **2011**, *53*, 11–34. [CrossRef]
8. De la Fuente-Mella, H.; Rojas Fuentes, J.L.; Leiva, V. Econometric modeling of productivity and technical efficiency in the Chilean manufacturing industry. *Comput. Ind. Eng.* **2020**, *139*, 105793. [CrossRef]
9. Wu, T.H.; Chang, C.C.; Chung, S.H. A simulated annealing algorithm for manufacturing cell formation problems. *Expert Syst. Appl.* **2008**, *34*, 1609–1617. [CrossRef]
10. Purcheck, G.F.K. A Linear–programming method for the combinatorial grouping of an incomplete power set. *J. Cybern.* **1975**, *5*, 51–76. [CrossRef]

11. Oliva-Lopez, E.; Purcheck, G. Load balancing for group technology planning and control. *Int. J. Mach. Tool Des. Res.* **1979**, *19*, 259–274. [CrossRef]
12. Sankaran, S.; Rodin, E.Y. Multiple objective decision-making approach to cell formation: A goal programming model. *Math. Comput. Model.* **1990**, *13*, 71–81. [CrossRef]
13. Shafer, S.M.; Rogers, D.F. A goal programming approach to the cell formation problem. *J. Oper. Manag.* **1991**, *10*, 28–43. [CrossRef]
14. Soto, R.; Kjellerstrand, H.; Gutiérrez, J.; López, A.; Crawford, B.; Monfroy, E. Solving manufacturing cell design problems using constraint programming. In *Advanced Research in Applied Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 400–406. [CrossRef]
15. Soto, R.; Crawford, B.; Almonacid, B.; Paredes, F.; Loyola, E. Machine-part cell formation problems with constraint programming. In Proceedings of the 2015 34th International Conference of the Chilean Computer Science Society (SCCC), Santiago, Chile, 9–13 November 2015. [CrossRef]
16. Soto, R.; Kjellerstrand, H.; Durán, O.; Crawford, B.; Monfroy, E.; Paredes, F. Cell formation in group technology using constraint programming and Boolean satisfiability. *Expert Syst. Appl.* **2012**, *39*, 11423–11427. [CrossRef]
17. Almonacid, B.; Aspée, F.; Soto, R.; Crawford, B.; Lama, J. Solving the manufacturing cell design problem using the modified binary firefly algorithm and the egyptian vulture optimisation algorithm. *IET Softw.* **2017**, *11*, 105–115. [CrossRef]
18. Aljaber, N.; Baek, W.; Chen, C.L. A tabu search approach to the cell formation problem. *Comput. Ind. Eng.* **1997**, *32*, 169–185. [CrossRef]
19. Lozano, S.; Adenso-Diaz, B.; Eguia, I.; Onieva, L. A one-step tabu search algorithm for manufacturing cell design. *J. Oper. Res. Soc.* **1999**, *50*, 509. [CrossRef]
20. Durán, O.; Rodriguez, N.; Consalter, L.A. Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. *Expert Syst. Appl.* **2010**, *37*, 1563–1567. [CrossRef]
21. Wu, T.Q.; Yao, M.; Hua Yang, J. Dolphin swarm algorithm. *Front. Inf. Technol. Electron. Eng.* **2016**, *17*, 717–729. [CrossRef]
22. Hamadi, Y.; Monfroy, E.; Saubion, F. What is autonomous search? In *Hybrid Optimization*; Springer: New York, NY, USA, 2010; pp. 357–391. [CrossRef]
23. Huang, C.; Li, Y.; Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* **2020**, *24*, 201–216. [CrossRef]
24. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
25. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
26. Kusiak, A.; Chow, W.S. Efficient solving of the group technology problem. *J. Manuf. Syst.* **1987**, *6*, 117–124. [CrossRef]
27. Boctor, F.F. A jinear formulation of the machine-part cell formation problem. *Int. J. Prod. Res.* **1991**, *29*, 343–356. [CrossRef]
28. Albadawi, Z.; Bashir, H.A.; Chen, M. A mathematical approach for the formation of manufacturing cells. *Comput. Ind. Eng.* **2005**, *48*, 3–21. [CrossRef]
29. Boulif, M.; Atif, K. A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem. *Comput. Oper. Res.* **2006**, *33*, 2219–2245. [CrossRef]
30. Venugopal, V.; Narendran, T. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Comput. Ind. Eng.* **1992**, *22*, 469–480. [CrossRef]
31. Gupta, Y.; Gupta, M.; Kumar, A.; Sundaram, C. A genetic algorithm-based approach to cell composition and layout design problems. *Int. J. Prod. Res.* **1996**, *34*, 447–482. [CrossRef]
32. Imran, M.; Kang, C.; Lee, Y.H.; Jahanzaib, M.; Aziz, H. Cell formation in a cellular manufacturing system using simulation integrated hybrid genetic algorithm. *Comput. Ind. Eng.* **2017**, *105*, 123–135. [CrossRef]
33. Nsakanda, A.L.; Diaby, M.; Price, W.L. Hybrid genetic approach for solving large-scale capacitated cell formation problems with multiple routings. *Eur. J. Oper. Res.* **2006**, *171*, 1051–1070. [CrossRef]
34. Banerjee, I.; Das, P. Group technology based adaptive cell formation using predator–prey genetic algorithm. *Appl. Soft Comput.* **2012**, *12*, 559–572. [CrossRef]

35. Tavakkoli-Moghaddam, R.; Javadian, N.; Khorrami, A.; Gholipour-Kanani, Y. Design of a scatter search method for a novel multi-criteria group scheduling problem in a cellular manufacturing system. *Expert Syst. Appl.* **2010**, *37*, 2661–2669. [CrossRef]
36. Chang, C.C.; Wu, T.H.; Wu, C.W. An efficient approach to determine cell formation, cell layout and intracellular machine sequence in cellular manufacturing systems. *Comput. Ind. Eng.* **2013**, *66*, 438–450. [CrossRef]
37. Lei, D.; Wu, Z. Tabu search-based approach to multi-objective machine-part cell formation. *Int. J. Prod. Res.* **2005**, *43*, 5241–5252. [CrossRef]
38. Chung, S.H.; Wu, T.H.; Chang, C.C. An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. *Comput. Ind. Eng.* **2011**, *60*, 7–15. [CrossRef]
39. Noktehdan, A.; Karimi, B.; Kashan, A.H. A differential evolution algorithm for the manufacturing cell formation problem using group based operators. *Expert Syst. Appl.* **2010**, *37*, 4822–4829. [CrossRef]
40. Soto, R.; Crawford, B.; Almonacid, B.; Paredes, F. A migrating birds optimization algorithm for machine-part cell formation problems. In *Advances in Artificial Intelligence and Soft Computing: 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, 25–31 October 2015, Proceedings, Part I*; Springer: Cham, Switzerland, 2015; pp. 270–281. [CrossRef]
41. Soto, R.; Crawford, B.; Almonacid, B.; Paredes, F. Efficient parallel sorting for migrating birds optimization when solving machine-part cell formation problems. *Sci. Program.* **2016**, *2016*, 9402503, [CrossRef]
42. Soto, R.; Crawford, B.; Vega, E.; Paredes, F. Solving manufacturing cell design problems using an artificial fish swarm algorithm. In *Advances in Artificial Intelligence and Soft Computing: 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, 25–31 October 2015, Proceedings, Part I*; Springer: Cham, Switzerland, 2015; pp. 282–290. [CrossRef]
43. Soto, R.; Crawford, B.; Vega, E.; Johnson, F.; Paredes, F. Solving manufacturing cell design problems using a shuffled frog leaping algorithm. In *The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), 28–30 November 2015, Beni Suef, Egypt*; Springer: Cham, Switzerland, 2016; pp. 253–261. [CrossRef]
44. Soto, R.; Crawford, B.; Alarcón, A.; Zec, C.; Vega, E.; Reyes, V.; Araya, I.; Olguín, E. Solving manufacturing cell design problems by using a bat algorithm approach. In *Advances in Swarm Intelligence: 7th International Conference, ICSI 2016, Bali, Indonesia, 25–30 June 2016, Proceedings, Part I*; Springer: Cham, Switzerland, 2016; pp. 184–191. [CrossRef]
45. Soto, R.; Crawford, B.; Olivares, R.; De Conti, M.; Rubio, R.; Almonacid, B.; Niklander, S. Resolving the manufacturing cell design problem using the flower pollination algorithm. In *Multi-Disciplinary Trends in Artificial Intelligence: 10th International Workshop, MIWAI 2016, Chiang Mai, Thailand, 7–9 December 2016, Proceedings*; Springer: Cham, Switzerland, 2016; pp. 184–195. [CrossRef]
46. Soto, R.; Crawford, B.; Carrasco, C.; Almonacid, B.; Reyes, V.; Araya, I.; Misra, S.; Olguín, E. Solving manufacturing cell design problems by using a dolphin echolocation algorithm. In *Computational Science and Its Applications—ICCSA 2016*; Springer: Cham, Switzerland, 2016; pp. 77–86. [CrossRef]
47. Kaveh, A.; Vaez, S.R.H.; Hosseini, P. Simplified dolphin echolocation algorithm for optimum design of frame. *Smart Struct. Syst.* **2018**, *21*, 321–333. [CrossRef]
48. Daryan, A.S.; Palizi, S.; Farhoudi, N. Optimization of plastic analysis of moment frames using modified dolphin echolocation algorithm. *Adv. Struct. Eng.* **2019**, *22*, 2504–2516. [CrossRef]
49. Gholizadeh, S.; Poorhoseini, H. Seismic layout optimization of steel braced frames by an improved dolphin echolocation algorithm. *Struct. Multidiscip. Optim.* **2016**, *54*, 1011–1029. [CrossRef]
50. Hamadi, Y.; Monfroy, E.; Saubion, F. An introduction to autonomous search. In *Autonomous Search*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 1–11. [CrossRef]
51. Soto, R.; Crawford, B.; Olivares, R.; Niklander, S.; Johnson, F.; Paredes, F.; Olguín, E. Online control of enumeration strategies via bat algorithm and black hole optimization. *Nat. Comput.* **2016**, *16*, 241–257. [CrossRef]
52. Soto, R.; Crawford, B.; Olivares, R.; Galleguillos, C.; Castro, C.; Johnson, F.; Paredes, F.; Norero, E. Using autonomous search for solving constraint satisfaction problems via new modern approaches. *Swarm Evol. Comput.* **2016**, *30*, 64–77. [CrossRef]
53. Salto, C.; Alba, E. Designing heterogeneous distributed GAs by efficiently self-adapting the migration period. *Appl. Intell.* **2011**, *36*, 800–808. [CrossRef]

54. Qin, A.; Suganthan, P. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Scotland, UK, 2–5 September 2005; pp. 1785–1791. [CrossRef]
55. Yi, W.; Gao, L.; Li, X.; Zhou, Y. A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. *Appl. Intell.* **2014**, *42*, 642–660. [CrossRef]
56. Han, M.F.; Liao, S.H.; Chang, J.Y.; Lin, C.T. Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. *Appl. Intell.* **2012**, *39*, 41–56. [CrossRef]
57. Liang, K.H.; Yao, X.; Newton, C.S. Adapting self-adaptive parameters in evolutionary algorithms. *Appl. Intell.* **2001**, *15*, 171–180. [CrossRef]
58. Nguyen, T.T.; Vo, D.N. Modified cuckoo search algorithm for short-term hydrothermal scheduling. *Int. J. Electr. Power Energy Syst.* **2015**, *65*, 271–281. [CrossRef]
59. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
60. Kaveh, A.; Farhoudi, N. A new optimization method: Dolphin echolocation. *Adv. Eng. Softw.* **2013**, *59*, 53–70. [CrossRef]
61. Valdivia, S.; Soto, R.; Crawford, B.; Caselli, N.; Paredes, F.; Castro, C.; Olivares, R. Clustering-based binarization methods applied to the crow search algorithm for 0/1 combinatorial problems. *Mathematics* **2020**, *8*, 1070. [CrossRef]
62. Taramasco, C.; Crawford, B.; Soto, R.; Cortés-Toro, E.M.; Olivares, R. A new metaheuristic based on vapor-liquid equilibrium for solving a new patient bed assignment problem. *Expert Syst. Appl.* **2020**, *158*, 113506. [CrossRef]
63. Soto, R.; Crawford, B.; Lanza-Gutierrez, J.M.; Olivares, R.; Camacho, P.; Astorga, G.; de la Fuente-Mella, H.; Paredes, F.; Castro, C. Solving the manufacturing cell design problem through an autonomous water cycle algorithm. *Appl. Sci.* **2019**, *9*, 4736. [CrossRef]
64. Taramasco, C.; Olivares, R.; Munoz, R.; Soto, R.; Villas, M.; de Albuquerque, V.H.C. The patient bed assignment problem solved by autonomous bat algorithm. *Appl. Soft Comput.* **2019**, *81*, 105484. [CrossRef]
65. King, J.R. Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. *Int. J. Prod. Res.* **1980**, *18*, 213–232. [CrossRef]
66. Waghodekar, P.H.; Sahu, S. Machine-component cell formation in group technology: MACE. *Int. J. Prod. Res.* **1984**, *22*, 937–948. [CrossRef]
67. Seifoddini, H.; Wolfe, P.M. Application of the similarity coefficient method in group technology. *IIE Trans.* **1986**, *18*, 271–277. [CrossRef]
68. Chandrasekharan, M.P.; Rajagopalan, R. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *Int. J. Prod. Res.* **1986**, *24*, 451–463. [CrossRef]
69. Mosier, C.; Taube, L. Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega* **1985**, *13*, 577–579. [CrossRef]
70. Chan, H.; Milner, D. Direct clustering algorithm for group formation in cellular manufacture. *J. Manuf. Syst.* **1982**, *1*, 65–75. [CrossRef]
71. Askin, R.G.; Chiu, K.S. A graph partitioning procedure for machine assignment and cell formation in group technologyy. *Int. J. Prod. Res.* **1990**, *28*, 1555–1572. [CrossRef]
72. Stanfel, L.E. Machine clustering for economic production. *Eng. Costs Prod. Econ.* **1985**, *9*, 73–81. [CrossRef]
73. McCormick, S.T.; Pinedo, M.L.; Shenker, S.; Wolf, B. Sequencing in an assembly line with blocking to minimize cycle time. *Oper. Res.* **1989**, *37*, 925–935. [CrossRef]
74. Srinivasan, V. A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Nav. Res. Logist. Q.* **1971**, *18*, 317–327. [CrossRef]
75. Carrie, A.S. Numerical taxonomy applied to group technology and plant layout. *Int. J. Prod. Res.* **1973**, *11*, 399–416. [CrossRef]
76. Kumar, C.S.; Chandrasekkharan, M.P. Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *Int. J. Prod. Res.* **1990**, *28*, 233–243. [CrossRef]
77. Boe, W.J.; Cheng, C.H. A close neighbour algorithm for designing cellular manufacturing systems. *Int. J. Prod. Res.* **1991**, *29*, 2097–2116. [CrossRef]
78. Lilliefors, H.W. On the kolmogorov-smirnov test for normality with mean and variance unknown. *J. Am. Stat. Assoc.* **1967**, *62*, 399–402. [CrossRef]

79. Mann, H.B.; Whitney, D.R. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* **1947**, *18*, 50–60. [CrossRef]
80. Eaton, J.; Bateman, D.; Hauberg, S.; Wehbring, R. *GNU Octave Version 3.8.1 Manual: A High-Level Interactive Language for Numerical Computations*; CreateSpace Independent Publishing Platform: Coates Valley, CA, USA, 2014; ISBN 1441413006.
81. Soto, R.; Crawford, B.; Palma, W.; Monfroy, E.; Olivares, R.; Castro, C.; Paredes, F. Top-kBased adaptive enumeration in constraint programming. *Math. Probl. Eng.* **2015**, *2015*, 580785. [CrossRef]