Solving The Manufacturing Cell Design Problem using Cuckoo Search

Ricardo Soto[1,2,3], Broderick Crawford[1,4,5], Ana Jaime[1], Maykol Ramírez[1], and Boris Almonacid[1]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
[2] Universidad Autónoma de Chile, Santiago, Chile
[3] Universidad Científica del Sur, Lima, Perú
[4] Universidad Central de Chile, Santiago, Chile
[5] Facultad de Ingeniería y Tecnología, Universidad San Sebastián, Bellavista 7, Santiago 8420524, Chile
{ricardo.soto, broderick.crawford}@pucv.cl
{ana.jaime.b, maykol.ramirez.g, boris.almonacid.g}@mail.pucv.cl

Abstract. The Manufacturing Cell Design Problem consists in the division of a manufacturing plant into cells, each one of them containing machines processing a group of parts. The goal is to increase the productivity by minimizing the exchange of material between cells. In this paper, we solve this problem by using Cuckoo Search, which is a easy-to-implement and fast-convergence metaheuristic inspired on the interesting reproduction strategy of cuckoo birds. We perform different experiments on a set of 90 well-known problem instances where our approach is able to reach the global optimum for all of them.

Keywords: Manufacturing Cell Design Problem, Optimization, Cuckoo Search, Metaheuristics

1 Introduction

Nowadays in the industry, specifically in terms of development and creation, there is a constant seek of higher performance in order to increase the benefits and reduce costs. The Manufacturing Cell Design Problem (MCDP) focuses on solving productivity and efficiency concerns in many industries by carefully organizing plants in production cells. Each cell contains machines that process a set of product parts. A solution to the problem provides an optimal path for the product-making process by minimizing the inter-cell moves of product parts among cells.

During the last years, different approches have been proposed to tackle the MCDP ranging from the classic complete search sphere to more recent metaheuristic approaches. For instance, considering complete search techniques we may found Linear Programming [12,11], Goal Programming [14,15] and Quadratic Models [2,6]. Now taking into account approximate methods, Tabu Search [1,7], Simulated Annealing [20], Particle Swarm Optimization [3] and

Genetic Algorithms [4,19] can be found in the literature. Also, more recent solving techniques such as the Shuffled Frog Leaping Algorithm [17] and Migrating Birds Optimization [16] have been reported as well. In this paper, we solve this problem by using Cuckoo Search, which is an efficient and fast-convergence metaheuristic inspired on the reproduction strategy of cuckoo birds. We illustrate promising experimental results where the proposed approach is able to reach the global optimum for a set of 90 well-known MCDP instances.

This paper is structured as follows. Section 2 presents the problem and the corresponding mathematical model. The method to solve the problem and the integration between them is described in Section 3. In Section 4, we present the experimental results followed by conclusions and future work.

2 The Manufacturing Cell Design Problem

The Manufacturing Cell production strategy consists in organizing a production plant into cells, where each cell contains a group of machines and parts processed by them. According to their similarities (similar shape, operation processes, required materials and tools, etc.), these parts may form families and each group of machines inside a cell will be concentrated in processing one of these families. The purpose of the MCDP is to find an optimal layout of a production plant minimizing the flow between cells. This goal is achieved if the generated cells can guarantee the entire fabrication of the product. In case this is not possible, there will be some pieces that will have to visit different cells during their fabrication process.

To achieve the reorganization of the production system, it is necessary to know which are the fabrication routes of each one of the parts i.e. to know which part visits which machines during its fabrication process. The machine-part binary matrix is used to represent this information. The rows of the matrix represent the machines and the columns the pieces or parts. Each component of the matrix is a '1' or a '0' depending if the selected machine processes the selected part or not. This representation does not provide information about the sequence in which the machines are visited nor the amount of available machines [9]. An example of a machine-part matrix is shown in Figure 1. As we said before, this is a binary matrix and if the component of the coordinate (i, j) is 0, it means that machine i does not process part j. On the other hand, if the component of the coordinate (i, j) is 1, it means that machine i does process part j.

Taking the example shown in Figure 1, from the matrix on the left, it could be determined a grouping of parts and machines in cells. The purpose of the MCDP is to organize the family structure so that, as shown in the matrix on the right, the movement of parts between cells occurs as less as possible. For this, it is necessary to generate a logical procedure that allows transforming the initial machine-pat matrix from the left into the machine-part matrix from the right. Thus the cells or groups are formed, and in this case there are 2 cells that are independent from each other.

	P1	P2	P3	P4	P5	P6	P7
M1	1	0	1	0	0	0	1
M2	0	1	0	1	0	1	0
M3	0	1	0	0	0	1	0
M4	1	0	0	0	1	0	1
M5	1	0	1	0	1	0	1

		P6	P4	P2	P7	P1	P5	P3
Cell 1	M5	0	0	0	1	1	1	1
	M1	0	0	0	1	1	0	1
	M4	0	0	0	1	1	1	0
Cell 2	M2	1	1	1	0	0	0	0
	M3	1	0	1	0	0	0	0

Fig. 1. Initial and Final Machine-Part matrix A_{ij}

The MCDP is conceived as a mathematical model with variables, domains and constraints, that must be satisfied in order to find an optimal organization of cells in terms of production costs. A mathematical formulation to the problem is given by Boctor [2] as shown below:

- P, represents the number of parts to be produced.
- M, represents the amount of machines.
- C, represents the number of cells.
- i, index of machines ($i = 1, .., M$)
- j, index of parts ($j = 1, .., P$)
- k, index of cells ($k = 1, .., C$)
- M_{max}, is the maximum amount of machines allowed in one cell.
- $A = [a_{ij}]$, Machine-Part binary matrix, with domain $[0, 1]$ and dimension $M \times P$, where:

$$a_{ij} \begin{cases} 1 & if\ machine\ i\ processes\ part\ j \\ 0 & otherwise \end{cases}$$

- $B = [y_{ik}]$, Machine-Cell binary matrix, with domain $[0, 1]$ and dimension $M \times C$, where:

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

- $C = [z_{jk}]$, Part-Cell binary matrix, with domain $[0, 1]$ and dimension $P \times C$, where:

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

The model of the MCDP presents a minimization objective function, where the goal is to reduce the exchange of material between cells as shown in Eq. 1.

$$Min : \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} z_{jk} (1 - y_{ik}) \tag{1}$$

This objective function is subjected to three constraints as depicted in the following, where Eq. 2 states that each machine belongs to one and only one cell. Eq. 3 guarantee that each part is assigned to one and only one cell, and Eq. 4 determines the maximum number of machines that a cell can contain.

$$\sum_{k=1}^{C} b_{ik} = 1, \forall i \quad (2) \qquad \sum_{k=1}^{C} c_{jk} = 1, \forall j \quad (3) \qquad \sum_{i=1}^{M} b_{ik} \leq M_{max}, \forall k \quad (4)$$

3 Cuckoo Search

Cuckoo Search (CS) is a bio-inspired metaheuristic algorithm has been recently proposed by Yang and Deb [22] and mimics the interesting reproduction strategy of the bird specie called cuckoo. Basically this behavior consists in an obligate brood parasitism: the cuckoos do not build nests, they lay their eggs in the nests of other birds (often other species). First, they look for a potential host nest and when is found, they lay their eggs inside of it. After that, they move to find other nests to lay more eggs. Another distinctive feature from the cuckoos is that some of them can mimic the color of the host eggs to increase the hatchability. However, something can go wrong and the host bird can discover the intruder eggs. If this happens, the host bird may throw the parasitic eggs away or abandon its nest and build a new one. The algorithm which draws inspiration from cuckoos adaption to breeding and reproduction, follows three idealized rules [21]:

1. Each cuckoo lays one egg at a time, and dump its egg in a randomly chosen nest.
2. The best nests with high quality of eggs will carry over to the next generations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $P_a \in [0, 1]$.

In terms of simplicity for the present work, we have two important considerations. Firstly, every egg will be thrown in a different nest, so each nest will contain only one egg. Secondly, the rule number three can be understood as follows: the fraction P_a of the n nests are replaced by new nests (with new random solutions).

It should also be noted that to find new nests, Cuckoos use a special algorithm called Lévy Flight, which simulates the flight pattern of various animals and insects. The greatest feature of Lévy Flight is its random flight, with certain restrictions of what it covers, however, thanks to this feature the search space can be traveled in a much broader way to find an optimum or a close solution to it [5].

3.1 Lévy Flight

Lévy Flight is a random walk where the steps have a power-law distribution, alternating extremely long, short and random jumps to define the trace. The behaviour of some animals when they move and look for food, shows the mathematical pattern used in Lévy Flight. For example, the fruit fly explores the landscape using this pattern [13]. In addition to that, Lévy Flight has been observed in the hunt route from albatross, lions, and spider monkeys. Even humans, unconsciously and by instinct, can move following this pattern [18]. The length of the step in the random walk is drawn from a Lévy Distribution, which has an infinite variance with an infinite mean. This shows a random walk process with a power-law step-length distribution with a heavy tail.[22].

The step size is obtained using the algorithm proposed by Mantegna [8]. In this algorithm, the *step* is calculated as follows:

$$step = \frac{u}{(|v|)^{1/\beta}} \tag{5}$$

In the *step* formula, β is a value between 1 and 2. The elements u and v are drawn from a normal distribution, that is to say:

$$u \sim N(0, \sigma_u^2),\ v \sim N(0, \sigma_v^2) \tag{6}$$

Where the term σ_u is given by the next formula:

$$\sigma_u = [\frac{\Gamma(1+\beta)\ sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]*\beta 2^{(\beta-1)/2}}]^{\frac{1}{\beta}},\ \sigma_v = 1 \tag{7}$$

3.2 Cuckoo Search Algorithm

The basic steps of the cuckoo search algorithm can be summarized on the pseudo code shown below:

Algorithm 1 Cuckoo Search

1: Objective Function Min:$\sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} z_{jk} (1 - y_{ik})$
2: Generate the initial population of de n host nests $X_i (i = 1, 2, ..., n)$
3: **while** (t <Max Generation) or (Optimum found) **do**
4: Get a cuckoo randomly by Lévy Flights;
5: Evaluate its quality/fitness F_i;
6: Choose a nest among n (say j) randomly;
7: **if** $F_i > F_j$ **then**
8: Replace j by the new solution;
9: **end if**
10: Nests are sorted in ascending order according to their fitness F_i;
11: A fraction (P_a) of worse nests are abandoned and new ones are built
12: **end while**
13: Post process results and visualization

This algorithm, first generates an initial population of host nests. Basically, this is an array that represents the initial set of solutions for CS and its length is given by the amount of available nests. This is because, as we said before, each nest can contain only one egg. First, inside each nest there is a random solution (egg). Each one of this solutions contains the matrix A, Y and Z. The matrix A and the constants of the MCDP (M,P,C and M_{max}) are taken from every instance of the problem. The matrix Y is generated randomly, assigning every machine to one cell and then, after a mathematical procedure using the matrix A and the matrix Y, we generate the matrix Z. This results with the necessary elements to form one solution, and after creating several of them, we obtain a randomly generated initial population.

In the main loop of the algorithm, we use Lévy Flights to take one machine or one part and change the cell where it belongs. To get a new solution via Lévy Flights, we take randomly one solution from the initial population. We calculate the *step* length which will be in the range $[0, M]$ if we want to change the matrix Y or in the range $[0, P]$ if we want to change the matrix Z. If the fitness from the modified solution is better than the fitness from another randomly taken solution from the initial population, it will replace it. If after a certain number of tries, Lévy Flights does not make an improvement in the fitness, it will be initialized again with a new random value from the initial population. After all this process, we sort the solutions in ascending order according to the fitness. The fraction of abandoned nests, is given by the value of the constant P_a, which we define in the beginning, indicating with a probability, the amount of worst solutions that will be dropped.

In order to force the step value to move between the interval $[0, M]$ or $[0, P]$, we use a V-Shaped transfer function proposed by Mirjalili [10]. After several experiments, the $V4$ function showed the best results. First, we use it to transform the step size to be a real number inside the interval $[0, 1]$. After that, we multiply that obtained value by M or P, depending on the case, to generate a number inside the desired intervals and choose a machine from matrix Y or a part from matrix Z. A representation of the structures used in our implementation is shown in Figure 2.

4 Experimental Results

The Cuckoo Search algorithm applied to the MCDP, has been implemented in Java and launched in an AMD A10 processor with 8GB RAM running Microsoft Windows 10. To carry out the tests, we have used the matrices drawn from a set of problems studied by F. Boctor [2]. There are 10 16×30 matrices (16 machines and 30 parts) and in each one of them, the number of cells (C) and the maximum of machines allowed in one cell (M_{max}) depends on the instance.

During the testing phase, the behavior of the implemented algorithm was studied under different parameters (P_a, generations, amount of nests n) which were modified one at a time in order to get the best performance from the metaheuristic. Each instance of the problem was executed 31 times. In our

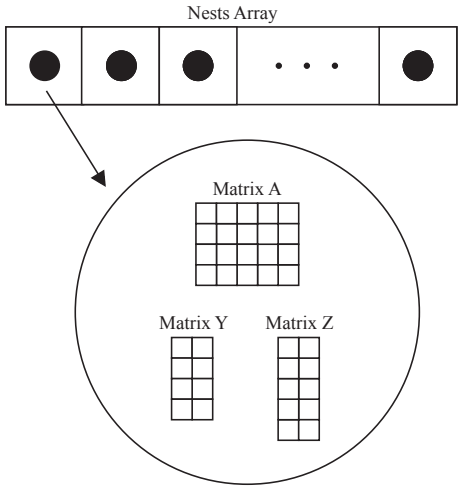

Fig. 2. Representation of CS applied to MCDP

experiments, we got the best results with 25 nests as the initial population, a $P_a = 0.5$ and $50,000$ generations. Results are depicted in tables 4 and 4. We compare our results with four different metaheuristic methods: Simulated Annealing (SA) [20], Shuffled Frog Algorithm [17], Migrating Birds Optimization [16], Particle Swarm Optimization [3]. We employ as quality measure the Relative Percentage Deviation (RPD), which is computed as follows:

$$RDP = \frac{(Z - Z_{opt})}{Z_{opt}} \times 100$$

where Z_{opt} is the best known optimum value and Z is the best optimum value reached by CS. Cuckoo Search able to reach the global optimum for the whole set of tested instances.

Table 1. Experiments using $C = 2$: Optimum values for Cuckoo Search (CS), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Migrating Birds Optimization (MBO), and Shuffled Frog Leaping Algorithm (SFLA).

Boctor	Mmax	Optimum	Cuckoo Search			SA	PSO	MBO	SFLA
Problem		Value	Optimum	Average	RPD%				
1	8	11	11	11,00	0,00	11	11	11	11
1	9	11	11	11,00	0,00	11	11	11	11
1	10	11	11	11,00	0,00	11	11	11	11
1	11	11	11	11,00	0,00	11	11	11	11
1	12	11	11	11,00	0,00	11	11	11	11
2	8	7	7	7,00	0,00	7	7	7	7
2	9	6	6	6,00	0,00	6	6	6	6
2	10	4	4	4,00	0,00	10	5	4	4
2	11	3	3	3,00	0,00	4	4	3	3
2	12	3	3	3,00	0,00	3	4	3	3
3	8	4	4	4,00	0,00	5	5	4	4
3	9	4	4	4,00	0,00	4	4	4	4
3	10	4	4	4,00	0,00	4	5	4	4
3	11	3	3	3,00	0,00	4	4	3	3
3	12	1	1	1,00	0,00	4	3	1	1
4	8	14	14	14,00	0,00	14	15	14	14
4	9	13	13	13,00	0,00	13	13	13	13
4	10	13	13	13,00	0,00	13	13	13	13
4	11	13	13	13,00	0,00	13	13	13	13
4	12	13	13	13,00	0,00	13	13	13	13
5	8	9	9	9,00	0,00	9	10	9	9
5	9	6	6	6,00	0,00	6	8	6	6
5	10	6	6	6,00	0,00	6	6	6	6
5	11	5	5	5,00	0,00	7	5	5	5
5	12	4	4	4,00	0,00	4	5	4	4
6	8	5	5	5,00	0,00	5	5	5	5
6	9	3	3	3,00	0,00	3	3	3	3
6	10	3	3	3,00	0,00	5	3	3	3
6	11	3	3	3,00	0,00	3	4	3	3
6	12	2	2	2,00	0,00	3	4	2	2
7	8	7	7	7,00	0,00	7	7	7	7
7	9	4	4	4,00	0,00	4	5	4	4
7	10	4	4	4,00	0,00	4	5	4	4
7	11	4	4	4,00	0,00	4	5	4	4
7	12	4	4	4,00	0,00	4	5	4	4
8	8	13	13	13,00	0,00	13	14	13	13
8	9	10	10	10,00	0,00	20	11	10	10
8	10	8	8	8,00	0,00	15	10	8	8
8	11	5	5	5,00	0,00	11	6	5	5
8	12	5	5	5,00	0,00	7	6	5	5
9	8	8	8	8,00	0,00	13	9	8	8
9	9	8	8	8,00	0,00	8	8	8	8
9	10	8	8	8,00	0,00	8	8	8	8
9	11	5	5	5,00	0,00	8	5	5	5
9	12	5	5	5,00	0,00	8	8	5	5
10	8	8	8	8,00	0,00	8	9	8	8
10	9	5	5	5,00	0,00	5	8	5	5
10	10	5	5	5,00	0,00	5	7	5	5
10	11	5	5	5,00	0,00	5	7	5	5
10	12	5	5	5,00	0,00	5	6	5	5

Table 2. Experiments using $C = 3$:: Optimum values for Cuckoo Search (CS), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Migrating Birds Optimization (MBO), and Shuffled Frog Leaping Algorithm (SFLA).

Boctor	Mmax	Optimum	Cuckoo Search			SA	PSO	MBO	SFLA
Problem		Value	Optimum	Average	RPD%				
1	6	27	27	28,00	0,00	28	-	27	-
1	7	18	18	20,00	0,00	18	-	18	-
1	8	11	11	14,00	0,00	11	-	11	-
1	9	11	11	13,00	0,00	11	-	11	-
2	6	7	7	11,00	0,00	7	-	7	-
2	7	6	6	8,00	0,00	6	-	6	-
2	8	6	6	7,00	0,00	7	-	6	-
2	9	6	6	7,00	0,00	6	-	6	-
3	6	9	9	10,00	0,00	12	-	9	-
3	7	4	4	7,00	0,00	8	-	4	-
3	8	4	4	5,00	0,00	8	-	4	-
3	9	4	4	5,00	0,00	4	-	4	-
4	6	27	27	27,00	0,00	27	-	27	-
4	7	18	18	19,00	0,00	18	-	18	-
4	8	14	14	15,00	0,00	14	-	14	-
4	9	13	13	15,00	0,00	13	-	13	-
5	6	11	11	12,00	0,00	11	-	11	-
5	7	8	8	11,00	0,00	9	-	8	-
5	8	8	8	10,00	0,00	9	-	8	-
5	9	6	6	8,00	0,00	8	-	6	-
6	6	6	6	9,00	0,00	8	-	6	-
6	7	4	4	6,00	0,00	5	-	4	-
6	8	4	4	5,00	0,00	5	-	4	-
6	9	3	3	5,00	0,00	4	-	3	-
7	6	11	11	14,00	0,00	11	-	11	-
7	7	5	5	8,00	0,00	5	-	5	-
7	8	5	5	7,00	0,00	5	-	5	-
7	9	4	4	7,00	0,00	5	-	4	-
8	6	14	14	16,00	0,00	14	-	14	-
8	7	11	11	15,00	0,00	11	-	11	-
8	8	11	11	14,00	0,00	11	-	11	-
8	9	10	10	12,00	0,00	10	-	10	-
9	6	12	12	17,00	0,00	12	-	12	-
9	7	12	12	14,00	0,00	12	-	12	-
9	8	8	8	11,00	0,00	13	-	8	-
9	9	8	8	11,00	0,00	8	-	8	-
10	6	10	10	13,00	0,00	12	-	10	-
10	7	8	8	11,00	0,00	14	-	8	-
10	8	8	8	9,00	0,00	8	-	8	-
10	9	5	5	8,00	0,00	8	-	5	-

4.1 Convergence to the Best Solution

The convergence when solving the fourth problem from Boctor [2] with an $M_{max} = 8$ is shown next in Figure 3 and 4. The chart from Figure 3 corresponds to the instance with $C = 2$ and the one from Figure 4 with $C = 3$. For both of them, we can see that in the first iteration, the best current value changes abruptly from higher to lower values. The optimum for the instance with $C = 2$ is reached quite early, in the generation number 449. On the other hand, we can see that after the first iteration, the instance with $C = 3$ changes gradually and $44,330$ generations were needed to reach the global optimum.

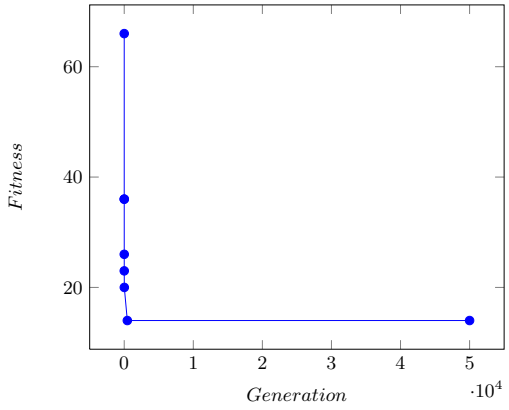

Fig. 3. Convergence graphic of the problem 4 with $M_{max} = 8$ and $C = 2$

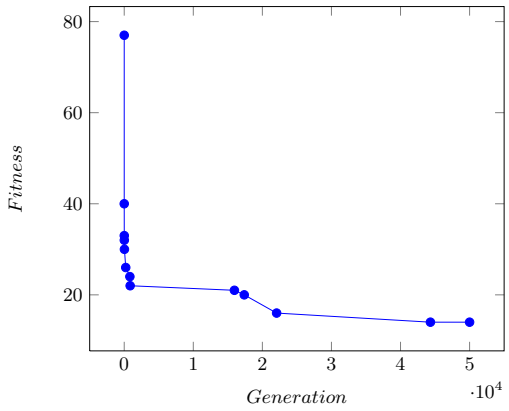

Fig. 4. Convergence graphic of the problem 4 with $M_{max} = 8$ and $C = 3$

5 Conclusions and Future work

In this paper we have presented a new Cuckoo Search algorithm for solving the MCDP. The metaheuristic is quite simple to implement and can fast be configured for reaching good results. Computational experiments have been conducted in 90 well-known MCDP instances, considering different number of cells, machines, parts and machines admitted in cells. The proposed algorithm was able to succeed in reaching the global optimum for all tested configurations in a reasonable amount of iterations. The analysis of solving processes have exhibited the rapid convergence of the algorithm abruptly diminishing the fitness of solutions while maintaining the robustness illustrated by the average values. As future work, we plan to experiment with parameter tuning and to implement new and modern metaheuristics to solve the MCDP. The integration of online control to the presented approach would be another line of research to follow in the short-term.

6 Acknowledgements

Ricardo Soto is supported by Grant CONICYT / FONDECYT / REGULAR / 1160455. Broderick Crawford is supported by Grant CONICYT / FONDECYT / REGULAR / 1130455. Boris Almonacid is supported by Postgraduate Grant Pontificia Universidad Católica de Valparaíso 2015 (INF-PUCV 2015).

References

1. Aljaber, N., Baek, W., Chen, C.L.: A tabu search approach to the cell formation problem. Computers & industrial engineering 32(1), 169–185 (1997)
2. Boctor, F.F.: A jinear formulation of the machine-part cell formation problem. THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 29(2), 343–356 (1991)
3. Durán, O., Rodriguez, N., Consalter, L.A.: Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. Expert Systems with Applications 37(2), 1563–1567 (2010)
4. Gupta, Y., Gupta, M., Kumar, A., Sundaram, C.: A genetic algorithm-based approach to cell composition and layout design problems. International Journal of Production Research 34(2), 447–482 (1996)
5. Gutowski, M.: L\'evy flights as an underlying mechanism for global optimization algorithms. arXiv preprint math-ph/0106003 (2001)
6. Kusiak, A., Chow, W.S.: Efficient solving of the group technology problem. Journal of manufacturing systems 6(2), 117–124 (1987)
7. Lozano, S., Adenso-Diaz, B., Eguia, I., Onieva, L., et al.: A one-step tabu search algorithm for manufacturing cell design. Journal of the Operational Research Society 50(5), 509–516 (1999)
8. Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. Physical Review E 49(5), 4677 (1994)

9. Medina, P.D., Cruz, E.A., Pinzón, M.: Generación de celdas de manufactura usando el algoritmo de ordenamiento binario (aob). Scientia et Technica 1(44), 106–110 (2010)
10. Mirjalili, S., Lewis, A.: S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation 9, 1–14 (2013)
11. Oliva-Lopez, E., Purcheck, G.: Load balancing for group technology planning and control. International Journal of Machine Tool Design and Research 19(4), 259–274 (1979)
12. Purcheck, G.F.: A linear–programming method for the combinatorial grouping of an incomplete power set. Journal of Cybernetics (1975)
13. Reynolds, A.M., Frye, M.A.: Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search. PloS one 2(4), e354 (2007)
14. Sankaran, S., Rodin, E.Y.: Multiple objective decision making approach to cell formation: a goal programming model. Mathematical and Computer Modelling 13(9), 71–81 (1990)
15. Shafer, S.M., Rogers, D.F.: A goal programming approach to the cell formation problem. Journal of Operations Management 10(1), 28–43 (1991)
16. 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, pp. 270–281. Springer (2015)
17. 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), November 28-30, 2015, Beni Suef, Egypt. pp. 253–261. Springer (2016)
18. Tran, T., Nguyen, T.T., Nguyen, H.L.: Global optimization using lévy flights. arXiv preprint arXiv:1407.5739 (2014)
19. Venugopal, V., Narendran, T.: A genetic algorithm approach to the machine-component grouping problem with multiple objectives. Computers & Industrial Engineering 22(4), 469–480 (1992)
20. Wu, T.H., Chang, C.C., Chung, S.H.: A simulated annealing algorithm for manufacturing cell formation problems. Expert Systems with Applications 34(3), 1609–1617 (2008)
21. Yang, X.S.: Nature-inspired metaheuristic algorithms. Luniver press (2010)
22. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. pp. 210–214. IEEE (2009)