Memetic chicken swarm algorithm for job shop scheduling problem

Soukaina Cherif Bourki Semlali, Mohammed Essaid Riffi, and Fayçal Chebihi
Laroseri Laboratory, Department of Computer Sciences, Faculty of Sciences,
University of Chouaib Doukkali, El Jadida, Morocco

ABSTRACT
This paper presents a Memetic Chicken swarm optimization (MeCSO) to solve job shop scheduling problem (JSSP). The aim is to find a better solution which minimizes the maximum of the completion time also called Makespan. In this paper, we adapt the chicken swarm algorithm which take into consideration the hierarchical order of chicken swarm while seeking for food. Moreover, we integrate 2-opt method to improve the movement of the rooster. The new algorithm is applied on some instances of OR-Library. The empirical results show the forcefulness of MeCSO comparing to other metaheuristics from literature in term of run time and quality of solution.

1. INTRODUCTION
The job-shop scheduling problem (JSSP) was formulated for the first time by Muth and Thompson in 1963. The JSSP is one of the NP-Hard problems [1] and the most known of the classical scheduling problems in the context of manufacturing [2], which help to improve competitiveness of many companies and organizations. The aim purpose of the job-shop scheduling problem is to find a schedule which minimizes the time required to complete a group of jobs (the makespan).

Historically, several algorithms are proposed in literature to solve the job shop scheduling problem by optimizing the makespan such as: branch and bound (BB)[3], simulated annealing (SA) [4], Tabu search (TS)[5][6], genetic algorithms (GA)[7][8][9], neural networks (NN)[10], ant colony optimization (ACO)[11], Particle swarm optimization (PSO)[12], Bee colony optimization (BCO)[13] and firefly algorithm (FA) [14]. Additionally, some researchers have developed an hybrid optimization strategy for JSSP such as parallel GRASP with path-relinking[15] and new hybrid genetic algorithm [16].

2. JOB-SHOP SCHEDULING PROBLEM
The JSSP can be briefly introduced [17] as a sequential allocation of a production schedule for a given set of jobs and resources that optimizes the completion time of all jobs which helps to minimize the makespan. As result, the makespan (the maximum job completion time) $C_{max}$ is the duration between the time of completion of last job and the starting time of the first job (1).

$$C_{max} = \max_{t_{ij}} (t_{ij} + p_{ij})$$

Where $t_{ij}$ is denoted as the starting time and $p_{ij}$ as the uninterrupted processing time.
The JSSP can be formulated by assigning a set of \( n \) jobs \( J = \{J_1, \ldots, J_n\} \) to a set of \( m \) machines \( M = \{M_1, \ldots, M_m\} \), each machine can process at most one operation at a time. As well, each job consists of a set of \( O_{ik} \), which contains \( m \) operations where \( i \) denotes the job of a specific operation and \( k \) represents the current machine \( M_k \). Each operation must be processed during an uninterrupted period of time on a given machine. In the JSSP, the order and the uninterrupted processing time must be taken into consideration.

The schedule as a solution for the JSSP can be modeled as a vector of a sequence of operation \( (C_{11}, C_{j1}, \ldots, C_{nm+1}) \) then the main goal is to find the minimum time of all processes, the problem is formulated as follows:

\[
\min C_{nm+1} \tag{2}
\]

Where

\[
C_{kl} \leq C_{ji} - d_{kl} ; j = 1, \ldots, n; i = 1, \ldots, m; kl \in P_{ji} \tag{3}
\]

\[
\sum_{j \in O(t)} r_{ji} \leq 1; i \in M; t \geq 0 \tag{4}
\]

\[
C_{ji} \geq 0; j = 1, \ldots, n; i = 1, \ldots, m \tag{5}
\]

The constraint (2) minimizes the finish time of operation \( o_{nm+1} \) (the makespan).

The constraint (3) represents the fact that between operations the precedence relations should be respected.

The constraint (4) describes that each machine can process one operation at a time.

The constraint (5) guarantees that the finish times to be positive.

The remainder of this paper is organised as follows: The section 2 represents the literature review of the problem. The Section 3 describes the proposed memetic-CSO algorithm. The Section 4 presents the results of the experimental study. The Section 5 gives a discussion of the empirical results. Finally, the Section 6 gives the conclusion and the prospects for further works.

3. FORMULATION OF THE PROBLEM

In the job-shop scheduling problem (JSSP), the solution can be depicted as a sequence of \( n \times m \) operations, which optimizes the completion time of all jobs and then helps to find a schedule with minimum makespan.

Let’s consider the following example with \( m = 3 \) machines and \( n = 3 \) jobs, where: \( J = \{\text{Job0}, \text{Job1}, \text{Job2}\} \) and \( M = \{0, 1, 2\} \)

\[
\text{Job0} = \{(0; 3), (1; 2), (2; 2)\}
\]

\[
\text{Job1} = \{(0; 2), (2; 1), (1; 4)\}
\]

\[
\text{Job2} = \{(1; 4), (2; 3)\}
\]

The representation of the matrix will be as bellow:

\[
\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 \\
0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 \\
0 & 1 & 2 & 0 & 2 & 1 & 1 & 2 \\
3 & 2 & 2 & 2 & 1 & 4 & 4 & 3
\end{pmatrix}
\]

The first line contains the operation number, the second line contains the job number, the third line contains the sequence number, the forth line contains the machine number and the last line contains the processing time of each operation.

As indicated in the Gantt chart representation Fig.1, the solution \( S = \{0, 6, 3, 4, 1, 5, 2, 7\} \) is given by a permutation of a set of operations on each machine, in this example the minimum makespan \( C_{\text{max}}=11 \). In this paper, the chicken can search food in a set of solutions \( S \) defined as the search space.
4. CHICKEN SWARM OPTIMIZATION

The Chicken swarm optimization (CSO) was introduced by Meng, X.B. And al. [18] and inspired by the behavior of a chicken swarm while searching for food. Each swarm is divided into several groups, which comprises one rooster, hens and chicks. The hierarchical order in the swarm is established by the fitness value. We refer the number of roosters, hens, chicks and mother hens by RN, HN, CN and MN.

The position update equation of the rooster can be formulated as:

\[
x_{i,j}^{t+1} = x_{i,j}^t \times (1 + \text{Randn}(0,\sigma^2))
\]  

\[
\sigma^2 = \begin{cases} 
1, & \text{if } f_i \leq f_k, \\
\exp\left(\frac{f_k - f_i}{|f_i| + \varepsilon}\right), & \text{otherwise } k \in [1,N], k \neq i
\end{cases}
\]  

where

- \text{Randn}(0,\sigma^2) is a Gaussian distribution
- \sigma^2 is a standard deviation
- The rooster index \( k \) is randomly selected from the rooster’s group.
- \( f \) is the fitness value of the corresponding \( x \).

The position update equation of the hen can be formulated as bellow:

\[
x_{i,j}^{t+1} = x_{i,j}^t + S1 \times \text{Rand} \times (x_{r1,j}^t - x_{i,j}^t) + S2 \times \text{Rand} \times (x_{r2,j}^t - x_{i,j}^t)
\]

and

\[
S1 = \exp\left(\frac{f_i - f_r1}{|f_i| + \varepsilon}\right) \text{ and } S2 = \exp\left((f_r2 - f_i)\right)
\]

where \text{Rand} \in [0, 1], \( r_1 \) is the index of the rooster and \( r_2 \) is the index of a random chicken from the swarm (where \( r_1 \neq r_2 \)).

Finally, the position update equation of the chick is formulated in [19] as follows:

\[
x_{i,j}^{t+1} = W \times x_{i,j}^t + FL \times (x_{m,j}^t - x_{i,j}^t) + C \times (x_{r,j}^t - x_{i,j}^t)
\]

Where \( W \) is a self-learning factor for chicks, \( FL \in [0, 2] \) is a randomly selected parameter to refer to the relationship between the chicks and its mother with the index \( m \) where \( m \in [1,N] \). Otherwise, \( C \) is a learning- factor from the rooster with the index \( r \).
5. ADAPTATION OF CHICKEN SWARM ALGORITHM TO JOB SHOP SCHEDULING PROBLEM

During the discretization of the original version of the chicken swarm algorithm in order to solve the job shop scheduling problem, the redefinition of operators is represented by the subtraction $\ominus$, the multiplication $\otimes$ and the addition $\oplus$ used in the original version [19].

Furthermore, we used the uniform crossover (UX) [20] in the position update equation of hens and chicks for the movement towards the leaders of groups and the sequential constructive crossover (SCX) [21] to simulate the movement towards the neighbors. $\odot$ operator represents the crossover operator and $\otimes$ operator as applying the chosen crossover to the equation. The addition operator indicates that the randomly chosen crossover is applied to the movement. The application of UX and SCX ensure the competition between groups in the swarm.

As well, we integrate the 2-opt neighborhood operator to realize the auto-improvement mechanism in the position equation of the roosters and the chicks. In this new adaptation each schedule of a group is chosen randomly. The MeCSO in pseudo-code is represented by algorithm 1.

Algorithm 1: MeCSO for jssp
1. Initialize P the size of swarm
2. Generate P chickens
3. initialize parameters: P, G, FL, C and w.
4. Evaluate the fitness values at t=0 for each chicken
5. Rank and establish a hierarchal order
6. Create groups and assign chicks to mother-hens
7. Update the position by equations 6, 8 and 9
8. Update the new solution if the fitness value is better.
9. Rank if G is reached until stop criterion.
10. Return results of MeCSO

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1. Experimental environment

The proposed algorithm MeCSO was coded in python and run on a DELL in visual studio 2017 and simulated with Intel(R) Core(TM) i7-6500 U CPU 2.5GHz (4 CPUs) 2.6 GHz and 16.00 GB of RAM and Microsoft Windows 10 Professional (64-bit) operating system. The performance of MeCSO was tested on different instances of OR-Library [22] 20 times in 100 iterations.

6.2. Default parameters

The table 1 shows the parameter values used in the new adaptation MeCSO. We execute different tests on instances Abz5 and Orb1 in order to choose the values which guarantee to obtain good results and converge towards the global optimum.

Table 1. The Parameters for the Memetic-CSO Algorithm

| Parameters of MeCSO          | Values |
|------------------------------|--------|
| Population size P            | 500    |
| Number of roosters (RN)      | 12     |
| Number of hens (HN)          | 25     |
| Number of chicks (CN)        | 63     |
| Number of iterations to update the algorithm (G) | 10 |
| Self-learning factor (W)     | 0.5    |
| Learning factor from the mother hens (FL) | 0.4 |
| Learning factor from the rooster (C) | 0.65 |
where

\[ CN = P - (NR + NH) \]  

(10)

6.3. RESULTS AND DISCUSSION:

We applied MeCSO on some instances of OR-library, the table 2 summarizes the obtained results of 20 runs. The first column represents different instances instance in OR-Library, the second column indicates the best Known solution (BKS), the third column describes the average of the best found solution \( \delta_{avg} \), the remaining columns represent the measures use to perform the quality of the solution. The proposed algorithm MeCSO allows to find the best-known solution about 51.08 \% from all tested instances.

Table 2. Numerical Results by MeCSO Applied to Some Instances of OR-library

| n × m | Instance | BKS | \( \delta_{avg} \) | \( T_{avg} \) (s) | Err(\%) |
|-------|----------|-----|-----------------|---------------|---------|
| 10 × 10 | Abz5 | 1234 | 1236 | 139 | 0.068 |
| 10 × 10 | Abz6 | 943 | 949 | 521 | 0.821 |
| 10 × 10 | Orb1 | 1059 | 1093 | 807 | 1.045 |
| 10 × 10 | Orb2 | 888 | 907 | 157 | 0.981 |
| 10 × 10 | Orb3 | 1005 | 1011 | 408 | 0.056 |
| 10 × 10 | Orb4 | 1005 | 1024 | 101 | 0.328 |
| 10 × 10 | Orb5 | 887 | 891 | 633 | 0.766 |
| 10 × 10 | Orb6 | 1010 | 1016 | 412 | 0.831 |
| 10 × 10 | Orb7 | 397 | 402 | 595 | 0.907 |
| 10 × 10 | Orb8 | 899 | 907 | 276 | 0.837 |
| 10 × 10 | Orb9 | 934 | 944 | 166 | 0.741 |
| 6 × 6 | Ft06 | 55 | 55 | 1 | 0 |
| 10 × 10 | Ft10 | 930 | 939 | 102 | 0.801 |
| 10 × 5 | LA01 | 666 | 666 | 1 | 0 |
| 10 × 5 | LA02 | 655 | 655 | 1 | 0 |
| 10 × 5 | LA03 | 597 | 599 | 21 | 0.086 |
| 10 × 5 | LA04 | 590 | 590 | 3 | 0 |
| 10 × 5 | LA05 | 593 | 593 | 1 | 0 |
| 15 × 5 | LA06 | 926 | 926 | 1 | 0 |
| 15 × 5 | LA07 | 890 | 890 | 2 | 0 |
| 15 × 5 | LA08 | 863 | 863 | 1 | 0 |
| 15 × 5 | LA09 | 951 | 951 | 1 | 0 |
| 15 × 5 | LA10 | 958 | 958 | 1 | 0 |
| 20 × 5 | LA11 | 1222 | 1222 | 2 | 0 |
| 20 × 5 | LA12 | 1039 | 1039 | 1 | 0 |
| 20 × 5 | LA13 | 1150 | 1150 | 1 | 0 |
| 20 × 5 | LA14 | 1292 | 1292 | 1 | 0 |
| 20 × 5 | LA15 | 1207 | 1207 | 3 | 0 |
| 10 × 10 | LA16 | 945 | 950 | 12 | 0.551 |
| 10 × 10 | LA17 | 784 | 784 | 84 | 0 |
| 10 × 10 | LA18 | 848 | 851 | 534 | 0.021 |
| 10 × 10 | LA19 | 842 | 850 | 126 | 0.352 |
| 10 × 10 | LA20 | 902 | 911 | 782 | 0.045 |
| 15 × 10 | LA21 | 1046 | 1085 | 477 | 0.755 |

The mathematical formulation of the percentage of error ERR (11) is represented as bellow:

Memetic chicken swarm algorithm... (Soukaina Cherif Bourki Semlali)
\[ \text{Err} = \frac{(\delta_{\text{avg}} - \text{BKS})}{\text{BKS}} \times 100 \]  

(11)

where \( \text{BKS} \) is the best known value, \( \delta_{\text{avg}} \) the average of the best found solution.

The proposed algorithm MeCSO seems to be promising to solve jssp in a reasonable time compared to GB algorithm [23] as represented in Fig.2. Furthermore, the algorithm allows to obtain good results in term of the global optimum compared to other algorithms from literature, such as [24] and [25] as represented in table 3 and GB algorithm [23] as represented in Fig.3.

Table 3. Average of the BFS of Some Algorithms in the Literature Compared to MeCSO

| Instance                  | MeCSO | Bondal(GA) [25] |
|---------------------------|-------|-----------------|
| Abz5                      | 1236  | 1339            |
| Abz6                      | 949   | 1043            |
| Ft06                      | 55    | 55              |
| Ft10                      | 1099  | 944             |
| LA01                      | 666   | 666             |
| LA02                      | 655   | 658             |
| LA03                      | 599   | 603             |
| LA04                      | 638   |                 |
| LA05                      | 590   | 590             |
| LA16                      | 950   | 977             |
| 666                       |       |                 |
| 619                       |       |                 |
| 593                       |       |                 |

Figure 2. Average time (s) of MeCSO and GB algorithm
7. CONCLUDING REMARKS

In this paper, we proposed a Memetic Chicken swarm optimization algorithm based on the original version of chicken swarm optimization (CSO) and 2-opt mechanism in order to solve the job shop scheduling problem. The empirical results show that MeCSO algorithm is efficient to solve this type of problem than the other algorithms from literature such as GB algorithm and GA in term of the quality of solutions and the computing time. In further research, we suggest to integrate the simulating annealing with the chicken swarm algorithm to ensure the redistribution of the swarm.

REFERENCES

[1] J. Adams, E. Balas, and D. Zawack, “The shifting bottleneck procedure for job shop scheduling,” Management science, vol. 34, no. 3, pp. 391–401, 1988.
[2] A. S. Jain and S. Meeran, “Deterministic job-shop scheduling: Past, present and future,” European journal of operational research, vol. 113, no. 2, pp. 390–434, 1999.
[3] P. Brucker, B. Jurisch, and B. Sievers, “A branch and bound algorithm for the job-shop scheduling problem,” Discrete applied mathematics, vol. 49, no. 1-3, pp. 107–127, 1994.
[4] M. Kolonko, “Some new results on simulated annealing applied to the job shop scheduling problem,” European Journal of Operational Research, vol. 113, no. 1, pp. 123–136, 1999.
[5] M. Dell’Amico and M. Trubian, “Applying tabu search to the job-shop scheduling problem,” Annals of Operations research, vol. 41, no. 3, pp. 231–252, 1993.
[6] C. Y. Zhang, P. Li, Y. Rao, and Z. Guan, “A very fast ts/sa algorithm for the job shop scheduling problem,” Computers & Operations Research, vol. 35, no. 1, pp. 282–294, 2008.
[7] R. Cheng, M. Gen, and Y. Tsujimura, “A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation,” Computers & industrial engineering, vol. 30, no. 4, pp. 983–997, 1996.
[8] Y. Wang et al., “A new hybrid genetic algorithm for job shop scheduling problem,” Computers & Operations Research, vol. 39, no. 10, pp. 2291–2299, 2012.
[9] L. Asadzadeh, “A local search genetic algorithm for the job shop scheduling problem with intelligent agents,” Computers & Industrial Engineering, vol. 85, pp. 376–383, 2015.
[10] S. Y. Foo, Y. Takefuji, and H. Szu, “Scaling properties of neural networks for job-shop scheduling,” Neurocomputing, vol. 8, no. 1, pp. 79–91, 1995.
[11] H. Nazir et al., “Solving job shop scheduling problem using an ant colony algorithm,” Journal of Asian Scientific Research, vol. 5, no. 5, pp. 261–268, 2015.
[12] D. Sha and C.-Y. Hsu, “A hybrid particle swarm optimization for job shop scheduling problem,” Computers & Industrial Engineering, vol. 51, no. 4, pp. 791–808, 2006.
[13] S. Sundar, P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon, “A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint,” Soft Computing, vol. 21, no. 5, pp. 1193–1202, 2017.
[14] A. Khadwilard, S. Chansombat, T. Theppakorn, P. Thapatsuwan, W. Chainate, and P. Pongcharoen, “Ap-
plication of firefly algorithm and its parameter setting for job shop scheduling,” *J. Ind. Technol*, vol. 8, no. 1, 2012.

[15] R. M. Aiex, S. Binato, and M. G. Resende, “Parallel grasp with path-relinking for job shop scheduling,” *Parallel Computing*, vol. 29, no. 4, pp. 393–430, 2003.

[16] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende, “A hybrid genetic algorithm for the job shop scheduling problem,” *European journal of operational research*, vol. 167, no. 1, pp. 77–95, 2005.

[17] S. French, *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Ellis Horwood Ltd, Publisher, 1982.

[18] X. Meng, Y. Liu, X. Gao, and H. Zhang, “A new bio-inspired algorithm: chicken swarm optimization,” in *International Conference in Swarm Intelligence*, pp. 86–94, Springer, 2014.

[19] D. Wu, F. Kong, W. Gao, Y. Shen, and Z. Ji, “Improved chicken swarm optimization,” in *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2015 IEEE International Conference on, pp. 681–686, IEEE, 2015.

[20] D. M. Tate and A. E. Smith, “A genetic approach to the quadratic assignment problem,” *Computers & Operations Research*, vol. 22, no. 1, pp. 73–83, 1995.

[21] Z. Ahmed, “A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem,” 2014.

[22] “http://people.brunel.ac.uk/ mastjjb/jeb/orlib/jobshopinfo.html.”

[23] F. Sayoti, M. E. Riffi, and H. Labani, “Optimization of makespan in job shop scheduling problem by golden ball algorithm,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 4, no. 3, pp. 542–547, 2016.

[24] A. Udomsakdigool and V. Kachitvichyanukul, “Multiple colony ant algorithm for job-shop scheduling problem,” *International Journal of Production Research*, vol. 46, no. 15, pp. 4155–4175, 2008.

[25] A. A. Bondal, *Artificial immune systems applied to job shop scheduling*. PhD thesis, Ohio University, 2008.