Journal of Theoretical and Applied Information Technology
20th February 2015. Vol.72 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CAT SWARM OPTIMIZATION TO SOLVE FLOW SHOP SCHEDULING PROBLEM

[1] ABDELHAMID BOUZIDI, [2] MOHAMMED ESSAID RIFFI
lab. LAROSERI, Depart. of Computer Science. Chouaib Doukkali University, EL JADIDA, MOROCCO
E-Mail : [1] mr.abdelhamid.bouzidi@gmail.com , [2] said@riffi.fr

Abstract :

Flow shop problem is a NP-hard combinatorial optimization problem. Its application appears in logistic, industrial and other fields. It aims to find the minimal total time execution called makespan. This research paper propose a novel adaptation never used before to solve this problem, by using computational intelligence based on cats behavior, called Cat Swarm Optimization, which is based on two sub modes, the seeking mode when the cat is at rest, and the tracing mode when the cat is at hunt. These two modes are combined by the mixture ratio. The operations, operators will be defined and adapted to solve this problem. To prove the performance of this adaptation, some real instances of OR-Library are used. The obtained results are compared with the best-known solution found by others methods.

Keyword : *Flow shop, scheduling, makespan, computational intelligent, cat swarm optimization.*

1. INTRODUCTION

The Flow shop scheduling [1] is one of the known problems in operational research. Given the whole applications fields, and the complexity of the problem, it has been a very active and prolific research area. To resolve this problem we should find the minimal make span by executing n jobs in m machine.

Many optimization algorithms based on computational intelligence had been proposed to solve the flow shop-scheduling problem, such as simulated annealing [2-3], tabu search [3-5], harmony search [6-7], genetic algorithm [8-9], Ant Colony optimization [10-11], bee colony optimization [12], particle swarm optimization [13-15], and others.

The present research paper aims to apply cat swarm algorithm never used before to solve FSSP. The research paper is organized as follows: in section II, a presentation and formulation of flow shop scheduling problem. In section III, a description of cat swarm optimization algorithm. In section VI, Cat swarm optimization applied to FSSP, and the results obtained by using some instance of OR-Library [21]. Finally, the conclusion and discussion.

2. FLOW SHOP SCHEDULING PROBLEM

2.1 PRESENTATION

The flow shop-scheduling problem (FSSP) is a combinatorial optimization problem in class NP-HARD [16], simulated first in 1954 by Johnson [17]. FSSP is a set of $\boldsymbol{n}$ unrelated jobs that should be processed in the same order as m machines. The problem is to find the schedule of jobs that have the best minimal total time of execution of all the process called make span, by respecting some constraints, which are:

- All jobs are independent, and available for processing at time zero.
- The machines are continuously available from time zero onwards
- Each machine can process one operation at a time.
- Each job can be manufactured at a specific moment on a single machine
- If a machine is not available, all the following jobs are assigned to a waiting queue.
- The processing of a given job in a machine cannot be interrupted once started.

A comprehensive list of these constraints, are grouped on categories, can be found in [1].

Setup times are sequence independent and are included in the processing.

2.2 FORMULATION OF PROBLEM:

The FSSP is composed of $\boldsymbol{n}$ job $\boldsymbol{J} = \{\boldsymbol{J_1}, \boldsymbol{J_2} \dots \boldsymbol{J_n}\}$, and $\boldsymbol{m}$ machine $\boldsymbol{M} = \{\boldsymbol{M_1}, \boldsymbol{M_2} \dots \boldsymbol{M_m}\}$, each job is composed of $\boldsymbol{m}$ distinct operations $\boldsymbol{O} = \{\boldsymbol{O_1}, \boldsymbol{O_2} \dots \boldsymbol{O_m}\}$. The operation in each job should respect the sequence of machine. Every operation is represented by a pair m_{o_k} and t_{o_k} (k∈ [1, (n*m)]), where m_{o_k} represents the machine on which the process o_k will be

Journal of Theoretical and Applied Information Technology
20th February 2015. Vol.72 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

executed, and t_{o_k} represents the processing time of operation o_k.

In order to apply CSO to the FSSP, it should be encoded with a generic solution to the problem. For n-jobs and m-machines, the solution is presented by a sequence of ***n*** jobs. The matrix INFO in fig.1 has ***m*n*** columns and four lines, this matrix is developed to represent information about each operation:

O_i: The number of operations in schedule ($i \in [1,(n*m)]$).

J_{o_i}: The job belonging to the operation o_i

M_{o_i}: The machine name where the operation o_i is processed.

T_{o_i}: The processing time of operation o_i.

$$\begin{pmatrix} o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 \\ J_{o_1} & J_{o_2} & J_{o_3} & J_{o_4} & J_{o_5} & J_{o_6} & J_{o_7} & J_{o_8} & J_{o_9} \\ M_{o_1} & M_{o_2} & M_{o_3} & M_{o_4} & M_{o_5} & M_{o_6} & M_{o_7} & M_{o_8} & M_{o_9} \\ T_{o_1} & T_{o_2} & T_{o_3} & T_{o_4} & T_{o_5} & T_{o_6} & T_{o_7} & T_{o_8} & T_{o_9} \end{pmatrix}$$

Fig. 1: Information matrix

For example, let's consider the following: 4*3 FSSP, where $n=3$, $m=3$, $J=\{J_1,J_2,J_3\}$, $M=\{M_1,M_2,M_3\}$, and for every J_i in J, $\boldsymbol{J_i=\{(m_{ik}, t_{ik})\}}$ for $k \in [1,3]$,

J1 = {(1, 6), (2, 1), (3, 4)}
J2 = {(1, 3, (2, 6)), (3, 2)}
J3 = {(1, 1), (2, 2), (3, 1)}
J4 = {(1, 2), (2, 1), (3, 5)}

The representation of matrix of information will be as following:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 6 & 1 & 4 & 3 & 6 & 2 & 1 & 2 & 1 & 2 & 1 & 5 \end{pmatrix}$$

Fig. 2: The information matrix of schedule to be used

A random solution is follow:

3	4	2	1

Fig. 3: An example of solution representation

The make span of solution in Fig.3 according to the rules of FSSP, is 18, it's indicated by GANT chart in fig.4, where M_i $(1 \le i \le 3)$ represents the machines, and each color represents the jobs.

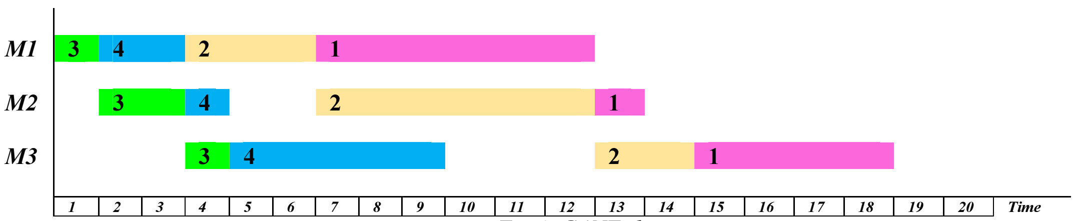

Fig.4: GANT chart

3. CAT SWARM OPTIMIZATION

Cat swarm optimization (CSO) algorithm was first introduced by Chu and Tsai [18] in 2006, and improved in 2011 by M.Orouskhani and Al. [19]. It was applied in combinatorial problem in 2014 [20] .The CSO adapted the natural behavior of cats composed by two modes: the seeking mode when the cat is resting and the tracing mode when the cat is hunting. These two modes are combined by mixture ratio (MR). The name, position, velocity and flag characterize each cat.

4. APPLY DISCRET CSO TO FSSP:

In this section, the definition of operators is operations used in CSO algorithm. Let us n jobs $J=\{J_1, J_2\ldots J_n\}$, m machines $M=\{M_1, M2\ldots, M_m\}$, and $S=\{s_1, s_2, \ldots, s_n\}$ a schedule presenting solution where for ***i*** between 0 and n, s_i is a job in J, and Card(S)= n.

Table1: Cat's Parameters

position	Is the solution/schedule presented by a vector of jobs *S*.
velocity	Sets of couples permutation *(Ji ,Jj)*, Let's v a velocity, $\boldsymbol{v=(i_k,j_k)[k:0 \rightarrow \vert v \vert]}$. |v| present the number of couples in v.
Flag	To know in each mode what are the cat is.

2.3 DEFINITION OF OPERATION:

To use CSO to discrete problem, some rules are recommended to be respected:

- Addition between two velocities $\mathbf{v_1}$, $\mathbf{v_2}$ is a new velocity v which contains permutation couples of $\mathbf{v_1}$ and $\mathbf{v_2}$.

Journal of Theoretical and Applied Information Technology
20th February 2015. Vol.72 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Addition between position **x** and velocity **v** is a new position **x'**, obtained by applying all permutation couples of **v** to **x**.
- Opposite of a velocity **v=(i_k,j_k)[k:0→|v|]** is **¬v=(i_k,j_k)[k : |v| →0]** ,and, (**v**) + (**¬v**) = **ø** .
- Subtraction between two position x1 and x2, is a velocity v. this operation is the opposite of addition:

x1- x2 = v ⇨ x2 + v = x1.

- Multiplication of a real r and a velocity **v= (i_k,j_k)[k :0 →|v|]**, is a new velocity. The possible cases according to the real **r**, are:

✓ If (r=0) then r*v= **ø**
✓ If (r ∈]0,1]) then **r * v = (i_k,j_k)[k : 0 → (c*|v|)]**
✓ If **(r>1)** then take the decimal part of **r**, after do the same two previous step.
✓ If **(r<0)** then **r * v= (-r)*(¬v)**, with **(-r)>0**. And do the previous steps.

The CSO algorithm is composed of seeking mode, and tracing mode combined by a mixture ratio. The processing of these two modes in CSO algorithm is as shown below:

1. Seeking mode:

It shows that the cat $_i$ is at rest, to observe the best place to move to. The parameters used in this mode are:

SMP: Seeking memory pool.
CDC: seeking range of the selected dimension.
SRD: counts of dimension to change
SPC: self-position consideration.

In a behavior of cats, SMP is the number of observations to consider before deciding the best position where to move. SPC gives to a cat the freedom whether to move or not. If the cat finds its current position as the best, then it will not change it, and stay on the same position. SRD and CDC are both necessary factors in updating the solution.

Seeking mode is as follows:

Step 1: put j copies of the present position of the cat k, with j = SMP. If the value of SPC is true or j = SMP-1, and retains the cat as one of the candidates.

Step 2: Generate a random value of SRD

Step 3: If the fitness (FS) are not equal, calculate the probability of each candidate by equation **(a)**, the default probability value of each candidate is 1.

Step 4: Perform mutation and replace the current position.

$$Pi = \frac{|FS_i - FS_{max}|}{FS_{max} - FS_{min}} \qquad \textbf{(a)}$$

2. Tracing mode:

This is the cat-hunting mode, where the cat traces its path, according to its own velocity to chase a prey or any moving object. The description of the process of each cat in this mode is as follows:

Step 1: update the velocities of each **cat** $_k$ according to equation **(b)**.

$$v'_k = w * v_k + r * c * (x_{best} - x_k) \qquad \textbf{(b)}$$

Where:
v'_k: The new velocity value
w : Inertia weight
x_{best}: is the best position in swarm.
v_k : the old velocity value (current value).
c: a constant.
r: a random value in the range **[0, 1]**.

Step 2: check if the velocities are of the highest order.

Step 3: update the position of k cat according to equation **(c)**.

$$x'_k = x_k + v_k \qquad \textbf{(c)}$$

Where:
x'_k: the new position values of the cat $_k$
x_k: the current position of cat $_k$
v_k: the velocity of cat $_k$

2.4 THE COMPLETE CSO ALGORITHM:

The full mode is composed of the SM and TM combined by a mixture ratio (**MR**), the flag is used to determinate the mode of each cat in swarm. The description of the process is:

Begin:
(1) Generate N cats
(2) Initialize flag, velocity, and position every cat.
(3) Initialize gbest with the lowest fitness cat in swarm.
(4) for each cat in swarm
    If the flag of the selected cat is TM
        Apply selected cat into TM process
    Else
        Apply selected cat into TM process
    EndIf
    Update gbest
End for
(5) Re-pick number of cats and set them into TM according to MR, and set other cats in SM.
If the condition is to terminate yes then complete the program
Else repeat **(4)** and **(5)**.
End.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. EXPERIMENTS AND COMPUTATIONNAL RESULTS

To prove the performance of discrete CSO to solve FSSP in this paper, this algorithm is applied to solve thirty-one chosen instances of FSSP in OR-LIBRARY [21]. The method is coded by C++ programming language, which runs on an Ultrabook characteristic's 2.1 GHz 2.59Ghz Intel Core i7 PC with 8G of RAM. Each instance runs for one hour in maximum. Table 2 shows the values of parameters used [20]:

Table 2: Used Parameters Values

SMP	5
CDC	0.8
MR	0.3
C	2.05
R	[0,1]
W	0.729

Table 3 shows the **instances** name, the job number **n** and the machine number **m**, best known solution (**BKS**) found by others algorithm [22-23], and the best solution obtained by applying CSO (**best**) to the selected instance in 10 times . The columns **T** in table 3, show average time execution in seconds to find the **BEST**, the percentage error (***Err* %**) value is obtained by

$$\boldsymbol{Err} = \frac{BEST - \text{BKS}}{\text{BKS}} \times 100$$

Table 3: Table Of Results

Instance	n * m	BKS	BEST	Err %	T (s)
Carlier					
Car1	11×5	7038	7038	**0.00**	01
Car2	13×4	7166	7166	**0.00**	01
Car3	12×5	7312	7312	**0.00**	01
Car4	14×4	8003	8003	**0.00**	01
Car5	10×6	7720	7720	**0.00**	01
Car6	8×9	8505	8505	**0.00**	01
Car7	7×7	6590	6590	**0.00**	01
Car8	8×8	8366	8366	**0.00**	01
Heller					
Hel1	100×10	516	516	**0.00**	27
Hel2	20×10	136	135	**-0.74**	06
Reeves					
ReC01	20×5	1247	1247	**0.00**	02
ReC03	20×5	1109	1109	**0.00**	03
ReC05	20×5	1242	1245	**0.24**	01
ReC07	20×10	1566	1566	**0.00**	03
ReC09	20×10	1537	1537	**0.00**	02
ReC11	20×10	1431	1431	**0.00**	01
ReC13	20×15	1930	1930	**0.00**	178
ReC15	20×15	1950	1950	**0.00**	101
ReC17	20×15	1902	1902	**0.00**	116
ReC19	30×10	2093	2099	**0.29**	18
ReC21	30×10	2017	2020	**0.15**	486
ReC23	30×10	2011	2020	**0.45**	22
ReC25	30×15	2513	2525	**0.48**	377
ReC27	30×15	2373	2396	**0.97**	61
ReC29	30×15	2287	2305	**0.79**	332
ReC31	50×10	3045	3058	**0.43**	900
ReC33	50×10	3114	3114	**0.00**	163
ReC35	50×10	3277	3277	**0.00**	07
ReC37	75×20	4951	5096	**2.93**	1583
ReC39	75×20	5087	5161	**1.45**	568
ReC41	75×20	4960	5087	**2.56**	2071

The table of result demonstrate that CSO can solve numerous instance in OR-library, and it also shows another best solution to instance "Hel2", minus one than the best known solution [23]. The error percent of the other executed instances is between 0.00 and 2.93.

6. CONCLUSION:

This research paper presents a new adaptation of CSO algorithm to solve the Flow Shop scheduling problem. The obtained results to some benchmark problem instances (of Carlier, Heller and Reeves) prove the performance of CSO algorithm to solve the problem up to 50 jobs, without any error. For up to 50 jobs, it approaches solution with a negligible percentage error. This proves the ability of the CSO algorithm to solve the FSSP. The future work is to extend the application of CSO algorithm for others kinds of scheduling problems, and multi-objective scheduling problem.

REFERENCES

[1] Gupta JND, Stafford Jr. EF., Flowshop scheduling research after five decades, European Journal of Operational Research, 169, pp. 699–711, 2006

[2] Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. Omega, 17(6), 551-557.

[3] Ishibuchi, H., Misaki, S., & Tanaka, H. (1995). Modified simulated annealing algorithms for the flow shop sequencing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

problem. *European Journal of Operational Research*, *81*(2), 388-398.

[4] Grabowski, J., & Wodecki, M. (2004). A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research*, *31*(11), 1891-1909.

[5] Ben-Daya, M., & Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, *109*(1), 88-95.

[6] Gao, K. Z., Pan, Q. K., & Li, J. Q. (2011). Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. *The International Journal of Advanced Manufacturing Technology*, *56*(5-8), 683-692.

[7] Pan, Q. K., Suganthan, P. N., Liang, J. J., & Tasgetiren, M. F. (2011). A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. *Expert Systems with Applications*,*38*(4), 3252-3259.

[8] Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, *30*(4), 1061-1071.

[9] Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, *22*(1), 5-13.

[10] Stützle, T. (1998, September). An ant approach to the flow shop problem. In*Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)* (Vol. 3, pp. 1560-1564).

[11] Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, *155*(2), 426-438.

[12] Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Chen, A. H. (2011). A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences*, *181*(16), 3459-3475.

[13] Jarboui, B., Ibrahim, S., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimisation for solving permutation flowshop problems. *Computers & Industrial Engineering*, *54*(3), 526-538.

[14] Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, *177*(3), 1930-1947.

[15] Tasgetiren, M. F., Sevkli, M., Liang, Y. C., & Gencyilmaz, G. (2004). Particle swarm optimization algorithm for permutation flowshop sequencing problem. In*Ant Colony Optimization and Swarm Intelligence* (pp. 382-389). Springer Berlin Heidelberg.

[16] Sotskov, Yu N., and N. V. Shakhlevich. "NP-hardness of shop-scheduling problems with three jobs." Discrete Applied Mathematics 59.3 (1995): 237-266.

[17] Johnson, S. M. (1954):Optimal two- and three-stage production schedules with setup times included,‖ Naval Research Logistics Quarterly, vol. 8, pp. 1-61,

[18] Chu, S. C., Tsai, P. W., & Pan, J. S. (2006). Cat swarm optimization. In PRICAI 2006: Trends in Artificial Intelligence (pp. 854-858). Springer Berlin Heidelberg.

[19] Orouskhani, M., Mansouri, M., & Teshnehlab, M. (2011). Average-inertia weighted cat swarm optimization. In *Advances in Swarm Intelligence* (pp. 321-328). Springer Berlin Heidelberg.

[20] Mohammed ESSAID RIFFI and Abdelhamid BOUZIDI (2014),"Discrete Cat Swarm Optimization for Solving the Quadratic Assignment Problem", International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 4, No. 6, pp. 85-92.

[21] Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, 1069-1072.

[22] Qian, B., Wang, L., Hu, R., Wang, W. L., Huang, D. X., & Wang, X. (2008). A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *38*(7-8), 757-777.

[23] Mircea ANCĂU (2012). ON SOLVING FLOWSHOP SCHEDULING PROBLEMS, *PROCEEDINGS OF THE ROMANIAN ACADEMY*, Series A, Volume 13, Number 1/2012, pp. 71–79