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 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 n job J = {J₁, J₂ ... Jₙ}, and m machine M = {M₁, M₂ ... Mₘ}, each job is composed of m distinct operations O = {O₁, O₂ ... Oₘ}. The operation in each job should respect the sequence of machine. Every operation is represented by a pair mₓk and tₓk (k∈ [1, (n*m)]), where mₓk represents the machine on which the process oₓ will be
The matrix INFO in fig. 1 has 4 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 $o_i$.

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 $J$, $J_i = $ \{(m_{i1}, t_{i1}), (m_{i2}, t_{i2}), (m_{i3}, t_{i3})\}$ for $k \in [1,3]$.

A random solution is follow:

$$J_1 = \{(1, 6), (2, 1), (3, 4)\}$$
$$J_2 = \{(1, 3, (2, 6)), (3, 2)\}$$
$$J_3 = \{(1, 1), (2, 2), (3, 1)\}$$
$$J_4 = \{(1, 2), (2, 1), (3, 5)\}$$

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. 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 $M_i$ $(1 \leq i \leq 3)$ represents the machines, and each color represents the jobs.
Step 1: update the velocities of each cat \( k \) according to equation (b).
\[ v'_k = w \cdot v_k + r \cdot c \cdot (x_{\text{best}} - x_k) \]  
(b)
Where:
- \( v'_k \): The new velocity value
- \( w \): Inertia weight
- \( x_{\text{best}} \): 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).
\[ x'_k = x_k + v'_k \]  
(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 determine 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
   - Update gbest
   - EndIf
   - 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.

#### End.
5. EXPERIMENTS AND COMPUTATIONAL 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>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td>5</td>
</tr>
<tr>
<td>CDC</td>
<td>0.8</td>
</tr>
<tr>
<td>MR</td>
<td>0.3</td>
</tr>
<tr>
<td>C</td>
<td>2.05</td>
</tr>
</tbody>
</table>

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


