APPLICATION OF DRAGONFLY ALGORITHM FOR MULTI OBJECTIVE SCHEDULING PROBLEMS IN FMS

B. SATISH KUMAR¹, G. JANARDHANA RAJU² & G. RANGA JANARDHANA³

¹Associate Professor, Department of Mechanical Engineering, Vignana Bharathi Institute of Technology, Hyderabad, Telangana, India
²Professor, Department of Mechanical Engineering, Nalla Narasimha Reddy Education Society’s Group of Institutions, Hyderabad, Telangana, India
³Professor, Department of Mechanical Engineering, JNTUA College of Engineering (Autonomous) Ananthapuramu, Andhra Pradesh, India

ABSTRACT
Scheduling plays an important role in Flexible Manufacturing System (FMS). Various Evolutionary Algorithms are used by different researchers to solve the multi objective scheduling problems. However, solutions obtained by some of these algorithms suffer from various issues such as struck in local optima, not support deadline constraint, poor convergence speed etc., of the FMS. In this paper, a new optimization technique called Dragonfly Algorithm (DA) is implemented for optimum scheduling of Multi objective Scheduling problems considered from the literature. The main inspiration of the DA algorithm originates from the static and dynamic swarming behaviors of dragonfly in nature. The Combined Objective Function (COF) is formulated by considering two objectives such as Minimization of machine idle time and Penalty cost minimization with equal weight-ages. MATLAB code has developed to determine the COF values and also for implementing Dragonfly Algorithm to obtain the optimum solution. It is observed that, the results obtained by DA algorithm are very competitive when compared with other well-known algorithms like Genetic Algorithm, Particle Swarm Optimization, Cuckoo Search, Modified Cuckoo Search & Jaya Algorithms.

KEYWORDS - Flexible Manufacturing System, Multi Objective Scheduling, Combined Objective Function & Dragonfly Algorithm

1. INTRODUCTION
In the Present global competitive market, fulfilling the customer requirements and delivering the goods as per the schedule is very important and also tough challenge for manufacturing companies. Scheduling plays vital role in manufacturing the quality products with the effective usage of limited resources to meet the customer requirements in dynamic market conditions. Flexible Manufacturing System is suitable for addressing the unpredictable changes in market environments. The typical view of FMS is shown in Fig.1 which consists of 4 automatic control machines controlled by common control centre and has one loading, unloading, inspection and material handling systems.
In the past, many optimization algorithms are used for solving the Multi Objective Scheduling Problem (MOSP). Sankar et al.[14] applied GA for solving MOSP to find optimum Combined Objective Function value and observed good results than SPT, LPT methods. J. Jerald et al.[13] implemented PSOA for solving the MOSP and obtained better results compared to other Algorithms. Deb S and Yang[12] developed Cuckoo Search (CS) Algorithm, also applied to solve engineering design optimization problems and noticed better solutions than the Particle Swarm Optimization Algorithm. Anna Lawrynowicz[11] conducted the survey on recent developments in building GA and also proposed Modified Genetic Algorithm. Shashikant et al.[8] applied the CS Algorithm for solving Multi Objective Scheduling Optimization Problem and obtained favorable results than LPT, SPT, GA & PSO. Genetic Algorithm is used in scheduling FMS by A.V.S. Sreedhar Kumar et.al.[9] and compared results with conventional scheduling rules. Nidhiry.N.M et.al[7] made an attempt to solve the MOSP by NSGA-II, results are compared with GA and PSO. Seyed Mirjali[6] used Dragonfly Algorithm(DA), Binary Dragonfly Algorithm(BDA) and Multi Objective Dragonfly Algorithm(MODA) for solving benchmark and case study problems and concluded that results are improved than GA and PSO. R.Venkat Rao[5] developed and applied Jaya Algorithm successfully on 24 constrained benchmark functions and revealed that Jaya Algorithm results are more effective. Naveen Sihag [3] applied a Novel Adaptive Dragonfly Algorithm for solving global optimization problems and recorded better results than standard DA. Deepanshi Nanda, Amit Chabra[4] proposed Deadline Aware Multi-Objective Dragonfly Optimization Technique. Make span and flow time are used to design multi-objective fitness function. Extensive experiments are done by tuning the various parameters of proposed technique. Experimental analysis reveals that the proposed technique performs better than existing job scheduling strategies. Cigdem inan and Hakan Gulcan[1] proposed modified DA for solving single and multi objective problems, obtained better results than DA. Satish Kumar et.al [2] used Jaya and Modified Cuckoo Search Algorithms and obtained better results than GA, PSO and CS algorithms. In recent years, the Dragonfly Algorithm developed and found to be effective for finding the optimum solutions. Population based DA is a meta-heuristic optimization algorithm has an ability to avoid local optima and get global optimal solution, due to this it is more suitable for practical applications without structural modifications in algorithm for solving different constrained or unconstrained optimization problems. DA integrated with adaptive technique reduces the computational times for highly complex problems. The remaining part of this paper is organized as follows, section 2 describes the Dragonfly algorithm and its algebraic equations, section 3 includes problem definition and assumptions, section 4 consists of MATLAB programming, section 5 discusses Results and Discussions, section 6 Conclusions.
2. DRAGONFLY ALGORITHM

The Dragonfly Algorithm (DA) was developed by Seyedali Mirjalili[6] and is one of the optimization techniques developed in recent years. Fig. 2 shows the Flow chart of Dragonfly Algorithm.

DA is a multi-objective optimization technique inspired from ordinary behavior of dragonflies, ideally dependent on exploration and exploitation. The dragonfly creates sub-warms around various areas for navigating, food searching and survival from enemies which is useful for convergence towards pareto-optimal solutions and coverage of optimal solution along the objectives. As swarms of living creature follow living instincts, so dragonfly individuals must have attracted toward efficient utilization of resources.

2.1 Implementation steps of Dragonfly Algorithm

- Initialization – Randomly generate a position($X$) matrix and step($\Delta x$) matrix between lb and ub
- Set $\text{iter} = 1$
- while $\text{iter} \leq \text{iter}_{\text{max}}$ do
  - Set $r = \Delta x_{\text{max}} + ((\text{ub} - \text{lb}) \times \frac{\text{iter}}{\text{iter}_{\text{max}}} \times 2)$
  - Set $w = 0.9 - \text{iter} \times \frac{(0.9 - 0.4)}{\text{iter}_{\text{max}}}$
  - Set $f = 2 \times \text{rand}$
if $\frac{iter}{iter \ max} < \frac{1}{2}$ then

- set $s = 2 \times \text{rand} \times (0.1 - \frac{iter}{iter \ max})$
- set $a = 2 \times \text{rand} \times (0.1 - \frac{iter}{iter \ max})$
- set $c = 2 \times \text{rand} \times (0.1 - \frac{iter}{iter \ max})$
- set $e = (0.1 - \frac{iter}{iter \ max})$

else

- set $s = a = e = 0$

end if

Evaluation

- Update food fitness, food position ($X^+$), enemy fitness and enemy position ($X^-$) among all individuals in $X$

- Identify number of neighboring individuals ($N_i$) for each individual $i$. If distance between $X_i$ and $X_j$ is less than '$r$' then individuals $i$ and $j$ are neighbors

- Compute separation ($S_i$), alignment ($A_i$) and cohesion ($C_i$) for each individual $i$ using Eqs. (1), (2), and (3) if neighbors exist else set all these values to zero

$$S_i = \sum_{j=1}^{N_i} (X_i - X_j) \quad (1)$$

$$A_i = \frac{\sum_{j=1}^{N_i} \Delta X_j}{N_i} \quad (2)$$

$$C_i = \frac{\sum_{j=1}^{N_i} X_j}{N_i} - X_i \quad (3)$$

- Compute distance ($D_{fi}$) between $X_i$ and $X^+$ and distance ($D_{ei}$) between $X_i$ and $X^-$ for each individual 'i'

- Print optimal solution and corresponding objective values. This completes one iteration, the process will be repeated for obtaining the optimum solution.

3. PROBLEM DEFINITION
Two research problems are considered from the literature i.e. Scheduling of 43 parts manufacturing on 16 machines [8] and Scheduling of 80 parts on 16 machines [7] by considering the multi objective function.

3.1. Objective Function

In the present work considered two objectives, i.e., Machine idle time Minimization and Penalty cost minimization equal weight-age is given for both the objectives and formulated Combined Objective Function (COF) shown in equation 4

\[ COF = W_1 \times \left( \frac{PC}{MP} \right) + W_2 \times \left( \frac{X_q}{TE} \right) \] (4)

\( W_1 \) = Weight-age factor for minimizing the total penalty cost
\( W_2 \) = Weight-age factor for minimizing the total elapsed time
\( PC \) = Penalty Cost
\( MP \) = Maximum Penalty
\( X_q \) = Total machines idle time
\( TE \) = Total Elapsed time

3.2. Assumptions

- There is only one machine of each type.
- Preemptive not allowed.
- Machines break downs are neglected.
- Each Job is to be processed only once on any machine as per process requirement.
- Job will move to next machine only after completing the process on current machine.
- In-process inventory is allowed.
- If any machine is busy, job will wait till machine is become available for processing.
- Processing sequence, processing time, batch size, deadline, and penalty cost for not meeting the deadline of all the parts are known.
- Assumed that processing time is inclusive of travelling time.

4. PROGRAMMING IN MATLAB

MATLAB program is written for calculating the Combined Objective Function (COF) value for the particular part sequence by using the equation 4. The two research problems considered are finding the optimum sequence of 43 parts and 80 parts processing on 16 machines. The total 43 factorial sequences and 80 factorial sequences are possible for 43 parts and 80 parts problem respectively. The part sequence which gives least COF value is optimum solution which is computed by MATLAB code by implementing the Dragonfly Algorithm as per the procedure explained in the section 2.
5. RESULTS & DISCUSSIONS

Initially DA implemented for finding the optimum sequence and COF value for 43 parts problem. MATLAB program run for 10, 20, 30, 40, 50 & 100 iterations. After 40 iterations there is no change in COF value, therefore process is stopped at 100 iterations. Results obtained by DA are compared with the other approaches like LPT, SPT, PSO, GA, CS, MCS, Jaya algorithms and results are tabulated in Table 1 and comparisons of different approaches are shown in Fig: 3.

<table>
<thead>
<tr>
<th>Algorithm /Method</th>
<th>Sequence</th>
<th>COF Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT[14]</td>
<td>20, 23, 38, 1, 9, 26, 22, 10, 34, 18, 36, 11, 25, 5, 16, 2, 40, 44, 31, 7, 24, 28, 17, 6, 29, 35, 37, 15, 39, 42, 12, 27, 33, 3, 43, 19, 13, 32, 30, 8, 14, 21</td>
<td>0.3240</td>
</tr>
<tr>
<td>LPT[14]</td>
<td>21, 14, 8, 30, 32, 12, 13, 19, 43, 3, 33, 27, 42, 39, 15, 37, 35, 29, 6, 17, 28, 24, 7, 31, 41, 4, 40, 2, 16, 5, 25, 11, 36, 18, 34, 10, 22, 26, 9, 1, 38, 23, 20</td>
<td>0.4850</td>
</tr>
<tr>
<td>PSO[13]</td>
<td>27, 30, 28, 10, 18, 15, 34, 42, 5, 33, 8, 37, 23, 25, 9, 2, 35, 43, 20, 6, 4, 36, 19, 17, 24, 39, 31, 12, 28, 32, 26, 16, 14, 22, 3, 1, 1, 11, 41, 29, 40, 21, 13, 7</td>
<td>0.2983</td>
</tr>
<tr>
<td>GA[14]</td>
<td>21, 28, 16, 42, 39, 25, 5, 23, 22, 35, 36, 37, 15, 1, 2, 8, 4, 3, 40, 20, 31, 11, 14, 30, 27, 43, 17, 10, 7, 26, 33, 18, 29, 8, 41, 32, 13, 12, 24, 6, 19, 38, 34</td>
<td>0.2740</td>
</tr>
<tr>
<td>CS[8]</td>
<td>8, 14, 28, 31, 3, 42, 26, 33, 22, 20, 5, 24, 12, 41, 18, 7, 10, 19, 23, 38, 4, 3, 40, 37, 21, 15, 17, 39, 6, 2, 34, 1, 29, 27, 16, 36, 30, 25, 32, 13, 43, 11, 9</td>
<td>0.2646</td>
</tr>
<tr>
<td>MCS[2]</td>
<td>19, 21, 28, 31, 32, 24, 30, 3, 39, 11, 14, 15, 41, 12, 13, 17, 16, 33, 35, 8, 43, 4, 36, 18, 30, 43, 22, 37, 42, 6, 29, 7, 27, 25, 10, 20, 2, 23, 1, 5, 38, 9, 26</td>
<td>0.1419</td>
</tr>
<tr>
<td>JA[2]</td>
<td>10, 3, 8, 11, 32, 28, 43, 36, 12, 19, 22, 2, 1, 5, 4, 21, 6, 33, 29, 26, 37, 7, 9, 13, 31, 39, 15, 25, 23, 34, 16, 18, 24, 30, 35, 38, 40, 17, 14, 41, 20, 42, 27</td>
<td>0.1311</td>
</tr>
<tr>
<td>DA</td>
<td>31, 1, 28, 16, 22, 5, 12, 32, 20, 7, 6, 9, 19, 8, 15, 2, 3, 4, 40, 10, 11, 33, 14, 18, 27, 17, 13, 21, 23, 24, 36, 25, 29, 35, 30, 37, 34, 41, 38, 43, 26, 39, 42</td>
<td>0.1279</td>
</tr>
</tbody>
</table>

Figure 3: Comparison of COF value obtained with LPT, SPT, GA, PSO, CS, MCS, JA & DA for 43 parts problem

It is noted from the Table:1 & Fig:3 that COF values obtained for the 43 parts problem by DA is 0.1279 which is optimal (i.e minimum) among the values of 0.3240, 0.4850, 0.2740, 0.2983, 0.2646, 0.1419, 0.1311 & 0.1279 for SPT, LPT, PSO, GA, CS, MCSA, JA & DA respectively.

For checking the performance of DA implemented on 80 parts problem for finding the optimum sequence & COF value and compared with the results obtained by the other researchers.
Table 2: COF values by LPT, SPT, MCS, JA & DA Algorithms for 80 parts problem

<table>
<thead>
<tr>
<th>Algorithm /Method</th>
<th>Sequence</th>
<th>COF Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT[2]</td>
<td>79,75,20,80,23,38,1,9,53,26,52,22,10,61,34,18,51,57,70,36,11,73,69,25,5,99,76,47,78,66,16,2,49,50,56,48,40,60,55,58,4,46,54,41,31,7,63,24,28,17,64,29,6,48,35,77,37,15,39,42,27,33,62,65,71,3,43,19,12,13,67,30,32,8,14,72,21</td>
<td>0.1658</td>
</tr>
<tr>
<td>LPT[2]</td>
<td>21,72,14,8,32,30,67,13,12,19,43,3,71,65,62,33,27,45,42,39,15,37,77,35,68,44,6,29,64,17,28,24,63,7,31,41,54,46,4,58,55,50,48,60,48,56,50,49,2,16,66,78,47,74,76,59,5,25,69,73,11,36,70,57,51,18,34,61,10,22,52,26,53,9,13,8,23,80,20,75,79</td>
<td>0.1486</td>
</tr>
<tr>
<td>MCS[2]</td>
<td>9,26,11,38,57,80,5,19,59,58,1,49,20,23,31,2,70,75,76,58,44,39,61,69,29,79,10,47,71,48,24,56,74,78,46,25,60,40,27,66,73,54,68,51,53,6,77,32,63,7,28,52,34,42,35,55,65,45,30,64,37,22,43,36,18,72,62,5,21,8,17,15,67,33,12,16,13,41,14</td>
<td>0.1321</td>
</tr>
<tr>
<td>JA[2]</td>
<td>80,4,14,59,15,9,3,17,47,12,30,2,28,37,8,68,10,52,27,33,11,12,32,35,63,16,29,40,22,23,24,25,26,31,21,70,39,58,20,42,44,50,43,45,65,46,34,48,72,49,51,53,55,36,6,56,5,60,62,19,38,54,64,66,67,69,71,76,73,57,41,78,18,7,47,75,77,61,1,79</td>
<td>0.1083</td>
</tr>
<tr>
<td>DA</td>
<td>77,7,80,9,34,49,60,50,15,27,26,38,5,62,53,30,57,35,24,44,69,4,12,65,13,45,10,43,42,14,19,40,16,18,20,55,22,23,72,25,17,28,8,29,32,66,33,31,54,36,41,3,37,1,2,39,59,76,63,46,68,6,48,47,51,21,52,58,56,70,61,64,67,11,71,73,78,74,75,79</td>
<td>0.1129</td>
</tr>
</tbody>
</table>

Figure 4: Comparison of COF value obtained with LPT, SPT, MCS, JA & DA for 80 parts problem

It is noted from the Table. 2 & Fig.4 that COF values obtained for the 80 parts problem by DA is 0.1079 which is optimal (i.e. minimum) among the values of 0.1486, 0.1658, 0.1321, 0.1083 & 0.1129 by LPT, SPT, MCS, JA & DA respectively.

6. CONCLUSIONS

The Dragonfly algorithm is implemented for solving Multi Objective Scheduling Problem of 43 parts processing on 16 machines. Previous researchers are solved the same problem by applying SPT, LPT, PSO, GA, CS, MCS & Jaya algorithms. In the present work, using Dragonfly algorithm the best possible sequence and COF values are obtained. It is observed from the results that DA Algorithm has recorded better results when compared to SPT, LPT, PSO, GA, CS, MCS and Jaya Algorithms.

Another problem of processing 80 parts on 16 machines is taken up and obtained best sequence by applying Dragonfly Algorithm. The COF value is compared with SPT, LPT, MCS and Jaya algorithms. It is concluded from the results that Dragonfly Algorithm has shown better results than other algorithms for this Multi objective Scheduling Problem.
From this work, it is concluded that Dragonfly Algorithm is one of the better options for solving the Multiobjective scheduling problems in Flexible Manufacturing Systems.

ACKNOWLEDGEMENT

The author gratefully acknowledges the support of his Ph.D. guides (co-authors) of this research article and also Management, Principal and HoD of Mechanical Engineering department of Vignana Bharathi Institute of Technology, Hyderabad for their encouragement and support.

REFERENCES


