# Modified Firefly Algorithm using Smallest Position Value for Job-Shop Schedulling Problems

Muhaza Liebenlito[1], Nur Inayah[1], Aisyah Nur Rahmah[1] and Ario Widiatmoko[2]

[1]*Departement of Mathematics, UIN Syarif Hidayatullah Jakarta, Jl. Ir. H. Juanda No. 95, Tangerang Selatan, Indonesia*

[2]*Department of Informatics, University of Sriwijaya, Jl. Srijaya Negara, Ilir Barat I, Kota Palembang, Indonesia*

Keywords: Job-Shop Scheduling Problem, Modified Firefly Algorithm, Smallest Position Value, Minimizing Makespan.

Abstract: In this paper, we will modify the firefly algorithm to find the minimum makespan of job-shop scheduling problem. Firefly algorithm generally is used to solve continuous optimization problem which is have to modify by adding smallest position value to fit the discrete optimization problems, named Modified Firefly Algorithm–Smallest Position Value (MFASPV). The result from MFASPV is compared with Bi-directional algorithm, Tabu Search, and Discrete Firefly Algorithm. The MFASPV obtain minimum makespan as good as Tabu Search and outperform the Discrete Firefly Algorithm and Bi-directional Algorithm.

## 1 INTRODUCTION

One of the scheduling problems often encountered by the manufacturing industry is the job-shop scheduling or Job-Shop Scheduling Problem (JSSP). JSSP is sorting out the creation or work of the job as a whole with the order of the machine through each different job. JSSP is classified into the combinatorial or discrete optimization problem. The computation complexity for JSSP has been categorized into Nondeterministic Polynomial-hard problem (NP-hard) if the $m \geq 3$, where $m$ is the number of machine(Garey et al., 1976).Because of its complexity, many research has been developed to solve this problem. In the paper (Dell'Amico & Trubian, 1993) use Tabu Search (TS) and Bi-directional (Bidir) algorithms to solve JSSP by minimizing the makespan.

In the 2009, Xin-She Yang developed a bio-inspired algorithm called Firefly Algorithm (FA) to handle continuous optimization problem (Yang, 2009). In that paper shown FA outperform the Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE). Furthermore, the FA can be applied in various continuous nonlinear optimization problem in any Engineering problems (Yang & He, 2013).

In the 2009, Tasgetiren et al. solved the flow-shop scheduling problem using PSO combined with Smallest Position Value (SPV) rule. The SPV rule used to convert continuous variables on PSO mechanism into discrete variables (Tasgetiren et al., 2009). Recently, the paper of K.C. Udaiyakumar and M. Chandrasekaran proved that Discrete Firefly Algorithm (DFA) which they proposed can be used to solve JSSP by minimizing makespan (Udaiyakumar & Chandrasekaran, 2014). However, four of the twenty-five of Lawrence problems that tested have not met the optimum value.

Based on the explanation above, we tried to modify the FA with SPV rule to solve the JSSP. The results will be compared with previous results TS and Bidir (Dell'Amico & Trubian, 1993) and DFA (Udaiyakumar & Chandrasekaran, 2014) which solved the same problem that provided by Taillard benchmark (Taillard, 1993).

## 2 PROBLEM DESCRIPTION

The JSSP can be defined as follows, given a sequence $\{J_j\}_{j=1}^{n}$ jobs and $\{M_i\}_{i=1}^{m}$ machines. Job $j$ consist of sequence of $k$ operations $O_{j1}, O_{j2}, \ldots, O_{jk}$ which must be processed in this order, i.e. we have precedence constraints of the form $O_{jk} \rightarrow O_{j,k+1}$,

Liebenlito, M., Inayah, N., Rahmah, A. and Widiatmoko, A.
Modified Firefly Algorithm using Smallest Position Value for Job-Shop Schedulling Problems.
DOI: 10.5220/0008516600230027
In *Proceedings of the International Conference on Mathematics and Islam (ICMIs 2018)*, pages 23-27
ISBN: 978-989-758-407-7

where $k = 1, 2, \ldots, K-1$(Brucker, 2007). The operation $O_{jk}$ of job has to be performed in the predefined order by specified machines $M_i$ within $t_{ij}$ processing time. Each machine can process only one job at the same time and each job can be processed by only one machine at the same time. The problem is to find a feasible schedule which minimizes the makespan which is some objective function depending on the finishing times $C_j$ of the last operations $O_{jk}$ of the jobs called. The makespan denoted

$$C_{max} = \max(C_j), j = 1,2,\ldots,n \qquad (1)$$

$$C_j = \sum_{i=1}^{m} t_{ij} + w_i$$

where $w_i$ is waiting time.

The problems can be represented as disjunctive graph (Bażewicz et al., 2000; Kuhpfahl, 2016). Let $G = (V(G), A(G), E(G))$ is disjunctive graph with the operation $k$-th of job $j$ denoted by node $(k/j) \in V(G)$ and node 0 as starting node and node 1 as ending node. Arc of node $(k/j)$ to node $((k+1)/j)$ is element of $A(G)$. For all job $j$, there is one arc that connects node 0 to $(1/j)$ and one arc connects node $(K/j)$ to node 1, where node $(K/j)$ represents as the last operation $K$ of job $j$. Each arc connects from node $(k/j)$ to node $((k+1)/j)$ and has a weight that represents as processing time of operation $k$-th of job $j$. The weight of arc which connects from node 0 to node $(1/j)$ is 0 and weight of arc from node $(K/j)$ to node 1 is processing time of operation $K$ of job $j$. $E(G)$ denotes a set of edges that connects node $(k/j) \in V(G)$ of all different jobs which processed on the same machines.

For more details see the following example, given two jobs $\{J_1, J_2\}$ with $k$ order of operations and the jobs processed in three machines $\{M_1, M_2, M_3\}$ within $t_{ij}$ processing time. The operations of each jobs can be represent as matrix,

$$O_{jk} = \begin{bmatrix} M_1 & M_2 & M_3 \\ M_2 & M_3 & M_1 \end{bmatrix}.$$

and its processing time,

$$t_{ij} = \begin{bmatrix} 4 & 25 \\ 2 & 38 \\ 20 & 14 \end{bmatrix}.$$

In Figure 1, there are three edges $\{e_1, e_2, e_3\}$which is the element of $E(G)$. Its edges have dash line which represent as the possible solution of the problem. The edge $e_1$ connects the vertices $(1/1)$ and $(3/2)$, it because $O_{11}$ and $O_{23}$ processed in the same machine $M_1$ as well as $e_2$ and $e_3$. Then the total weight of its longest path represent as makespan.

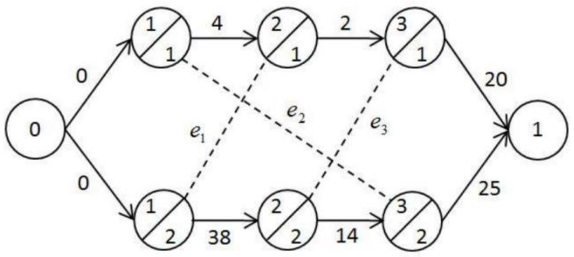

Figure 1: Representation of disjunctive graph for the example problem.

## 3 FIREFLY ALGORITHM

Firefly Algorithm (FA) is meta-heuristic algorithm inspired by flashing of the fireflies. FA first introduced by Xin-She Yang to solve Multimodal Optimization (Yang, 2009). There are three important things on the FA, that is:

- Light intensity proportional to the objective function $I(\mathbf{x}) \propto f(\mathbf{x})$.
- The attractiveness function of fireflies denoted

$$\beta(r) = \beta_0 e^{-\gamma r_{vw}^2} \qquad (2)$$

where $r_{vw} = \|\mathbf{x}_v - \mathbf{x}_w\|_2; v, w = 1,2,\ldots,N$ represents the distance between any two fireflies $\mathbf{x}_v$ and $\mathbf{x}_w$; and the number of fireflies denoted $N$. The Parameters $\beta_0 = 1$ and $\gamma \in [0, \infty)$.

- The movement of fireflies denoted

$$x_v^{(t+1)} = x_v^{(t)} + \beta(r)\left(x_w^{(t)} - x_v^{(t)}\right) + \alpha \otimes \left(Z \sim U(0,1) - \frac{1}{2}\right) \qquad (3)$$

where $\alpha$ is a vector represents controlling step-size parameter and it has a value $\alpha = (0,1)$; $Z$ is random variables whose uniformly distribution $(0,1)$; the operator $\otimes$ is vector-scalar multiplication.

The pseudo-code FA formulate as Algorithm 1.

Algorithm 1: Pseudo-code continuous FA.

```
Input:    Objective Function $f(\mathbf{x})$; number of
          population $N$; FA parameters: $\alpha, \beta_0$, and $\gamma$;
          and maximum iteration MAXITER.
Output:   Best $I_N$.
1.        Generate the initial population $\mathbf{x}_v$, $v = 1,2,..,N$
2.        Evaluate $I_v = f(\mathbf{x}_v)$
3.        while ($t <$ MAXITER) do
4.            for $v = 1$ to $N$ do
5.                for $w = 1$ to $N$ do
6.                    if $I_v > I_w$ then
7.                        Calculate the attractiveness
                          with distance $r$ using (1).
8.                        Move firefly $v$ to $w$ using
                          (2).
9.                    end if
10.                   Evaluate the objective function
11.               end for
12.           end for
13.           Rank the fireflies and find the current
              best.
14.       end while
```

In Algorithm 1 starts with input FA parameters, maximum iteration MAXITER, number of firefly population $N$, and define the objective function $f$. If we will find maximum of the objective function then set $I_v < I_w$ in step 6. Otherwise, if we find minimum of objective function then set $I_v > I_w$.

## 4 OUR PROPOSED ALGORITHM

The JSSP belongs to the combinatorial optimization problem where the decision variable is positive integer. Therefore, the Algorithm 1 should be adapted in order to solve the JSSP problem. To change continuous decision variables into discrete variables in Algorithm 1, we use SPV which firstly introduced by (Tasgetiren et al., 2004). The pseudo-code SPV has shown as follows:

Algorithm 2: Pseudo-code SPV.

```
Input:    $Y, r, s$.
Output:   $X$.
1.        $m = r \times s$
2.        $Y' = \{y_i \in Y | \, y_i' = sort(y_i)\}$
3.        $X' = index\_sort(Y)$
4.        $X = \{x_i' \in X' | x_i = (x_i' \, mod \, s) + 1\}$
```

On Algorithm 2, input $Y$ is a set of random numbers which have $m$ elements. The input $r$ is the number of machine and $s$ is the number of job. In Step 2 Algorithm 2, sort the elements of $Y$ ascendingly. Then in Step 3, $X'$ contains the index of elements of $Y$ that has been sorted.

On problem example in section II we have $s = 2$ and $r = 3$, so based on Algorithm 2 in Step 2 we get $m = 3 \times 2 = 6$. Let we generated the values of

$$Y = \{0.9755, 0.4326, 0.0397, 0.1821, 0.7702, 0.6918\}.$$

Based on Algorithm 2 in Step 2, obtain the sorted elements of $Y$ that stored in

$$Y' = \{0.0397, 0.1821, 0.4326, 0.6918, 0.7702, 0.9755\}.$$

In Step 3, $X'$ is set that contains the indices of sorted elements of $Y$, i.e.

$$X' = \{3, 4, 6, 5, 2, 1\}.$$

On Algorithm 2 in Step 4, each of elements of $X'$ the modular operation and depend on the number of job $s = 2$, we get

$$X = \{2, 1, 1, 1, 2, 2\}.$$

Set $X$ is feasible solution of the problem example in section II which represents the order of operations

$$O_{21} \rightarrow O_{11} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{22} \rightarrow O_{23}.$$

In Figure 2, node $(1/1)$ is predecessor from $(3/2)$, because the operation $O_{11}$ is executed before the operation $O_{23}$. The successor from $(1/2)$ is$(2/1)$, because $O_{12}$ is executed after $O_{21}$ finished. Thus, longest path of its disjunctive graph is

$$0 \rightarrow (1/2) \rightarrow (2/1) \rightarrow (3/1) \rightarrow (2/2) \rightarrow (2/3) \rightarrow 1$$

whose total weight is $0 + 38 + 2 + 20 + 14 + 25 = 99$.

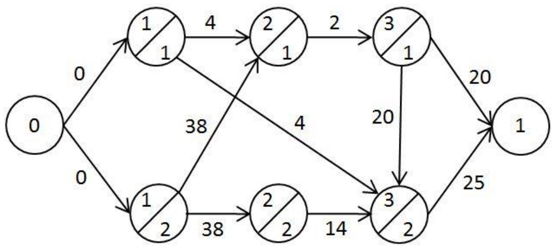

Figure2: Disjunctive graph for the feasible solution in Problem Example in Section II.

## 5 EXPERIMENTAL RESULTS

The result from our proposed algorithm called MFASPV. Compared to the previous results from Bidir and TS (Dell'Amico & Trubian, 1993), DFA (Udaiyakumar & Chandrasekaran, 2014) on the same benchmark Lawrence data provided by OR-Library (Beasley, 1990). The summary of benchmark data can be seen in Table 1. All experiments were performed on Notebook Intel Celeron N2840 @2.16 GHz with 4 GB RAM, and the code was compiled using Microsoft Visual C++. We use the number population $N = 50$ and maximum iteration $\mathtt{MAXITER} = 100$ for all benchmark data. The FA Parameters which used in this experiment i.e. $\alpha = 0.5$, $\gamma = 0.1$, $\beta_0 = 1$, and $\delta = 0.97$. To measure the performance of each algorithm, we use the formula

$$PF = 100\% - \left(\left|\frac{c_{\max(b)} - c_{\max(a)}}{c_{\max(b)}}\right| \times 100\%\right) \quad (4)$$

where $C_{\max(a)}$ is the results from each algorithms and $C_{\max(b)}$ is the optimal value for Lawrence data that provided by Taillard (Taillard, 1993).

Table 1: Summary of benchmark data.

| Job | Machine | Problem Names |
|---|---|---|
| 10 | 5 | La05 |
| 15 | 5 | La09, La10 |
| 20 | 5 | La11, La14 |

The best makespan results obtained using MFASPV, Bidir, TS, and DFA are shown in Table 2. From Table 2 can be seen the comparison of the best makespan results from the Bidir, TS, DFA, and MDFA-SPV algorithms. Referring to Opt, the TS and MFASPV algorithms on JSSP is able to produce the best makespan for all benchmark, while DFA is able to produce the best makespan in four problems and the Bidir algorithm is able to produce the best makespan only in two problems.

Table 2: Summary of benchmark data.

| Problem Name | Opt | Bidir | TS | DFA | MFA SPV |
|---|---|---|---|---|---|
| La05 | 593 | 593 | 593 | 593 | 593 |
| La09 | 951 | 1017 | 951 | 951 | 951 |
| La10 | 958 | 958 | 958 | 958 | 958 |
| La11 | 1222 | 1259 | 1222 | 1222 | 1222 |
| La14 | 1292 | 1294 | 1292 | 1295 | 1292 |

To evaluate the performance of the four algorithms, we used (4). The percentage of performance is presented in Table 3. From Table 3 seen that the average performance of the Bidir and DFA algorithms is less than 100%, it means that the makespan results obtained using both algorithms have not been able to achieve Opt. While the average performance for TS and MDFA-SPV algorithms is 100%, which means the results of makespan obtained using both algorithms are able to achieve the Opt.

Table 3: Summary of benchmark data.

| Problem Name | Bidir | TS | DFA | MFASPV |
|---|---|---|---|---|
| La05 | 100 | 100 | 100 | 100 |
| La09 | 93.059 | 100 | 100 | 100 |
| La10 | 100 | 100 | 100 | 100 |
| La11 | 96.972 | 100 | 100 | 100 |
| La14 | 99.845 | 100 | 99.767 | 100 |
| **Mean** | **97.975** | **100** | **99.953** | **100** |

## 6 CONCLUSIONS

In this paper, our proposed algorithm which named as MFASPV is tested using Taillard benchmark problem available in the literature. MFASPV compared with previous results Bidir, TS, and DFA to find minimum value of makespan from the data benchmark that provided by OR-Library. The performance of MFASPV is found to be good and able to achieve the best for five Lawrence problems.

## REFERENCES

Bażewicz, J., Pesch, E. & Sterna, M., 2000. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2): 317–331.

Beasley, J. E., 1990. OR-Library: Distributing Test Problems by Electronic Mail. *The Journal of the Operational Research Society*, 41(11): 1069–1072.

Brucker, P., 2007. *Scheduling algorithms*. 5th ed. Berlin ; New York: Springer.

Dell'Amico, M. & Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3): 231–252.

Garey, M.R., Johnson, D.S. & Sethi, R., 1976. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2): 117–129.

Kuhpfahl, J., 2016. *Job Shop Scheduling with Consideration of Due Dates: Potentials of Local Search Based Solution Techniques*. Gabler Verlag.

Taillard, E., 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2): 278–285.

Tasgetiren, F., Chen, A., Gencyilmaz, G. & Gattoufi, S., 2009. Smallest Position Value Approach. In *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. Studies in Computational Intelligence. Springer, Berlin, Heidelberg: 121–138.

Tasgetiren, M. F., Sevkli, M., Liang, Y. & Gencyilmaz, G., 2004. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In *In Proceedings of the 2004 Congress on Evolutionary Computation (CEC'04*. 1412–1419.

Udaiyakumar, K. C. & Chandrasekaran, M., 2014. Application of Firefly Algorithm in Job Shop Scheduling Problem for Minimization of Makespan. *Procedia Engineering*, 97: 1798–1807.

Yang, X.-S., 2009. Firefly Algorithms for Multimodal Optimization. In *Stochastic Algorithms: Foundations and Applications*. Lecture Notes in Computer Science. International Symposium on Stochastic Algorithms. Springer, Berlin, Heidelberg: 169–178.

Yang, X.-S. & He, X., 2013. Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*, 1(1): 36–50.