ISSN:0975-9646

An Iterative Random PSO to minimize Total-flow-time in Flowshop Scheduling

M. M. Ghumare[#1], Mrs. L. A. Bawoor[*2], Dr. S. U. Sapkal[#3]

*Department of Computer Engineering[#1,*2],Savitribai Phule Pune University, Pune*
Department of mechanical Engineering[#3], Walchand College of Engineering, Sangali
India

***Abstract*— Flowshop scheduling is an interdisciplinary challenge of addressing major optimality criteria of minimizing makespan and total-flow-time. The enumerations for finding the probabilities for improving the utilization of resources turn this problem towards NP-Hard. Particle Swam Optimization (PSO) has proven its ability to find of near optimal solution when problem size is large. Local search techniques increase ability of PSO and prevent PSO from getting stuck into local optima. This paper proposes modification in PSO an Iterative Random PSO without use of any local search technique. It uses re-initialization of population at every iteration; due to which there is no possibility of getting stuck into local optima. The proposed method achieves total-flow-time objective of flowshop scheduling. Computational results are obtained with 110 benchmark instances of Taillard and compared for the total-flow-time criterion. Ultimately, 80 instances out of 110 best known solutions provided by $DPSO_{VND}$were improved by IRPSO. There is still scope of improving IRPSO for dataset of 5 machines, it gives near optimal solution but not better one.**

***Keywords*— Metaheuristics; particle swarm optimization (PSO); iterative random PSO (IRPSO); flowshop scheduling; total-flow-time(tft), optimization.**

I. INTRODUCTION

Flowshop scheduling is an interdisciplinary challenge and less contributed by computer science researchers. Flowshop scheduling is to processing n jobs on m machines. Major challenges to no-wait (continuous) flowshop scheduling include total flow time, makespan and computation time these are known as optimality criteria[1]. Flowshop scheduling is NP-Hard problem.

Flowshop scheduling aims to find out sequence on jobs(n) to be processed on all machines(m). Opted sequence must be able to minimize makespan or total-flow-time.

Flowshop scheduling can be done with greedy methods, dynamic programming and artificial intelligence techniques such as heuristics and metaheuristics; but it still faces problem in achieving optimality criteria. Dynamic programming techniques are computationally slow and cannot find out solution for large number of problem size. Heuristics are feasible to find solution for even greater problem size and also have computationally better performance. If problem size goes on increasing then finding out approximate optimal solution with heuristic is less feasible. To overcome this drawback we are adopting metaheuristics.

Metaheuristics work better on higher problem size giving approximate optimal solution. It uses local approach; constructive approach; or evolutionary approach. Metaheuristics include genetic algorithm and swarm intelligence techniques such as ant colony algorithm(ACO), bee colony algorithm(BCO), particle swarm optimization(PSO).

PSO is invented by James Kennedy and Russell Eberhart (1995) to synchrony of flocking behavior of birds' efforts to maintain an optimum distance between themselves and their neighbors [2].PSO is evolutionary algorithm that has local search ability.

PSO has broadly shown its ability is optimization. Following are some areas where PSO is used for optimization:

1. Data clustering [3];
2. Data classification [4];
3. Wireless sensor networks [5];
4. Cloud computing [6];
5. Production scheduling [7-20];

PSO has effective performance in production shop scheduling; including job shop scheduling, flowshop scheduling, open shop scheduling. Production scheduling can be optimized by optimizing one or more criteria among, total flow time, makespan, computation time.

Total-flow-time is contributed less as compare to makespan and computation time by the researchers. Minimizing tft results into increased productivity. Among flowshop and job shop scheduling, flowshop scheduling has less contribution; hence we opted to optimize flowshop scheduling. Flowshop scheduling with increased jobs and machines size are solved with PSO to minimize makespan, total-flow-time and computational time objectives.

B. Liu, L. Wang *et al.* [7] optimized no-wait flowshop by minimizing total flow time; where simulated annealing is used with PSO to enhance local search ability. C.-J. Liao and P. Luarn[8] used discrete version of PSO(DPSO); they incorporated a local search scheme using insertion and interchange mechanism into the proposed PSO algorithm (called PSO-LS), with this *tft* is minimized but it requires more computation time. Manas Ranjan Singh & S. S. Mahapatra [9] considered flexible flowshop scheduling for optimization with PSO using mutation and improved makespan optimization. Q-K. Pan, M. Fatih Tasgetirenc *et al.* [10] used discrete PSO DPSO and local search with variable neighborhood descent (VND) algorithm based on

variable neighborhood search (VNS); it minimizes makespan and total flow time for no-wait flowshop. I.-H. Kuo, S.-J. Horng *et al.* [11] used individual enhancement(IE) scheme with PSO for enhancement of local search ability and as well used random encoding to enhance global search ability of PSO and successfully minimized makespan. P. Damodaran, A. G. Rao *et al.* [12] proposed heuristics to modify particle position. Improvement in heuristic is proposed with constructive Knapsack heuristic and batch formation NEH heuristic for particle generation; they achieved minimization of makespan. M. Akhshabi, R. Tavakkoli-Moghaddam and F. Rahnamay-Roodposhti [13] extended PSO with memetic algorithm (MA) that hybridizes with a local search method; it significantly improved optimization of tft. S. Chowdhury, W. Tong, *et al.* [14] developed mixed-discrete particle swarm optimization (MDPSO) for diversity preservation. Separate diversity metrics and diversity preservation mechanism is carried out for continuous and discrete design variables. MDPSO gives better results in case of unconstrained problem set.

With the survey [15] we have found that PSO with heuristics for local search increases performance of PSO called as hybrid PSO (HPSO). HPSO is able to optimize makespan, tft and computation time.

In this paper we introduce modified PSO called iterative random PSO (IRPSO) without using any local search algorithm considering total-flow-time objective. Computational results depict efficiency of IRPSO as compared with discrete PSO by Quan-Ke. Pan, M. Faith Tasgetiren, Yun-Chia Liang [10].

In section-II we discuss about basics of flowshop scheduling; section-III gives PSO algorithm; section-IV describes proposed PSO (*i.e.* IRPSO) and section-V and section-VI depicts experimental setup and computational results respectively; section-VII concludes the paper.

II. FLOWSHOP SCHEDULING

Flowshop scheduling is given as, set of n jobs and m machines to be processed in processing time (p_{ij}) *i.e.* processing time of i^{th} job on j^{th} machine with the only constraint that each job is to be processed once on every machine. Sequence α given with $\{\alpha_1, \alpha_2 \ldots, \alpha_n\}$ is n jobs to be processed on m machines. Ability of flowshop to allow processing of jobs on distinctive machines with constant sequence α allows each possible sequence to get considered for scheduling. Given matrix of size n x m with processing time p_{ij} will have (n!) number of permutations *i.e.* feasible solution from which optimal sequence is to be opted.

Delay between two machines is calculated by δ time. $\delta_{i,k}$ is minimum delay time between start of job i and start of job k where job k follows job i. ($1 \le i \le n$, $1 \le k \le n$, $i \ne k$). Delay matrix represents delay time between current job and next job to be submitted.

Makespan is total time requires for one job to complete from all machines. C_i is completion time of i^{th} job given by *eq.* (1). $\delta_{i,k}$ is given by by *eq.* (2). TFT is number total time required by all machines to complete entire given jobs by *eq.* (3). The problem to determine a sequence α that minimizes TFT.

For i= 1, 2,…., n and j= 1, 2,…., m

$$C_i = \sum_{j=1}^{m} p_{i,j} \quad (1)$$

$$\delta_{i,k} = p_{i,1} + \max\left(\sum_{j=1}^{r} p_{i,j} - \sum_{j=1}^{r-1} p_{k,j}, 0\right) \text{ where } 2 \le r \le \quad (2)$$

$$TFT = \sum_{i=1}^{n} C_i + \sum_{i=1}^{n} (n+1-i)\delta(\alpha_{i-1,i}) \quad (3)$$

When number of machines and jobs increases, enumerations to find out total permutation also increase drastically. Flowshop scheduling is NP-Hard more than three machines. Conventional algorithms with exact method to find of sequence of jobs on machines are not able to provide solution for more than three machines and three jobs. Thus we are adopting metaheuristic particle swarm optimization (PSO) to optimize flowshop scheduling.Page Layout

Your paper must use a page size corresponding to A4 which is 210mm (8.27") wide and 297mm (11.69") long. The margins must be set as follows:

- Top = 19mm (0.75")
- Bottom = 43mm (1.69")
- Left = Right = 14.32mm (0.56")

Your paper must be in two column format with a space of 4.22mm (0.17") between columns.

III. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is algorithm that start with initialization of population (*i.e.* swarm). Swarm is initialized with position and velocity. Search space is where swarms move to get appropriate position. Particle changes their position and velocity with following equations. Particles have their own best coordinates *i.e.* known as *pbest.* Particle finds their best position by coordinating with neighborhood particle and this is known as *lbest*. When particle choose best value from their entire topological neighbor then this best value is known as *gbest*. Finding the *gbest* from *lbest* boosts performance of PSO. Thus it is challenging to find *lbest* as value of lbest contributes to gbest.

$$V_i(k+1) = \omega * V_i(k) + c_1 * \text{rand}(\,) * (P_i(k) - X_i(k)) + c_2 * \text{rand}() * (g(k) - X_i(k)) \quad (4)$$

$$X_i(k+1) = X_i(k) + V_i(k+1) \quad (5)$$

Where,
ω is inertia weight
$V_i(k)$ is velocity of particle i at iteration k.
$X_i(k)$ is the position of particle i at iteration k.
$V_i(k+1)$ is velocity of particle i at iteration k+1.
$X_i(k+1)$ is the position of particle i at iteration k+1.
rand( ) is random number between (0,1).
c1 cognitive acceleration coefficient.
c2 social acceleration coefficient.

IV. ITERATIVE RANDOM PSO

From survey of "Application of Particle Swarm Optimization for Production Scheduling" [15] we have analyzed that, parameters of PSO that affect performance are,

i. Local and global acceleration coefficient,
ii. Inertia weight,
iii. Initialization of particles.
iv. Termination Criteria

We have focused on these basics, and successfully modified these parameters to increase performance of PSO without use of any heuristic for local search. We are proposing PSO with slight modification in dynamics mentioned above; modified algorithm is given below.

Particles update their position depending on choice of initial population and finds *pbest* and *lbest*. If the initial population choice goes incorrect; updating particles at every iteration may not give near optimal solution tough local search applied. Thus choosing initial population randomly may affect results of PSO. To overcome this drawback we are proposing new method called **Iterative Random PSO** (IRPSO).

IRPSO uses random initialization of particles at every iteration and updates the particles depending on given initialization. Each job is kept constant on first position for random initialization.

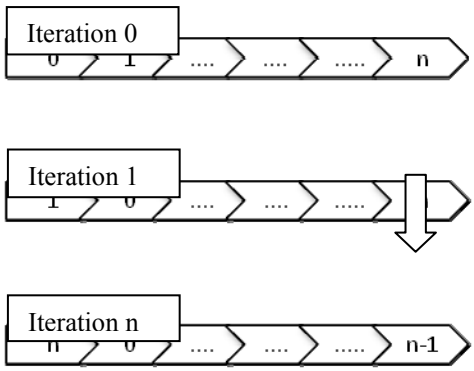

Fig.1 Population Initialization at Each Iteration

Velocity is initialized with the following equation, and random value is taken between 0 to 1.

$$v_{min} + (v_{max} - v_{min}) * \text{random}(0,1) \qquad (6)$$

At first iteration when particle is initialized, velocity of particles is calculated with eq. (1). Afterwards the velocity of the particles is updated using particle at global best velocity.

Updating particles at every iteration is done with velocity and position change using Eqs. (4) & (5). Each of this iteration finds a local best which in turns contribute to find out global best.

Linearly decreasing value of inertia weight (ω) may lack global search ability of PSO at the end of run. As re-initialization of particles at every run avoids PSO to lack global search ability; ω is initialized with constant value (0.2).

Total number of evaluation and iteration causes premature convergence in PSO. To avoid premature convergence; algorithm is terminated after *n* runs i.e. when n^{th} job is initialized at first position algorithm is terminated.

PSO has always been found to stuck into local optima. Tough use of heuristics for local search with PSO has less possibility to get stuck to local optima; it may get stuck to local optima. IRPSO will never get stuck to the local optima due to re-initialization of population after each iteration.

Algorithm for IRPSO:

Algorithm stated in fig.1 describes working of IRPSO, i_size is number of iterations which is equal to number of jobs *n* and p_size is particle count which is also *n*.

Algorithm IRPSO:

```
begin
    for: i := 0 to i_size do
        Initialize particle with random initialize vi and xi
        Sort particle
        Find out lbest
        for: i := 0 to i_size do
            Update particles using eqs. (4) & (5)
            (Velocity of particle lbest for updating position)
            Find lbest in updated particles
        endfor
        Compare lbest with gbest
        if lbest < gbest then
            gbest = lbest
        endif
    endfor
    random()
    begin
        for: j := 0 to p_size do
            seq[]={j, random arrangement of p_size}
        endfor
    end
end
```

Fig 2. Pseudo-code for IRPSO

V. EXPERIMENTAL SETUP

According to metaheuristic PSO discussed in Section I, we applied the iterative random PSO described in Sections III and IV.

Parameters to be considered in PSO implementation are local and global acceleration coefficient c1 and c2. Values for c1 and c2 were finalized as given in table below,

TABLE I
VALUES FOR C1, C2

m	c1	c2
10	2.7	2.05
20	3.3	2.7

This provides a way for a fair comparison of discrete HPSO by controlled and unbiased experiments, which conforms to some of the criticism discussed by Quan-Ke. Pan *et al*. (2008). For example, we simply switch the neighborhood criterion to be used without any other change (Without any local search heuristics). We apply the metaheuristics for benchmark data sets of Taillard (1993) which have been generated for unrestricted flow-shop scheduling. The data sets are available from the OR library. We use problem instances with 20 (ta011–ta030), 50 (ta041– ta060) and 100 (ta071–ta090). All computations were performed using Intel CORE i5 processer 2.50GHz. Totally, 50 out of 60 best known solutions provided by $DPSO_{VND}$ were ultimately improved by proposed IRPSO.

VI. Computational Results

To compare the proposed heuristic with the existing heuristics, we carried out the experimentation by considering problem sizes with number of jobs (n) = 20, 50 and 100 and number of machines (m) = 10 and 20. Ten independent problem instances were considered for each problem size. Performance is measured with relative percentage deviation (RPD) and average relative percentage deviation (ARPD) given by *eq. 7* and *eq. 8* respectively,

$$RPD = \left(\frac{(DPSO_{VND} - IRPSO)}{IRPSO}\right) \times 100 \qquad eq.\ (7)$$

$$ARPD = \left(\frac{\sum_{k=1}^{k} RPD}{k}\right) \qquad eq.\ (8)$$

Table 2 displays comparative evaluation of the proposed heuristic and $DPSO_{VND}$ based on the ARPD and the percent of best heuristic solutions. The results show that average ARPD of the proposed heuristic is significantly less compared to the existing method. With respect to RPD, the proposed method performs better than either $DPSO_{VND}$ for 50 out of 60 instances.

All results calculated below are collected from five runs. Positive values of ARPD predicts effectiveness of IRPSO with respect to $DPSO_{VND}$ whereas, negative value shows IRPSO is not able to provide near optimal solution.

TABLE III
PERFORMANCE COMPARISON OF $DPSO_{VND}$ WITH IRPSO

Sr. No.	Instances	n × m	ARPD
1.	ta011-ta020	20 × 10	**6.124**
2.	ta021-ta030	20 × 20	**10.928**
3.	ta041-ta050	50 × 10	**2.187**
4.	ta051-ta060	50 × 20	**7.717**
5.	ta071-ta080	100 × 10	**-7.646**
6.	ta081-ta090	100 × 20	**1.322**

Time required to by IRPSO for different set of jobs is given in table below,

TABLE III
TIME TAKEN FOR EXECUTION

Jobs × Machines (n × m)	Time (Seconds)
20 × 20	0.72
50 × 20	5.97
100 × 20	38.95

VII. Conclusions

This paper has proposed an efficient meta-heuristic algorithm, Iterative Random PSO (IRPSO) in order to solve the no-wait flow shop scheduling problem minimizing the total flow time. The results obtained from proposed IRPSO are compared with Discrete PSO algorithm hybridized variable neighborhood descent ($DPSO_{VND}$). The results have clearly proved that the proposed IRPSO has substantially outperformed $DPSO_{VND}$ for 10 and above machines. Results have shown to be statistically significantly better than those produced by $DPSO_{VND}$. IRPSO avoid to get stuck in local optima due to re-initialization of population at every iteration.

If the problem size is less than 200 then IRPSO produces promising results without overhead of local search technique. If problem size increases it is recommended to use local search heuristic to get near optimal solutions.

Future research directions can be recommenced as follows: (1) modifying IRPSO for more than 200 jobs. (2) considering multi-objective functions at different practical constraints, and (3) expanding the proposed algorithm to other similar scheduling problems.

References

[1] P.K. Gupta, D.S. Hira, *"Operation Research"*, pp. 404-406,S. Chand,1983 J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceeding. ICNN'95 - International. Conference Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[2] Yuhui Shi and Russell Eberhart, *"Empirical study of particle swarm optimization"*, Evolutionary Computation, 1999

[3] Sandeep Rana, Sanjay Jasola, Rajesh Kumar," A review on particle swarm optimization algorithms and their applications to data clustering", Artificial Intelligence Review , Volume 35, Issue 3, pp 211-222 , March 2011

[4] Xue B, Zhang M, Browne WN, "Particle swarm optimization for feature selection in classification: a multi-objective approach." IEEE Transactions on Cybernetics, 2013

[5] Kulkarni, R.V., Venayagamoorthy, G.K. "Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey ",IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, (Volume:41 , Issue: 2), July 2010

[6] C. Tsai, J. J. P. C. Rodrigues, and S. Member, *"Metaheuristic Scheduling for Cloud : A Survey,"* IEEE SYSTEMS JOURNAL, vol. 8, no. 1, pp. 279–291, 2014.

[7] B. Liu, L. Wang, and Y.-H. Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 31, no. 9–10, pp. 1001–1011, Jan. 2006.

[8] C.-J. Liao and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3099–3111, Oct. 2007.

[9] M. R. Singh and S. S. Mahapatra, "A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks," *Int J Adv Manuf Technol*, vol. 62, no. 2012, pp. 267–277, 2012.

[10] Q.-K. Pan, M. Fatih Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop

scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2807–2839, Sep. 2008.

[11] I.-H. Kuo, S.-J. Horng, T.-W. Kao, T.-L. Lin, C.-L. Lee, T. Terano, and Y. Pan, "An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 7027–7032, Apr. 2009.

[12] P. Damodaran, A. G. Rao, and S. Mestry, "Particle swarm optimization for scheduling batch processing machines in a permutation flowshop," *Int. J. Adv. Manuf. Technol.*, vol. 64, no. 5–8, pp. 989–1000, Mar. 2012.

[13] M. Akhshabi, "A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time," *Int J AdvManuf Technol*, vol. 72, no. 2014, pp. 1181–1188, 2014.

[14] S. Chowdhury, W. Tong, A. Messac, and J. Zhang, "A mixed-discrete Particle Swarm Optimization algorithm with explicit diversity-preservation," *Structural Multidisciplinary. Optimization.*, vol. 47, no. 3, pp. 367–388, Dec. 2012.

[15] M. M. Ghumare, "Application of Particle Swarm Optimization for Production Scheduling," *ICCUBEA*, vol. 1, no. 2, DOI 10.1109/ICCUBEA.2015.100, 2015.

[16] R. Ruiz, C. Maroto, and J. Alcaraz, "Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics," *Eur. J. Oper. Res.*, vol. 165, no. 1, pp. 34–54, Aug. 2005.

[17] A. Abraham, Fatos Xhafa, *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*. 2008.

[18] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," *Comput. Oper. Res.*, vol. 32, no. 8, pp. 2197–2212, Aug. 2005

[19] C.-T. Tseng and C.-J. Liao, "A discrete particle swarm optimization for lot-streaming flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 191, no. 2, pp. 360–373, Dec. 2008.

[20] Y. Marinakis and M. Marinaki, "Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem," *Soft Comput.*, vol. 17, no. 7, pp. 1159–1173, Feb. 2013.

[21] M. Rabiee, R. S. Rad, M. Mazinani, and R. Shafaei, "An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines," *Int J AdvManuf Technol*, vol. 71, no. 2014, pp. 1229–1245, 2014