Single Objective for Partial Flexible Open Shop Scheduling Problem using Hybrid Particle Swarm Optimization Algorithms

M. Nagamani¹ and E. Chandrasekaran²

¹Global Institute of Engineering and Technology, Vellore - 632509, Tamil Nadu, India; nagamanim1983@gmail.com
²Veltech University, Avadi, Chennai - 600085, Tamil Nadu, India; e_chandrasekaran@yahoo.com

Abstract

Objectives: Scheduling is an optimization problem in computer science and operation research in which ideal jobs are assigned to particular times. Nowadays, the problem is presented as an online problem, that is, each job is presented and the online algorithm needs to make a decision about the job before the next job is presented. Methods and Statistical Analysis: The proposed approach is used to solve the open job scheduling problem in which any job can be connected with the available machine. That was implemented in matlab to available best results with hybrid evolutionary algorithm. Findings: In this paper, a hybrid algorithm based on the particle swarm optimization is proposed, for flexible, open shop scheduling problem, to minimize the make-span. First an effective new approach using two decisions based on parallel priories dispatching rules is applied. Next we develop a hybridizing HPSO, that presents new components for updating velocity and position using evolutionary operators, with an adaptive neighbourhood procedure based on the insert-interchange fitness function, selection, mutation, crossover. Application/Improvements: The performance of the proposed a new hybrid algorithm is compared to other benchmark problems.

Keywords: Dispatching Rules, Evolutionary Operators, Flexible Open Shop Problem, Local Search, Particle Swarm Optimization

1. Introduction

The flexible, open shop scheduling problem (FOSP) can be defined by a set of n jobs to be scheduled through s stages in series. Each stage j (j = 1, 2, ........s) has m_j identical and parallel machines. The job i (i = 1, 2, ...., n) require a processing time p_{ij} at stage j, and has to be processed without preemptive by exactly only one machine at one stage. The objective is to find a schedule of jobs, which would minimize the make-span. Many research and researches in the literature have been attempted to solve the flexible, open shop, using several approaches such as exact methods, heuristic and metaheuristic. Branch and bound is the only method widely used to solve the FOSP. The proposed a branch and bound method based on m-machines problems. The presented an enhanced branch and bound procedure based on energetic reasoning and of global operations. The applied a branch and bound algorithm for a flexible, open shop scheduling problem with setup time and assembly operations.

An efficient heuristic algorithm for the special case when the second stage contains only the one machine. The proposed metaheuristic algorithms for a two-stage flexible, open shop with and of multiprocessor tasks. The presented an effective parallel ambition algorithm to solve HFS with multiprocessor tasks. Recently, several metaheuristics have been described to solve the HFS. The presented a parallel Tabu Search (TS) to solve large, complicated size problem instances of HFS. Developed diverged approaches based on Evolutionary Algorithms (EA) for flexible, open shop with multiprocessor task of problems. An ant colony optimization for HFS is introduced an approach hybridizing particle swarm optimization with bottle neck metaheuristic and proposed a hybrid discrete
particle swarm optimization for the no-idle permutation flexible open-shop scheduling problem. An improved advanced cuckoo algorithm to minimize the maximum completion of time. The presented two algorithms inspired by the natural immune system (QIA), with the objectives to minimize the make-span and the mean flowtime. The proposed new approaches based on an artificial immune system for HFS. In this paper, we will present a hybrid particle swarm algorithm incorporated with mutation-based local search, that operates by the use of compound neighbourhood structures. Due to the importance in solution method, the objective function would be calculated by a new heuristic, that consist to combine parallel priority dispatching rules to assign jobs to machines at each stage.

2. Hybrid Particle Swarm Optimization Algorithm

Hybrid Particle Swarm Optimization (HPSO) is an evolutionary biologically inspired optimization, based on the behaviour and intelligence of swarms. It was first originally developed by Kennedy. HPSO is initialized by a population of random numbers uniformly generated in the interval [0,1].

\[
X_i^t = c_2 \otimes F_2(c_1 \otimes F_1(X_i^{t-1}), P_i^{t-1}), G^{t-1}) \tag{3}
\]

Note that \(X_i^t\) is the position of the particle is its \(P_i^t\) personal and best position, and \(G_i^t\) is the best position of the whole particles in the swarm. The updated equation consisting of three components:

- The second component represents the "cognition" part of the particle and \(f_1\) is the crossover operator with the probability of \(c_1\).
- The third component this corresponds to the social part of the particle \(f_2\) of the particle, \(f_3\) the crossover operator with the probability of \(c_2\).

In addition, to that we add a new term in equation (3) that represents the best neighbor found by the neighbouring structures[1]. The particle will be updated as follows:

\[
X_i^t = c_4 \otimes F_4(c_2 \otimes F_2(c_1 \otimes F_1(X_i^{t-1}), P_i^{t-1})), G^{t-1}) \tag{4}
\]

The operator \(f_4\) corresponds to the local search applied and to the particle with the probability of \(c_4\).

2.2 Mutation Operator

In the proposed algorithm the inverse mutation is used, it works as stated below:

- Two positions are randomly selected in the sequence of its order.
- In this portion between these two positions is inverted.

2.3 Crossover Operators

Two crossover operators are used here:

2.3.1 Uniform Crossover

Random binary masks with the same size of the parents are generated. The (0) of the mask define the positions preserved to the first parent, and the (1) of the mask corresponds to the positions preserved to the second parent. The illustration of the uniform crossover is given in below.
2.3.2 Right Corner Crossover

Firstly proposed in, this operator starts by choosing randomly two positions from the first parent. The block determined by the two point has moved to the right corner of the offspring. This is the complete the remaining jobs from these condparent.

2.4 Mutation based Local Search

In our paper, we strongly propose a mutation-based local search referring to the well known NEH method. The proposed local search starts from an initial solution, and attempts to improve the present solution by generating compound neighborhood structures. More or formally, two neighborhoods are defined, the insert neighborhood and the interchange neighborhood. The insert neighborhood is created by insert moves, that consist to remove the job currently in the below Figure 1.

Position I and insert it into another position j. The interchange neighborhood, uses per pair wise interchange moves that interchange two random positions in the job sequence to obtain a fresh new mutated sequence. The procedure operates by extending the both neighborhoods Consecutively, by exploring the insert neighborhood first, then the interchange second neighborhood. This approach combines two local search structures, with the aim of exploring effectively the solution space and improves the exact convergence. The following is the pseudo code of the mutation-based local search:

**Mutation-based Local Search Pseudo Code procedures**

Input: \( \pi_0 \) the initial solution.
Output: \( \pi^* \) the best solution found so far.

\[
\pi^* \leftarrow \pi_0
\]

Repeat Until a given stopping criterion is met completely.

\[
\pi^0 = \text{LS}(\pi_0), \text{ the local search based on insert neighborhood.}
\]

\[
\pi^0 = \text{interchange} \quad \text{LS}(\pi^0), \text{ the local search procedure based on a new interchange neighborhood.}
\]

If \( \pi^0 \) is better than \( \pi^* \), then \( \pi^* \leftarrow \pi^0 \).
End if

\[
\pi_0 \leftarrow \pi^0
\]

2.5 Make-span Metaheuristic

We consider an effective metaheuristic a calculation of the makespan \( (C_{\text{max}}) \), characterized by the use of two decision methods based on priority dispatching algorithm, including FIFO (First In First Out), LPT (Longest Processing Time). The SPT (Shortest Processing Time): The proposed metaheuristic combines the classical list scheduling, wherein the jobs are assigned at the first available machine according FIFO rule, and a modified fully list scheduling that uses three parallel priority dispatching algorithm: FIFO, FIFO+LPT and FIFO+SPT, and then adopts at each stage the rule that generates the sequence giving a smallest completion time. The heuristic makes a choice between the two scheduling lists described above and selects the one that gives minimize value of the make - span.

3. The Proposed Algorithm

In our proposed approach incorporates hybrid PSO, mutation-based local search and the make-span metaheuristic. The DPSO well assures the diversification and a large reexploration of the solution space. However the local search is employed more to intensify the search and do improve the convergence. In order to increase the quality of solution evaluation, the make-span metaheuristic takes advantage of two decision method at atime. We summarize the steps of the proposed algorithm given below:

Step 1: Generate only initial population randomly.
Step 2: Evaluate the particles using the metaheuristic method
Step 3: Find P best and G best accordingly.
Step 4: Update the particles using (4) equation
Step 5: Evaluate the particles in the swarm optimization
Step 6: Find P best and G best.
Step 7: Stop if the stopping criterion is met, or otherwise return to step 4.

3.1 Benchmark Scheme

The performance of the proposed hybrid algorithm was being tested on benchmark problems that are largely used in the said literature. The benchmark problems consist of 77 instances, divided into 53 easy problems and 24 hard problems. Accordingly the machine configuration plays a vital role on the complexity of problems. There are four machine configurations a,b,c and d, which corresponds
to the bottleneck stage. The following is the meaning of the letters of machine configuration as stated

- There is one machine at the middle stage (bottleneck) traffic, and three machines at the other stages of the traffic.
- There is one machine at the first stage of bottleneck and three machines at the other stages. In the line
- There are two machines at the middle stage (bottleneck), and three machines at the other stages stated.
- There are three machines at each stage and there is no bottleneck stage.

For example, the notation j10c5b3 means 10 jobs, 5 stages and the letter b define the machine configuration, where there are other three machines at each stage except the first stage which is bottleneck with only one machine left. The letters j and c are the abbreviations of job and stage stated respectively.

4. Numerical Results

In our computational experiments, we consider the 24 covehard problems. The comparison was performed using four algorithms:

1. The Immunoglobulin-based Artificial Immune System algorithm (IAIS)
2. The Ant Colony Optimization (ACO)
3. The Artificial Immune System algorithm (AIS)
4. The Quantum-inspired Immune Algorithm (QIA)

The computing environment of all the algorithms is dissimilar. For this reason, the comparison is made on the basis of the solution quality, evaluated by the percentage deviation between the solution and the greatest Lower Bound (LB) which is defined as stated below:

\[
Relative\ Deviation = \left( \frac{C_{best} - LB}{LB} \right) \times 100
\]

The hybrid algorithm was limited to and with 1600s, or otherwise to the lower bound was attained. If the lower-bound was not found within the limited time, the search was stopped orbit and the best solution was accepted as the final solution. The proposed algorithm was implemented in C++ and was run ten times to obtain the best Cmax value alone. Note that for the four compared algorithms as stated, IAIS, ACO and AIS are also limiting their runtime on 1600s. However QIA was running a limited and fixed number of iterations. For all considered algorithms, the numerical results were obtained from their original papers alone. With reference to the computing environment, the IAIS algorithm was programmed in C++, the ACO was implemented using Microsoft Visual Basic studio software, the algorithm AIS was coded in Excel, Microsoft, and QIA was coded in Matlab.

There are four essential parameters in our hybrid method, the Population size Ps, the probability of mutation \(\omega\), the crossover probabilities \(c_1\) and \(c_2\), and the local search probability \(c_3\). We implemented one mindedly our algorithm with \(Ps = 20\) and \(c_1 = c_2 = 0.8\). For \(\omega\) and \(c_3\), a parametric study was established by the set of values \(\{0.1, 0.2, ..., 0.9\}\). Three problems j10c5c1, j10c5d1 and j15c5c5 are considered from the benchmark problems. For each parameter value, 20 tests were carried out. Table 1 and 2 illustrate foreach parameter, the number of times the Lower Bound (LB) was attained. We were using the parameters with high number of times LB was attained, thus attained \(\omega = 0.4\) and \(c_3 = 0.4\).

The numerical comparisons of HPSO algorithm, IAIS, ACO, AIS and QIA are given below in table 2, where columns represent the make span \(C_{max}\) in seconds, the Lower Bound (LB) and the percentage of deviation is calculated between the Lower Bound (LB) and \(C_{max}\) described in equation (5). HPSO can solve 18 problems out of 24 hard problems, that representative (75%), whereas IAIS and AIS solve 16 problems (66.7%). The ACO can solve 12 problems of the 18 problems respectively. In table 1, we provided the performance of the HPSO algorithm among the other compared algorithms, where the first column represents the percentage of solved problems (% solved problems) and these conndcolumn gives the average percentage of deviation of the 24 hard problems (% deviation). The third column explains the number of problems considered among the 24 hard problems (number of pb).

| Problem | % solved problems | % deviation | Number of Pb |
|---------|-------------------|-------------|--------------|
| HPSO    | 74.3              | 2.80        | 24           |
| IAIS    | 66.7              | 3.02        | 24           |
| QIA     | 60.0              | 5.04        | 12           |
| ACO     | 66.7              | 4.10        | 18           |
| AIS     | 66.7              | 3.13        | 24           |

Table 1. The performance of HPSO algorithm
5. Conclusions and Perspectives

In this paper has been examined the hybrid open shop problem, with the sole objective to minimize the makespan. We have proposed a method HPSO algorithm to solve the problem. The proposed HPSO has been used mutation operator and crossover operators to update the positions of the particles in the swarm. The developed HPSO incorporates the mutation-based local search, which combines two local search strategies based on the inserted neighborhood and the interchanged neighborhood. In order to improve the performance of the evaluation, the above said make-span metaheuristic introduced in our HPSO algorithm combines two decision methods based on priority dispatching rules. The performance of the proposed HPSO has been tested and proved on benchmark problems, and compared to four different algorithms from the literature. The computational mathematical results perform the efficiency of our algorithm. The future works may consider other scheduling problems, such as hybrid open shop with various objectives.

6. References

1. Bochenek B, Fory´s P. Structural optimization for post-buckling behavior using particle swarms. Structural Multidisciplinary Optimization. 2006; 32(6):521–31.
2. Carlier J, N´eron E. An exact method for solving the multi-processor flow-shop. RAIRO Operations Research. 2000; 34(1):1–25.
3. N´eron E, Baptiste P, Gupta JND. Solving hybrid flow shop problem using energetic reasoning and global operations. Omega, 2001; 29(6):501–11.
4. Fattahi P, Hosseini SMH, Jolai F, Tavakkoli-Moghaddam R. A branch and bound algorithm for hybrid flow-shop scheduling problem with setup time and assembly operations. Applied Mathematical Modelling. 2014; 38(1):119–34.

5. Oguz C, Ercan MF, Edwin Cheng TC, Fung YF. Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. European Journal of Operational Research. 2003; 149(2):390–403.

6. Kahraman C, Engin O, Ihsan K, Ozturk RE. Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach. Applied Soft Computing. 2010; 10(4):1293–300.

7. Bozejko W, Pempera J, Smutnicki C. Parallel tabu search algorithm for the hybrid flowshop problem. Computers and Industrial Engineering. 2013; 65(13):466–74.

8. Pan QK, Wang L. No-idle permutation flowshop scheduling based on a hybrid discrete particle swarm optimization algorithm. The International Journal of Advanced Manufacturing Technology. 2008; 39(7):796–807.

9. Niu Q, Zhou T, Ma S. A quantum-inspired immune algorithm for hybrid flowshop with make span criterion. Journal of Universal Computer Science. 2009; 15:765–85.

10. Hamid T, Ali ARH, Mehdi Y, Touraj M. Using gravitational search algorithm for in advanced reservation of resources in solving the scheduling problem of works in workflow workshop environment. Indian Journal of Science and Technology. 2015; 8(11):1–16.

11. Engin O, Ceran G, Yilmaz MK. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. Applied Soft Computing. 2011; 11(3):3056–65.

12. Engin O, Doyen A. A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Generation Computer Systems. 2004; 20(6):1083–95.

13. Chung TP, Liao CJ. An immunoglobulin-based artificial immune system for solving the hybrid flowshop problem. Applied Soft Computing. 2013; 13(8):3729–36.

14. Bharathi T, Krishnakumari P. Application of modified artificial fish swarm algorithm for optimizing association rule mining. Indian Journal of Science and Technology. 2014; 7(12):1906–15.

15. Kennedy J, Eberhart R. Particle Swarm Optimization. IEEE Conference on Neural Networks. 1995; 4. p.1942–48.

16. Pan QK, Tasgetiren M F, Liang Y C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computer and Operations Research. 2008; 35(9):2807–39.

17. Liao CJ, Tjandradjaja E, Chung TP. An approach using particle Swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. Applied Soft Computing. 2012; 12(6):1755–64.

18. Serifoglu FS, Ulusoy G. Multiprocessor task scheduling in multistage hybrid flow-shops: A genetic algorithm approach. The Journal of the Operational Research Society. 2004; 55(4):504–12.

19. Alaykyran K, Engin O, Doyen A. Using ant colony optimization to solve hybrid flowshop scheduling problems. The International Journal of Advanced Manufacturing Technology. 2007; 35(5):541–50.

20. Marichelvam MK, Prabaharan T, Yang XS. Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize make span. Applied Soft Computing. 2014; 19(10):93–101.