International Journal of Industrial Engineering Computations 7 (2016) 671–680

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Heuristics for no-wait flow shop scheduling problem

Kewal Krishan Nailwal[a*], Deepak Gupta[b] and Kawal Jeet[c]

[a]*Department of Mathematics, A.P.J College of Fine Arts, Jalandhar, Haryana, India*
[b]*Department of Mathematics, M.M. University, Mullana, Ambala, Haryana, India*
[c]*Department of Computer Science, D.A.V. College, Jalandhar, Punjab, India*

C H R O N I C L E

Article history:
Received November 4 2015
Received in Revised Format December 21 2015
Accepted February 25 2016
Available online February 25 2016

Keywords:
Flow shop scheduling
Makespan
Heuristic
No-wait

A B S T R A C T

No-wait flow shop scheduling refers to continuous flow of jobs through different machines. The job once started should have the continuous processing through the machines without wait. This situation occurs when there is a lack of an intermediate storage between the processing of jobs on two consecutive machines. The problem of no-wait with the objective of minimizing makespan in flow shop scheduling is NP-hard; therefore the heuristic algorithms are the key to solve the problem with optimal solution or to approach nearer to optimal solution in simple manner. The paper describes two heuristics, one constructive and an improvement heuristic algorithm obtained by modifying the constructive one for sequencing *n*-jobs through *m*-machines in a flow shop under no-wait constraint with the objective of minimizing makespan. The efficiency of the proposed heuristic algorithms is tested on 120 Taillard's benchmark problems found in the literature against the NEH under no-wait and the MNEH heuristic for no-wait flow shop problem. The improvement heuristic outperforms all heuristics on the Taillard's instances by improving the results of NEH by 27.85%, MNEH by 22.56% and that of the proposed constructive heuristic algorithm by 24.68%. To explain the computational process of the proposed algorithm, numerical illustrations are also given in the paper. Statistical tests of significance are done in order to draw the conclusions.

1. Introduction

Scheduling is regarded as decision making process in manufacturing and serving industries to allocate the resources to tasks over a given time interval to optimize one or several criteria. With different industrial setups, there are different forms of resources and tasks. Flow shop scheduling deals with processing of jobs through machines in a particular manner to optimize a given criterion. The optimization can be the minimization (cost) or maximization (profit) related to the problem. In other words, flow shop consists of *m*-machines in series. Every job is to be processed on all the *m*-machines. The jobs have to adopt a fixed technological route for processing on each machine i.e., they have to be processed first on machine 1, then on machine 2, and so on. After completion on one machine, a job joins the queue at the next machine (Pinedo, 2010). The problem related to processing of *n*-jobs through *m*-

* Corresponding author. Tel: +919815077469
E-mail: kk_nailwal@yahoo.co.in (K. K. Nailwal)

doi: 10.5267/j.ijiec.2016.2.005

machines in a flow shop is a typical combinatorial optimization problem. For *n*-jobs to be processed on *m*-machines in a flow shop scheduling, there are *n*-factorial distinct sequences of jobs possible for each machine and hence $(\lfloor n)^m$ distinct possible schedules. To calculate the sequence from such a large number of possibilities to optimize the given measure of performance is really a tedious task (Baker, 1974). Makespan as a measure of performance is widely studied problem and is defined as the total time elapsed when the set of all jobs completes processing on all the machines. The objective for this measure of performance is to complete all the jobs as early as possible. If a job processing order on all machines is maintained throughout in a schedule, then the schedule is defined to be a permutation schedule. Several hundreds of research papers in scheduling can be found solving the problem $F_m / prmu / C_{\max}$ which is regarded as the classical problem in literature of the scheduling. The sub-problem $F_2 / prmu / C_{\max}$ put forward by Johnson (1954) finds the optimal solution of the problem. This finding paves the way for this branch of scheduling. One of the important heuristic for $F_m / prmu / C_{\max}$ that exists in the literature is Nawaz et al. (1983) known popularly as NEH. The problem of minimizing makespan under continuous environment i.e. no-wait of jobs is written in the form $F_m / no-wait / C_{\max}$. The continuous flow shop originates in the scheduling theory because of the production environment in industry. In many flow shops, the production environment is such that the delay in job processing between the subsequent machines is not allowed i.e. the assumption of infinite storage capacity between the machines in flow shop is no longer valid. For maintaining the continuous flow of jobs, the processing of jobs is delayed on the first machine so that the jobs do not wait in the subsequent processing on machines. Some such typical examples of manufacturing include metal casting, plastic manufacturing and food industries can be found in Aldowaisan and Allahverdi (2004). For example, the process of making iron sheets in industry involves the no-wait situation as the sequence in which the jobs are processed after the heating of iron is to be continuous so that the temperature of heated iron falls within the permissible interval specified. This constraint is necessary for the defect free production of iron sheets and making the good quality product. In food processing industry, the food is canned immediately after the food is prepared so that the food quality is maintained. However, to maintain freshness in the food the continuous flow in the sequence of jobs processing is maintained throughout the process.

The two criteria of minimizing makespan and total flow time in no-wait flow shop scheduling problems have been widely studied in the scheduling literature. The problem with no-wait constraint in flow shop scheduling with minimization of total flow time as criterion has been studied by Van Deman and Baker (1974), Rajendran and Chaudhuri (1990), Chen et al. (1996), Aldowaisan and Allahverdi (1998, 2004), Allahverdi and Aldowaisan (2000), Bertolissi (2000), Gao et al. (2013), Akhshabi et al. (2014) and Laha and Sapkal (2014). The problem of scheduling of jobs with the objective as makespan with no-wait constraint in flow shop has been studied by many researchers. The three machine flow shop problem for minimizing makespan as objective is proved to be NP-hard by Rock (1980), therefore the problem $F_m / no-wait / C_{\max}$ is also NP-hard. Thus, the solution of the problem $F_m / no-wait / C_{\max}$ can be better found by heuristic algorithms in better time frame. The heuristics solutions can broadly be categorized into two types: constructive and improvement solutions. The constructive algorithm is the one which builds job sequence by assigning jobs some priority or index using some procedure. Szwarc (1983) provided the solutions to flow shop problem without interruptions in job processing using Gilmore –Gomory's algorithm. Bonney and Gundry (1976), King and Spachis (1980) proposed a heuristic with makespan as objective. Gangadharan and Rajendran (1993) and Rajendran (1994) developed heuristics with a performance better than those of Bonney and Gundry (1976), King and Spachis (1980) based on preference relations and job insertion. Laha and Chakraborty (2009) proposed a heuristic based on the fundamentals of job insertion which builds an *n*-job sequence, incrementally. Plenty of constructive heuristics have been developed whereas the literature of improvement heuristics contains only few algorithms such as Komaki and Kayvanfar (2012). They proposed an improvement heuristic algorithm based on delay between adjacent jobs having two phases. The advance branch of heuristics regarded as metaheuristics can also be considered as the improvement heuristics. Fink and Vob (2003) provided the solution to flow shop problems which are continuous in nature using metaheuristics. Aldowaisan and

Allahverdi (2003) proposed hybrid heuristics for no-wait flow shop scheduling with makespan as objective and found better results than the heuristic of Rajendran (1994). Grabowski and Pempera (2005) presented heuristic algorithms for no-wait based on the traditional descending and tabu search approaches. Chaudhry and Munem Khan (2012) presented a spreadsheet based genetic algorithm (GA) approach to minimize makespan under no-wait situation for scheduling *n*-jobs through m-machines. Some other important metaheuristics including Pan et al. (2008a), Tseng and Lin (2010) and Ding et al. (2015). Ding et al. (2015) recently proposed constructive modified NEH (MNEH) as initial solution for metaheuristics with better performance than MNEH. Riyanto and Santosa (2015) proposed solution to no-wait flow shop scheduling by hybridization of ant colony optimization (ACO) algorithm with local search (LS). The bi-criteria problem with minimization of makespan and total flow time was solved by Pan et al. (2008). Hall and Sriskandarajah (1996) gave an exhaustive survey for the problems in flow shop under no-wait situation usually found in the manufacturing setups. Some noteworthy theoretical works in flow shop scheduling without intermediate storage were provided by Gupta (1976), van der Veen and van Dal (1991) and Szwarc (1981). Reddi and Ramamoorthy (1972) and Wismer (1972) studied no-wait flow shop scheduling problem as an asymmetric travelling salesman problem.

The review given by Framinan et al. (2004) describes a general framework in which the development of heuristics should be implemented and the categories in which existing heuristics can be fitted. The general framework for development of heuristics must possess three phases known as index development, solution construction and solution improvement and can use more than one phase for the development. Also, the order of the development should be in the manner above stated. Based on these phases we develop the improvement heuristic algorithm. For index development phase, the arrangement of jobs is executed in the reverse order of the NEH algorithm. For the construction phase, we propose the steps given in section 3. Finally the solution is improved using a heuristic technique. In the present paper, we present a constructive and an efficient improvement heuristic for solving *n*-job, *m*-machine flow shop scheduling problem without interruptions in processing of jobs with criteria of minimizing makespan. The problem is to schedule *n*-jobs on *m*-machines under no-wait constraint found in the manufacturing industries. The remaining composition of the paper is as follows: Section 2 defines the problem with assumptions; Section 3 presents the proposed heuristic; Section 4 explains the proposed algorithms with the help of numerical illustration; the comparative results are presented in Section 5 and the conclusion is drawn in Section 6.

2. Problem Formulation

Let some job $i\,(1 \le i \le n)$ is to be scheduled on machine $j(1 \le j \le m)$ in the same technological order with criteria to be optimized as minimization of makespan $C^*_{\max}$ under no-wait. Let $t_{i,j}$ be the time of processing of the job i on the machine j, T_i be the sum total of processing times corresponding of job i on m machines, $D(p,q)$ be the minimum delay of the initiation of job p on the first machine after the job p is completed under no-wait constraint and can be calculated by Reddi and Ramamoorthy (1972) formula as

$$D(p,q) = \max\left(t_{p,2} - t_{q,1}, t_{p,2} + t_{p,3} - (t_{q,1} + t_{q,2}), \cdots, t_{p,2} + t_{p,2} + t_{p,4} + \ldots\ldots + t_{p,m} - (t_{q,1} + t_{q,2} + \ldots\ldots + t_{q,m-1}), 0\right)$$

$$= \max_k \left(\sum_{j=2}^{k} t_{p,j} - \sum_{j=1}^{k-1} t_{q,j}, 0\right), 2 \le k \le m$$

The calculation for $C^*_{\max}$ are as follows:

For i= 1,2,3,….,n and j= 1,2,3,……,m

$$C^*_{1,1} = t_{1,1}$$

$$C^*_{1,j} = t_{1,(j-1)} + t_{1,j}, j = 2,3,4,.....,m$$

$$C^*_{i,1} = t_{(i-1),1} + t_{i,1} + D(i-1,i), i = 2,3,4,.....,n$$

$$C^*_{i,j} = \max(C^*_{i,(j-1)}, C^*_{(i-1),j}) + t_{i,j}$$

and the makespan under no-wait is $C^*_{\max} = C^*_{n,m}$

The proposed algorithm has the following assumptions:

1. All jobs and machines are at one's disposal at the start of the processing.
2. Jobs pre-emption is not permitted.
3. The machines are available throughout the processing and never breakdown.
4. All processing times of the machines are deterministic and well known.
5. Each job is processed through each of the machine exactly once.
6. Each machine can perform only one task at a time.
7. A job is not available to the next machine until and unless processing on the current machine is completed.
8. The processing time of jobs include the setup times on machines or otherwise can be ignored.

3. Proposed algorithm

The proposed constructive and the improvement heuristic algorithm follows the generation framework for the development of heuristics. The constructive heuristic (PCH) follows phase I and improvement heuristic (PIH) is improved form of PCH framed by heuristic technique based on the phase II. The various steps involved in the development of the proposed algorithms are explained as follows:

Phase I

Step 1: Find the sum total of the processing time T_i of every job i (i=1,2,3,…,n) on the given m-machines by the expression: $T_i = \sum_{j=1}^{m} t_{i,j}$.

Step 2: Exhibit the job list according to the ascending values of T_i so obtained in step 1. In case of the tie, the sequence which is listed first according to the smaller index is taken for further calculations.

Step 3: Take the first two jobs from the job lists. Find the best possible (having minimum $C^*_{\max}$) two-job partial sequence by arranging them in all possible ways and select it as the current partial sequence.

Step 4: Take the next job from the job list and insert in all possible positions of the partial sequence obtained in step 3. Find the partial sequence with minimum $C^*_{\max}$. This is the current sequence for further construction of final sequence of jobs. If this job happens to be the second last job then the step 4 is skipped and move to step 5.

Step 5: Consider the next two jobs from the unscheduled job list. Find the best possible two-job partial sequence (known as block) from these i.e. with minimum $C^*_{\max}$. Generate all the sequences by inserting the two-job partial sequence at all possible locations of the partial sequence so obtained in step 4. Select the sequence with minimum makespan as the current sequence.

Step 6: Next the first job of the latest block of jobs is inserted at all the possible locations of the current sequence to generate the possible sequences. If any of these sequences has better result, then record that sequence as the current sequence. Further this step is repeated for second job of the latest block.

Step 7: Repeat the step 4 and 5 alternatively for the next jobs present in the job list, otherwise stop. The steps are repeated until all the jobs are scheduled. The sequence obtained is the best sequence with minimum $C^*_{\max}$.

Phase II

Step 1: Note the time of processing of the last job on the last machine in the sequence obtained in the step 7 of phase I and denote this time of processing as α . List the jobs having time of processing more than α on the last machine.

Step 2: If no job exists corresponding to step 1 of phase II, then the sequence obtained in the step 1 of phase II is the final best sequence with minimum makespan.

Step 3: If the jobs corresponding to step 1 of phase II exists, then list these jobs. Pick the first job from this list and insert at all the possible locations of the sequence obtained in step 1. Note the improvement in the value of the makespan with these insertions of jobs. Update the sequence as the final best sequence having minimum makespan, otherwise retain the sequence obtained in the step 7 of phase I as final best sequence.

4. Numerical Illustration

Consider a 5-job, 3-machine flow shop instance. The processing time of all five jobs on three machines are given in the Table 1.

Table 1
Flow shop scheduling instance

Jobs	Machines		
	Machine 1	Machine 2	Machine 3
1	3	2	4
2	4	5	3
3	1	4	5
4	1	3	2
5	4	3	7

According to step 1, find the sum total of every job on all the machines asT_1=9, T_2=12, T_3=10, T_4=6, T_5=14. Arranging the jobs in non-decreasing order of the values of T_i (i=1,2,3,4,5), we get the order of jobs in job list as {4,1,3,2,5}. Take the first two jobs namely {4, 1} from the job list. Calculate the value of $C^*_{\max}$ for two possible arrangement produced from {4, 1} namely 4-1 and 1-4. The corresponding values of $C^*_{\max}$ for the partial sequence 4-1 and 1-4 are 10 and 11, respectively. Therefore, the partial sequence 4-1 is picked for further treatment and is considered as the best current partial sequence. Now, picking the next job 3 from the job list and inserting it at all the possible locations of the partial sequence 4-1 generates the partial sequences 3-4-1, 4-3-1 and 4-1-3 with $C^*_{\max}$ = 16, 17 and 15, respectively. The partial sequence 4-1-3 with minimum $C^*_{\max}$ is taken as the best current partial sequence. Further, pick the next two jobs from the job list namely {2, 5}. The partial sequence 5-2 having minimum $C^*_{\max}$ from this pair is inserted as block at all the possible locations of the current partial sequence obtained in the previous step generating the sequences 5-2-4-1-3, 4-5-2-1-3, 4-1-5-2-3 and 4-1-3-5-2 with $C^*_{\max}$ = 28, 27, 27and 25, respectively. Thus, the current best sequence becomes 4-1-3-5-2. Now, inserting the first job i.e. job 5 in the last block of jobs at all the possible locations of the best current sequence 4-1-3-5-2 generates the sequences 5-4-1-3-2, 4-5-1-3-2, 4-1-5-3-2, 4-1-3-5-2 and 4-1-3-2-5 with $C^*_{\max}$ = 28, 27, 26, 25 and 25, respectively. Since no improvement is made therefore the sequence 4-1-3-5-2 retains itself as the best current sequence. Further the second job 2 of the last pair is selected for inserting at all the possible locations of the best current sequence 4-1-3-5-2 generating the sequences 2-4-1-3-5, 4-2-1-3-5, 4-1-2-3-5, 4-1-3-2-5 and 4-1-3-5-2 with $C^*_{\max}$ = 30, 29, 29, 25 and 25, respectively. The sequence 4-1-3-5-2 retains itself as the best current sequence. Note the time of processing of the last job on the last machine in the 4-1-3-5-2 i.e. time of processing of job 2 on machine 3. Here α = 3 as per step 1 of phase II. The jobs having time of processing more than α on the last machine are {1, 3, 5}. Performing step 3 of phase II, we get the final best sequence as 4-1-3-5-2.

5. Computational Results

The performance evaluation of the proposed algorithms is tested against the NEH under no-wait and the MNEH proposed by Ding et al. (2015) on the Taillard's instances. The instances of Taillard (1993) is a set of 120 problems including 10 instances for each pair of (n,m)= {(20,5), (20,10), (20,20), (50,5), (50,10), (50,20), (100,5), (100,10), (100,20), (200,10), (200,20), (500,20)} (available at

http://people.brunel.ac.uk/_mastjjb/jeb/orlib/files/flowshop2.txt

or

http://www.lifl.fr/_liefooga/benchmarks/benchmarks/index.html.

The proposed algorithms are implemented in MATLAB-R2008a and are made to run on i-3 processor.

Table 2
Makespan values on Taillard's instances

Problem Description		Makespan		Problem Description		Makespan	
Taillard's Instance	Upper Bound	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)	Problem Instance	Upper Bound	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)
20×5				50×10			
1	1486	1558	1532	1	4274	4495	4446
2	1528	1596	1577	2	4177	4409	4345
3	1460	1529	1503	3	4099	4282	4197
4	1588	1593	1590	4	4399	4592	4562
5	1449	1488	1473	5	4322	4492	4432
6	1481	1494	1485	6	4289	4488	4456
7	1483	1523	1520	7	4420	4676	4637
8	1482	1539	1510	8	4318	4559	4500
9	1469	1501	1501	9	4155	4366	4340
10	1377	1459	1416	10	4283	4439	4428
Average	1480.3	1528	1510.7	Average	4273.6	4479.8	4434.3
20×10				50×20			
1	2044	2090	2061	1	6129	6321	6287
2	2166	2256	2184	2	5725	5953	5873
3	1940	2019	1980	3	5862	6236	6204
4	1811	1879	1868	4	5788	6052	6041
5	1933	2038	1985	5	5886	6257	6087
6	1892	1951	1941	6	5863	6179	6161
7	1963	2035	2035	7	5962	6211	6124
8	2057	2218	2204	8	5926	6294	6167
9	1973	2103	2010	9	5876	6169	6113
10	2051	2139	2139	10	5958	6249	6160
Average	1983	2072.8	2040.7	Average	5897.5	6192.1	6121.7
20×20				100×5			
1	2973	3035	3035	1	6397	6949	6863
2	2852	2955	2898	2	6234	6715	6604
3	3013	3080	3040	3	6121	6542	6450
4	3001	3142	3102	4	6026	6396	6315
5	3003	3036	3024	5	6200	6580	6525
6	2998	3056	3056	6	6074	6578	6413
7	3052	3157	3126	7	6247	6784	6630
8	2839	2985	2985	8	6130	6590	6487
9	3009	3012	3012	9	6370	6854	6740
10	2979	3093	3050	10	6381	6846	6725
Average	2971.9	3055.1	3032.8	Average	6223.5	6683.4	6575.2
50×5				100×10			
1	3161	3417	3338	1	8077	8588	8374
2	3432	3646	3637	2	7880	8291	8214
3	3211	3447	3359	3	8028	8488	8339
4	3339	3544	3517	4	8348	8796	8721
5	3356	3612	3542	5	7958	8466	8395
6	3347	3498	3452	6	7801	8291	8099
7	3231	3402	3327	7	7866	8513	8361
8	3235	3497	3418	8	7913	8486	8348
9	3072	3211	3200	9	8161	8622	8504
10	3317	3526	3493	10	8114	8497	8356
Average	3270.1	3480	3428.3	Average	8017.5	8503.8	8371.1

Table 1 and Table 2 describe the problem description, the value of the makespan obtained from the proposed PCH and PIH algorithms along with the best known solutions (upper bounds). The performance of the proposed heuristic algorithms with all other heuristic algorithms discussed is calculated by Relative Percentage Deviation calculated as:

$$\text{Relative Percentage Deviation (RPD)} = \frac{Avg.Makespan_{heuristic} - Avg.Makespan_{best}}{Avg.Makespan_{best}} \times 100,$$

where, $Avg.Makespan_{heuristic}$ is the value of the average makespan obtained by the heuristic for a particular set of problems and $Avg.Makespan_{best}$ is the value of the average best known makespan (upper bound).

Table 2
Makespan values on Taillard's instances

Problem Description		Makespan		Problem Description		Makespan	
Taillard's Instance	Upper Bound	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)	Problem Instance	Upper Bound	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)
100×20				200×20			
1	10700	11396	11287	1	19681	21051	20855
2	10594	11308	11035	2	20096	21188	20932
3	10611	11247	11049	3	19913	21167	20825
4	10607	11285	11061	4	19928	21128	20848
5	10539	11080	10973	5	19843	21017	20869
6	10690	11277	11166	6	19942	21348	20997
7	10825	11429	11345	7	20112	21473	21298
8	10839	11492	11301	8	20056	21271	21102
9	10723	11355	11299	9	19918	21055	20657
10	10798	11334	11241	10	19935	21038	20841
Average	10692.4	11320.3	11175.7	Average	19957.7	21173.6	20922.4
200×10				500×20			
1	15319	16542	16207	1	46689	49821	49234
2	15085	16225	16016	2	47275	50604	49736
3	15376	16361	16081	3	46544	49651	48972
4	15200	16149	16050	4	46899	49937	49506
5	15209	16187	15997	5	46741	49862	49403
6	15109	16142	15895	6	46941	50380	49699
7	15395	16523	16395	7	46509	49608	49269
8	15237	16294	16014	8	46873	50071	49666
9	15100	16189	15979	9	46743	49627	49203
10	15340	16308	16118	10	46847	50057	49455
Average	15262.7	16292	16075.2	Average	46806.1	49961.8	49414.3

Table 3
Relative Percentage Deviation on Taillard Instances

Problem Instances	Average Upper Bound	Average Makespan				Relative Percentage Deviation (RPD)			
		NEH	MNEH	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)	NEH	MNEH	Proposed Heuristic (PCH)	Proposed Heuristic (PIH)
20×5	1480.3	1540.5	1543.6	1528	**1510.7**	4.07	4.28	3.22	**2.05**
20×10	1983	2053.9	2052.9	2072.8	**2040.7**	3.58	3.52	4.53	**2.91**
20×20	2971.9	3062.5	3089.5	3055.1	**3023.7**	3.05	3.96	2.8	**1.74**
50×5	3270.1	3520.4	3472.7	3480	**3428.3**	7.65	6.2	6.42	**4.84**
50×10	4273.6	4522	4478.6	4479.8	**4434.3**	5.81	4.8	4.82	**3.76**
50×20	5897.5	6230.4	6156.9	6192.1	**6121.7**	5.64	4.4	5	**3.8**
100×5	6223.5	6719	6674.2	6683.4	**6575.2**	7.96	7.24	7.39	**5.65**
100×10	8017.5	8537.1	8515	8503.8	**8371.1**	6.48	6.21	6.07	**4.41**
100×20	10692.4	11247.2	11247.3	11320.3	**11175.7**	5.19	5.19	5.87	**4.52**
200×10	15262.7	16354.6	16263.8	16292	**16075.2**	7.15	6.56	6.74	**5.32**
200×20	19957.7	21089.7	21028.8	21173.6	**20922.4**	5.67	5.37	6.09	**4.83**
500×20	46806.1	49710.5	49680.4	49961.8	**49414.3**	6.21	6.14	6.74	**5.57**
Overall Average						5.71	5.32	5.47	**4.12**

For the values of NEH under no-wait, MNEH heuristic algorithm and the best known solutions can be referred to Ding et al. (2015) for the Taillard's instances. Out of the 120 Taillard's problem instances the proposed heuristic improves the results of NEH, MNEH and the PCH by 27.85%, MNEH by 22.56% and that of the proposed constructive heuristic algorithm by 24.68% by applying simple heuristic technique with little more computational efforts used by the phase II of the algorithm. The PCH algorithm performs better than the NEH is clear from the Table 3. Also, statistically there is no significant difference between

the PCH algorithm and the MNEH. To support this, we apply the test of significance on the RPDs means of PCH and MNEH. For this, we setup a hypothesis that there is no difference between the two RPD means of PCH and MNEH at 5% level of significance. We assume that the RPD values of both algorithms have been drawn from normal population. For applying the t-test we first prove that the populations have the same variance. For this we apply F-test to both the RPD values of PCH and MNEH algorithm.

$F = \frac{S_1^2}{S_2^2}(S_1^2 > S_2^2) = \frac{34.25}{30.68} = 1.16$ where, S_1^2 corresponds to PCH algorithm and S_2^2 corresponds to MNEH algorithm have usual meanings for *F*-test.

The value of F=1.16 < 2.23 (table value) with degrees of freedom (11, 11) at 5% level of significance. Therefore, we may conclude that the two RPD values of PCH and MNEH algorithm have come from two normal populations having the same variance. Hence, the basic condition of t-test holds and now we apply t-test to RPD values to test the difference between them for the PCH and MNEH algorithm.

Under the null hypothesis stated above, the test statistic is given by

$t = \frac{\overline{x_1} - \overline{x_2}}{S\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$, here $n_1 = n_2 = 12$, $\overline{x_1} = 5.47$, $\overline{x_2} = 5.32$, $S_1^2 = 34.25, S_2^2 = 30.68$ and

$S^2 = \frac{1}{n_1 + n_2 - 2}[(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2] = 32.47$

Therefore, $|t| = 0.133 < 2.07$(table value) with $n_1 + n_2 - 2 = 22$ degrees of freedom at 5% level of significance. Hence, null hypothesis is accepted and it may be concluded that there was no significant difference between the means of the RPDs values of the PCH and MNEH algorithm. It further implies that the PCH algorithm can also be taken as initial solution instead of MNEH algorithm for various metaheuristics.

The RPD of all the heuristics considered are reported in Table 3 for Taillard instances and are plotted against the problem instances in the Fig 1. The minimum RPDs values are marked as bold in the Table 3. From the Table 3 and the Fig1, it is clear that the PIH algorithm gives improved solution than other heuristics algorithms on Taillard's instances.

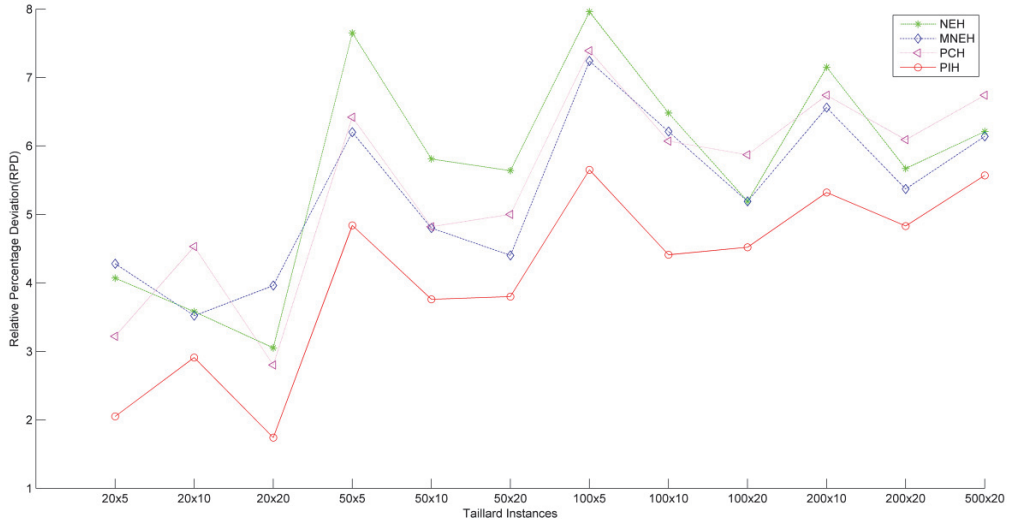

Fig. 1. Plot of RPDs of Heuristics on Taillard Problem Instances

6. Conclusion

The work in this paper presented an alternative constructive heuristic algorithm along with an improvement heuristic algorithm for solving no-wait permutation flow shop scheduling problems with the criteria of minimizing makespan. The PCH algorithm can be the better initial solution than the NEH used by many metaheuristics existing in the literature. Also, the statistical results showed that the PCH algorithm can be an alternative to MNEH as an initial solution to metaheuristics solutions. The improvement heuristic outperforms all heuristics on the Taillard's instances by improving the results of NEH by 27.85%, MNEH by 22.56% and that of PCH algorithm by 24.68%. The average relative percentage deviation being the comparison parameter is calculated from the best known upper bounds found in the literature. Further, the average relative percentage deviation of the PIH algorithm is 4.12% for the 120 Taillard's benchmark instances considered and that of NEH, MNEH and PCH are 5.71%, 5.32 and 5.47%. We tried this comparison on the Taillard instances only but the comparison of these heuristics with more small and large hard instances is required so as to enrich the real scheduling literature.

References

Akhshabi, M., Tavakkoli-Moghaddam, R., & Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *The International Journal of Advanced Manufacturing Technology*, *70*(5), 1181-1188.

Aldowaisan, T., & Allahverdi, A. (1998). Total flowtime in no-wait flow shops with setup times. *Computers and Operations Research*, 25(9), 757-765.

Aldowaisan, T., & Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers & Operations Research*, *30*(8), 1219-1231.

Aldowaisan, T., & Allahverdi, A. (2004). A new heuristic for m-machine no-wait flow shop to minimize total completion time. *Omega*, *32*(5), 345-352.

Allahverdi, A., & Aldowaisan, T. (2000). No-wait and separate setup three-machine flow shop with total completion time criterion. *International Transactions in Operational Research*, *7*(3), 245-264.

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*, New York, John Wiley and Sons .

Bertolissi, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology*,*107*(1-3), 459-465.

Bonney, M. C., & Gundry, S. W. (1976). Solutions to the constrained flow shop sequencing problem. *Operations Research Quarterly*, *27*(4), 869-883.

Chaudhry, I. A., & Khan, A. M. (2012). Minimizing makespan for a no-wait flowshop using genetic algorithm. *Sadhana*, *37*(6), 695-707.

Chen, C., Neppalli, R. V., & Aljaber, N. (1996). Genetic algorithms applied to the continuous flow shop problem. *Computers and Industrial Engineering*, *30*(4), 919-929.

Ding, J.Y., Song, S., Gupta, J.N.D., Rui, Z., & Raymond, C. (2015). An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing*, *30*, 604-613.

Fink, A., & Voß, S., (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, *151*(2), 400-414.

Framinan, J. M., Gupta, J. N., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 1243-1255.

Gangadharan, R., & Rajendran, C. (1993). Heuristic algorithms for scheduling in the no-wait flowshop. *International Journal of Production Economics*,*32*(3), 285-290.

Gao, K., Pan, Q., Suganthan, P. N., & Li, J. (2013). Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. *The International Journal of Advanced Manufacturing Technology*, *66*(9-12), 1563-1572.

Grabowski, J., & Pempera, J.L. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers and Operations Research*, *32*(8), 2197-2212.

Gupta, J.N.D. (1976). Optimal flowshop schedules with no intermediate storage space. *Naval Research Logistics Quarterly*, *23*(2), 235–243.

Hall, N.G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, *44*(3), 510-525.

Johnson, S.M. (1954). Optimal two and three stage production schedule with setup time included. *Naval Research Logistics Quartertly*, *1*, 61-68.

King, J.R., & Spachis, A.S., (1980). Heuristics for flow shop scheduling. *International Journal of Production Research*, *18*(3), 343-357.

Komaki, GH. M., & Kayvanfar, V, (2012). Solving No-wait Flow shop by Heuristic Algorithm. Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management Istanbul, Turkey, July 3 - 6.

Laha, D., & Chakraborty, U.K., (2009). A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *41*(1-2), 97-109.

Laha, D, Sapkal, S.U. (2014). An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop. *Computers & Industrial Engineering*, 67, 36–43.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem.*Computers & Operations Research*, *35*(9), 2807-2839.

Pan, Q.K., Wang, L., & Zhao, B.H. (2008a). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, *The International Journal of Advanced Manufacturing Technology*, *38*(7), 778–786.

Pinedo, M. L. (2002). *Scheduling: theory, Algorithms, and Systems*. Prentice- Hall, NJ: Upper Saddle.

Rajendran, C. (1994). A no-wait flow shop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society*, *45*(4), 472-478.

Rajendran, C., & Chaudhuri, D. (1990). Heuristic algorithms for continuous flow-shop problem. *Naval Research Logistics Quarterly*, *37*(5), 695-705.

Reddi, S.S., & Ramamoorthy, C. V. (1972). On the flow shop sequencing problem with no-wait in process. *Operations Research Quarterly*, *23*, 323-331.

Riyanto, O.A.W., Santosa, B. (2015). *ACO-LS Algorithm for Solving No-wait Flow Shop Scheduling Problem*, in Intan, R., Chi, C-H, Palit, H.N., & Santoso, L.W. (Eds.), Intelligence in the Era of Big Data, Vol. 516 of the series Communications in Computer and Information Science, Springer Berlin Heidelberg, , pp. 89-97.

Rock, H. (1980). The three-machine no-wait flowshop problem is NP-complete. *Journal of Association for Computing Machinery*, *31*(2), 336-345.

Szwarc, W. (1981). A note on the flow-shop problem without interruptions in job processing. *Naval Research Logistics Quarterly*, *28*(4), 665-669.

Szwarc, W. (1983). Solvable cases of the flow-shop problem without interruptions in job processing. *Naval Research Logistics Quarterly*, *30*(1), 179-183.

Tseng, L.Y., & Lin, Y.T. (2010). A hybrid genetic algorithm for no-wait flowshop scheduling problem, *International Journal of Production Economics*, *128* (1), 144–152.

Van der Veen, J.A.A., & Van Dal, R., (1991). Solvable cases of the no-wait flow-shop scheduling problem. *Journal of the Operational Research Society*, *42*(11), 971–980.

Van Deman, J.M., & Baker, K. R., (1974). Minimizing mean flowtime in the flowshop with no intermediate queues. *AIIE Transactions*, *6*(1), 28-34.

Wismer, D. A., (1972). Solution of the flow shop-scheduling with no intermediate queues. *Operations Research*, *20*(3), 689-697.