Ingeniería e Investigación
ISSN: 0120-5609
Facultad de Ingeniería, Universidad Nacional de Colombia.

Chaudhry, Imran Ali; Elbadawi, Isam A-Q.; Usman, Muhammad; Tajammal Chugtai, Muhammad
Minimising Total Flowtime in a No-Wait Flow Shop (NWFS) using Genetic Algorithms
Ingeniería e Investigación, vol. 38, no. 3, 2018, September-December, pp. 68-79
Facultad de Ingeniería, Universidad Nacional de Colombia.

DOI: https://doi.org/10.15446/ing.investig.v38n3.75281

DOI: http://dx.doi.org/10.15446/ing.investig.v38n3.75281

Minimising Total Flowtime in a No-Wait Flow Shop (NWFS) using Genetic Algorithms

Minimizar el tiempo de flujo total en un Flow shop sin escalas (NWFS) utilizando algoritmos genéticos

Imran Ali Chaudhry[1], Isam A-Q. Elbadawi[2], Muhammad Usman[3], Muhammad Tajammal Chugtai[4]

ABSTRACT

This paper considers a no-wait flow shop scheduling (NWFS) problem, where the objective is to minimise the total flowtime. We propose a genetic algorithm (GA) that is implemented in a spreadsheet environment. The GA functions as an add-in in the spreadsheet. It is demonstrated that with proposed approach any criteria can be optimised without modifying the GA routine or spreadsheet model. Furthermore, the proposed method for solving this class of problem is general purpose, as it can be easily customised by adding or removing jobs and machines. Several benchmark problems already published in the literature are used to demonstrate the problem-solving capability of the proposed approach. Benchmark problems set ranges from small (7-jobs, 7 machines) to large (100-jobs, 10-machines). The performance of the GA is compared with different meta-heuristic techniques used in earlier literature. Experimental analysis demonstrate that solutions obtained in this research offer equal quality as compared to algorithms already developed for NWFS problems.

Keywords: Genetic algorithm (GA), Scheduling, No-wait, Flow shop.

RESUMEN

Este documento considera un problema de secuenciación de líneas de flujo sin espera (NWFS), donde el objetivo es minimizar el tiempo de flujo total. Proponemos un algoritmo genético (GA) que se implementa en un entorno de hoja de cálculo. El GA funciona como un complemento en la hoja de cálculo. Se demuestra que, con el enfoque propuesto, cualquier criterio puede optimizarse sin modificar la rutina del GA o el modelo de hoja de cálculo. Además, el método propuesto para resolver este problema de clase es de propósito general, ya que se puede personalizar fácilmente agregando o eliminando tareas y máquinas. Varios problemas de referencia ya publicados en la literatura se usan para demostrar la capacidad de resolución de problemas del enfoque propuesto. El conjunto de problemas de la evaluación tiene un rango que varía desde pequeños (7 trabajos, 7 máquinas) hasta grandes (100 trabajos, 10 máquinas). El rendimiento del GA se compara con diferentes técnicas meta-heurísticas utilizadas en la literatura anterior. El análisis experimental demuestra que las soluciones obtenidas en esta nueva búsqueda ofrecen igual calidad que los algoritmos ya desarrollados para el problema NWFS.

Palabras clave: Algoritmo genético (AG), Secuenciación, Líneas de flujo sin espera, Flow shop.

Received: June 8th 2018
Accepted: December 3rd 2018

Introduction

Scheduling is an important aspect of any manufacturing concern. The importance of efficient scheduling function cannot be denied as it ensures timely dispatch of products to the market before the competitors, thus yielding higher profits. The primary objective in any scheduling problem is to efficiently allocate jobs to the available machines and to determine the start and ending time of each operation, such that certain objective function is minimised or maximised. The schedule developed should also satisfy various production constraints. In order to achieve high-efficiency production, efficient scheduling algorithms/schemes are therefore considered to be a key factor.

Flow shop scheduling is one of the widely studied models of the manufacturing environment. In a general flow shop

[1] Affiliation: Professor, Department of Industrial Engineering, College of Engineering, University of Hail, Ha'il, Saudi Arabia.
E-mail: imran_chaudhry@yahoo.com.

[2] Affiliation: Associate Professor, Department of Industrial Engineering, College of Engineering, University of Hail, Ha'il, Saudi Arabia.
E-mail: isam149@gmail.com.

[3] Affiliation: Associate Professor, Department of Electrical Engineering, College of Engineering, University of Hail, Ha'il, Saudi Arabia.
E-mail: m.usman@uoh.edu.sa.

[4] Affiliation: Assistant Professor, Department of Electrical Engineering, College of Engineering, University of Hail, Ha'il, Saudi Arabia.
E-mail: mt.chughtai@uoh.edu.sa.

How to cite: Chaudhry, I. A., Elbadawi, I. A-Q., Usman, M., and Chughtai, M. T. (2018). Minimising Total Flowtime in a No-Wait Flow Shop (NWFS) using Genetic Algorithms. *Ingeniería e Investigación*, 38(3), 68-79. DOI: 10.15446/ing.investig.v38n3.75281

scheduling problem, there are *n*-jobs that are required to be scheduled on *m*-machines to typically minimise total completion time or makespan. All jobs follow the same processing order. The flow shop scheduling problem has received considerable attention since its introduction in 1954 (Johnson, 1954). Over the years, numerous efficient techniques and meta-heuristics have been proposed by various researchers. Gupta *et al*. (2006) has given a detailed survey of flow shop scheduling research. Tyagi *et al*. (2013) also present a survey of the evolution of flow shop scheduling problems and possible approaches for their solution.

No-wait flow shop (NWFS) is an extension of general flow shop, where all the operations of a particular job are required to be processed in a continuous manner, i.e. there are no intermediate buffers between the machines and all operations are to be processed without interruptions. Pharmaceutical processing, concrete ware production, oil refineries, etc., are some examples of no-wait flow shop scheduling. A comprehensive analysis of research and applications of NWFS has been made by Hall *et al*. (1996). The problem is categorised to be NP-hard even for a simple 3-machine case (Hans, 1984).

In this paper, a NWFS scheduling problem is presented, where the objective is to minimise total flowtime of all jobs. A spreadsheet-based genetic algorithm (GA) is proposed for the problem. Empirical analysis has been made for flow shop benchmark problems proposed by Carlier (1978), Reeves (1995), Heller (1960) and Taillard (1993). The performance of the proposed GA is compared with different meta-heuristics that have been reported earlier in the published literature. The rest of this paper is organised as follows: Section 2 gives an overview of past research for minimisation of total flowtime in NWFS scheduling environment. Section 3 gives problem definition and assumptions. Brief overview of GA and its components is given in section 4. Section 5 presents implementation details of Reddi *et al*. (1972) equation for no-wait flow shop model with a numerical example. Section 6 presents empirical analysis for various benchmark problems taken from already published literature. Finally, section 7 concludes the paper.

Past Research

The first reported instance to address no-wait scenario in flow shop scheduling was presented by Reddi *et al*. (1972). The authors converted the corresponding problem into a travelling salesman problem and solved it in polynomial time by using an algorithm proposed by Gilmore *et al*. (1964). Due to the large number of research papers available on no-wait flow shop scheduling, we will restrict the literature review to the papers addressing only the objective function of flowtime that were published from year 2011 onwards.

Gao *et al*. (2011a) minimise total flowtime in NWFS problem using a discrete harmony search algorithm (DHS). In the first step, job permutation is represented by a harmony. Harmony memory is then initialised by using a new heuristic based on the NEH heuristic method (Nawaz *et al*. (1983). In the second step, novel pitch adjustment rule is employed in the improvisation to produce a new harmony. The local exploitation ability of the algorithm is enhanced by embedding a local search procedure. Laha *et al*. (2011) also minimise total flowtime by a constructive heuristic. The priority of a job in a sequence is determined by the sum of its processing times on the bottleneck machine(s). Computational experiments show that the proposed heuristic performs significantly well compared to Bertolissi heuristic (Bertolissi (2000)). Shafaei *et al*. (2011) minimise mean flowtime in a two-stage flexible no-wait flow shop problem. The authors develop six meta-heuristic algorithms based on imperialist competitive algorithm (ICA), ant colony optimisation (ACO) and particle swarm optimisation (PSO) to solve the problem. Then, they use 36 different problems (18 small and 18 large-scale problems) to test the performance of the algorithms. The results of the numerical experiments show that the proposed algorithms significantly outperform other algorithms in terms of solution quality and CPU time.

Gao *et al*. (2012) also consider minimisation of total flowtime in a NWFS scheduling problem using a hybrid harmony search (HHS) algorithm. NEH heuristic (Nawaz *et al*. (1983) is firstly used to form an initial harmony memory. Secondly, this memory is divided into several small groups, where each group independently executes its evolution process. However, all groups share information reciprocally by dynamic re-grouping mechanism. Thirdly, a variable neighbourhood search algorithm (VNS) is embedded in the HHS algorithm to stress the balance between global and local exploration. A speed-up method is applied to reduce the running time requirement. Computational simulations are carried out on well-known benchmark problems. The results show that the proposed HHS outperforms other methods published in the literature.

Guang *et al*. (2012) consider multi-objective NWFS problem using an evolved discrete harmony search algorithm to minimise total makespan, maximum tardiness and total flowtime. A job-permutation-based encoding scheme is applied to enable the continuous harmony search algorithm to be used for all sequencing problems. An archive set of non-dominated solutions is dynamically updated during the search process. The authors demonstrate that the proposed algorithm produces superior quality solutions in terms of searching diversity level, efficiency and quality. Tasgetiren *et al*. (2013) also consider a multi-objective NWFS problem to minimise the makespan and total flowtime. A variable iterated greedy algorithm with differential evolution is proposed to solve the problem. A differential evolution algorithm is used to optimise the parameters of the iterated greedy algorithm. Gao *et al*. (2013) present four composite and two constructive heuristics to minimise the flowtime.

The heuristics are based on constructive heuristic proposed by Laha *et al*. (2008), Bertolliso heuristic (Bertolissi, 2000) and standard deviation heuristic (Gao *et al.*, 2011a). The performance of the proposed heuristics is tested on benchmark flow shop problems already published in the literature. Experimental results show that the proposed heuristics perform better than the existing ones.

Sapkal *et al*. (2013) propose a constructive heuristic to minimise flowtime. In the initial sequence, the sum of processing times of individual jobs on the bottleneck machines are used to prioritise the jobs. Final job sequence is obtained by a new job insertion technique based on NEH heuristic (Nawaz *et al.*, 1983)1983. The authors demonstrate that the proposed heuristic outperforms Rajendran *et al*. (1990) and Bertolissi (2000) heuristics without effecting the average computational time. Akhshabi *et al*. (2014) propose a hybrid algorithm based on particle swarm optimisation (PSO) and a local search method to minimise total flowtime. Laha *et al*. (2014a) minimise total flowtime by a penalty-shift-insertion algorithm. A penalty-based heuristic derived from Vogel's approximation method for classic transportation problem is used to generate the initial sequence. In the second phase, a forward shift heuristic is used to improve the solution. The solution is further improved by a job-pair and a single-job insertion heuristic. Laha *et al*. (2014b) also propose a constructive heuristic to minimise flowtime in a NWFS scheduling problem. Similarly, Chaudhry *et al*. (2014) also present a GA approach to minimise total flowtime in no-wait flow shop scheduling problem. The performance of the GA is compared with well-known benchmark problems.

Zhu *et al*. (2015) also propose an iterative search method to minimise flowtime. Huang *et al*. (2015) propose a new heuristic algorithm named "Ant colony optimization (ACO) with flexible update". The proposed heuristic overcomes the limitations of traditional ACO algorithm. Nagano *et al*. (2015) consider minimisation of flowtime in a NWFS with sequence dependent setup times. A constructive heuristic is proposed to minimise flowtime by breaking the problem into quarters. The performance of the proposed algorithm is compared with previously reported heuristic algorithms. Qi *et al*. (2016) also consider minimisation of flowtime by a fast-local neighbourhood search algorithm. The algorithm initially constructs an unscheduled job sequence according to the total processing time and standard deviation of jobs on the machines. In the first step, the job sequence is optimised using a basic neighbourhood search algorithm. Then, an innovative local neighbourhood search scheme is designed to search for the partial neighbourhood in each iterative processing and calculate its solution with an objective increment method. The experimental results show that the proposed approach performs better than previous approaches in terms of quality and robustness of the solution.

Ying *et al*. (2016) propose a self-adaptive ruin-and-recreate algorithm to minimise flowtime in a no-wait flow shop scenario. Bewoor *et al*. (2017a) present a hybrid PSO algorithm to solve this class of problem. The proposed algorithm initialises population efficiently with the NEH heuristic technique (Nawaz *et al.*, 1983)1983 and uses an evolutionary search guided by PSO, as well as simulated annealing based on a local neighbourhood search to avoid getting stuck in local optima and to provide the appropriate balance of global exploration and local exploitation. Bewoor *et al*. (2017b) present a PSO algorithm to minimise flowtime in a no-wait flow shop problem. The authors show that the proposed PSO algorithm outperforms GA and Tabu Search (TS) algorithms. Bewoor *et al*. (2018) also present a hybrid PSO algorithm for minimisation of flowtime in a foundry. Extensive computational experiments are carried out based on various casting (job) characteristics viz. casting type, mould size and type of alloy, where size of job (n) is considered as 10, 12, 20, 50 and 100. Miyata *et al*. (2018) study the impact of preventive maintenance policies in the performance of constructive heuristics for the no-wait flow shop problem with total flowtime minimisation. Díaz Ramírez *et al*. (2018) apply a mixed integer programming for production-scheduling in a chemical industry that identifies lot size and product sequence to maximise profit.

The proposed GA presented here is an extension of earlier work (Chaudhry *et al.*, 2014; Chaudhry *et al.*, 2012). In the current research, we present a spreadsheet-based GA for a NWFS scheduling environment, where the objective is to minimise total flowtime. As compared to previous studies, the proposed approach is general purpose and domain independent whereby it can be used for the optimisation of any objective function without changing the spreadsheet model or the GA routine. Similarly, the spreadsheet model can be extended to cater for more machines and jobs without any change to the basic GA routine. Spreadsheets have been used extensively for scheduling, as highlighted by Astaiza A (2005) for examination scheduling.

Problem Description and Assumptions

The general no-wait flow shop scheduling can be described as follows: there are n jobs from a set of jobs {j = 1, 2, 3, 4…, n} that are required to be processed through m machines {k = 1, 2, 3, 4…, m}. Each job j has a sequence of m operations (o_{j1}, o_{j2}…, o_{jk}) that are required to be processed through m machines in a continuous manner, such that the completion time of o_{jk} is equal to the earliest start time of $o_{j,k+1}$ for k = 1, 2, 3…, m-1. In other words, there has to be no waiting time between successive operations of each of the n jobs. The problem is then to find the sequence of jobs that would minimise the total flowtime of all the jobs.

The flowtime criterion for a schedule provides the measure of the time that a job spends in the system. The total flowtime for a sequence of jobs is the sum of the completion times of all the jobs. Minimisation of total flowtime criterion leads to rapid turn-around of jobs, stable utilisation of resources, and minimisation of work-in-process inventory

costs (Framinan *et al.*, 2003)2003. The total flowtime is thus given by:

$$\sum_{j=1}^{n} F_j = \sum_{j=1}^{n} C_{mjr}, \quad (1)$$

where is the completion time of job j_n on machine k_m, if it is scheduled in *r* position.

Other assumptions in this study are as follows:

a. All jobs are available at $t = 0$.

b. Processing times of operations are known in advance and deterministic.

c. An operation once started cannot be disrupted, i.e. no pre-emption of operations and jobs.

d. A machine, at any time, can process at most one job only.

e. At any given time, each job can be processed on only one machine.

f. There are no setup times for preparing a machine to process an operation.

g. Time for the movement of jobs between machines is negligible.

Genetic Algorithms

Genetic Algorithms (GA) belong to population-based meta-heuristics that are based on Darwin's theory of natural evolution. GAs were first proposed by Holland (1975) and his colleagues at the University of Michigan. These are general algorithms that work well in variety of situations. They are quickly able to provide a reasonable solution to the problem as they can traverse through large search spaces fast. GAs are most effective in a search space for which little is known. The first reported application of GAs for scheduling was presented by Davis (1985). Delgado *et al.* (2005) have also applied genetic algorithms for scheduling manufacturing cell tasks. Similarly, Frutos *et al.* (2012) apply genetic algorithms for multi-objective scheduling procedures in non-standardised production processes.

GAs start with a population of solutions (prospective solutions called chromosomes). Solutions from one population are taken into the next population with a view of getting better solutions in successive generations. In the first step, based on the fitness, two parent solutions are selected to form a child solution by employing the crossover operator. Afterwards, crossover mutation is applied to make random changes in the solution and form newer solutions. The algorithm then compares the fitness of the child solutions with the rest of the members of the population thus using the principle of survival of the fittest to discard the worst performing member of the population.

In this research, we have used permutation representation for the chromosome. For parent selection, rank-based selection method is used, while steady-state reproduction is used to produce offspring for the next generation (Whitley *et al.*, 1988). For crossover operation, an order crossover (Davis, 1985) is used as it works best with the permutation representation by preserving the relative order of the genes and avoids duplicate genes in the chromosome. In the mutation operation, individual genes are swapped to form new chromosomes. The number of swaps increases or decreases corresponding to increase or decrease in the mutation rate. The details about various GA components, i.e., selection, reproduction, crossover and mutation, are given in Chaudhry *et al.* (2017). The flowchart of the GA as implemented in this research is shown in Figure 1.

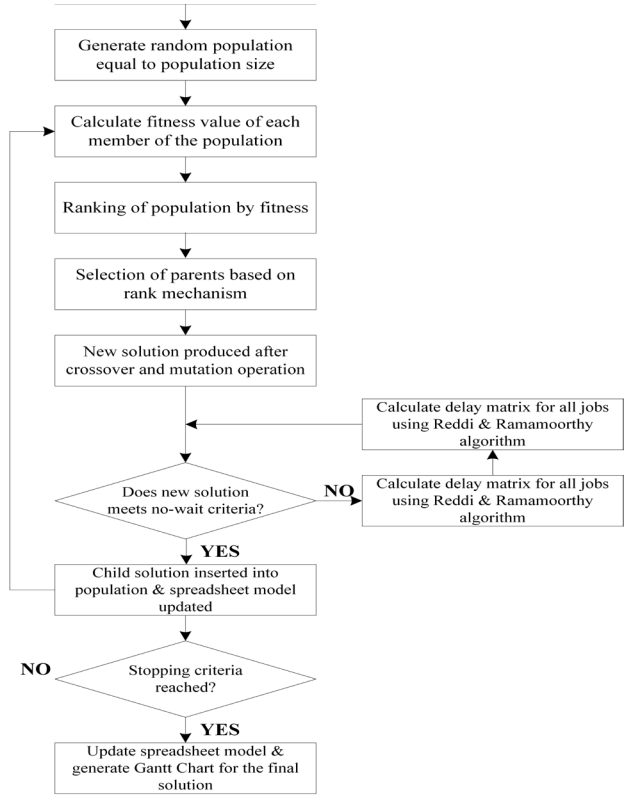

Figure 1. GA implementation flow chart.
Source: Authors

Implementation Details

As stated earlier, the no-wait scheduling model in this research is based on the start delay matrix proposed by Reddi *et al.* (1972). This section describes the implementation details of this matrix.

Consider a 5-job, 4-machine flow shop problem, as given in Table 1, where the objective is to minimise total flowtime. The optimal job sequence to minimise flowtime in the given problem is 4-1-5-2-3. The Gantt chart for the problem with waiting times between successive operations of the jobs is given in Figure 2. It can be seen from Figure 2 that Job 1 waits for 1, 4 and 3 time units between operation 1-2, 2-3 and 3-4, respectively. Waiting times between operation 1-2, 2-3 and 3-4 of Job 5 are 3, 5 and 5 time units, respectively. For Job 2, the waiting times are 8, 9 and 8 time units between operations 1-2, 2-3 and 3-4, respectively, whereas Job 3 waits for 10, 12 and 13 time units between operation 1-2, 2-3 and 3-4, respectively. As per no-wait constraint, all operations are required to be processed in continuation, i.e. there should not be any waiting time between successive operations of a particular job.

Table 1. Job data for the example problem

	Process time on			
Job	**M1**	**M2**	**M3**	**M4**
1	2	4	8	10
2	3	4	7	11
3	2	6	9	12
4	1	5	9	13
5	3	6	8	14

Source: Authors

In this research work, we use a two-step procedure for the NWFS scheduling problem. The first stage calculates the delay factor for each job sequence. The start of the job is then delayed by as many time units as have been calculated in stage 1. Reddi *et al.* (1972) equation is used to calculate the delay factor for job *i* after job *j*.

If *F (i, j)* gives the minimum delay between the completion of job J_i and the start of job J_j, then the delay *F (i, j)* would be calculated by equation 2 (Reddi *et al.*, 1972)1972, as follows:

$$\begin{aligned} F(i,j) &= \max(t_{i2}-t_{j1}, t_{i2}+t_{i3}-(t_{j1}-t_{j2}), \ldots\ldots\ldots, \\ &\quad t_{i2}+t_{i3}+t_{i4}+\ldots\ldots+t_{im}-(t_{j1}+t_{j2}+\ldots\ldots+t_{j(m-1)}), 0) \\ &= \max\left(\sum_{n=2}^{k} t_{in} - \sum_{n=2}^{k=1} t_{jn}, 0\right) \qquad 2 \le k \le m \end{aligned} \tag{2}$$

From equation (2), we can observe that if job J_i proceeds with no-wait in process, then the time to complete job J_i is independent of the jobs that will precede and follow it. The minimum time for starting job J_j after completion of J_i on the first machine, i.e. *F(i, j)*, is the function of the parameters of job J_i and J_j only. Hence, the minimum timing of any sequence (j_1, j_2, j_3, …, j_n) must incorporate times *F(i, j)* between successive pairs of jobs J_i, J_j (Reddi *et al.*, 1972)1972.

The corresponding schedule for the Gantt chart in Figure 2 would be as shown in Figure 3.

Empirical Analysis

Empirical analysis was carried out to compare the performance of the proposed GA with earlier studies. The experiments were carried out on four different sets of benchmark problems taken from already published literature. The experiments were conducted on a Core *i*3 1,8 GHz computer with 4 GB RAM. Being a stochastic optimisation technique, the performance of a GA is dependent on different parameters, namely: crossover & mutation rates and the population size. Repeated tests were therefore conducted to determine the best set of values for aforesaid parameters. The best values were found to be 0,65 and 0,06, and 65 for crossover & mutation rates and the population size, respectively. Each problem was then run for 100 000 iterations that corresponded to 3 mins on the aforementioned computer. The results presented in the subsequent sub-section are based on 30 simulation runs, i.e. each problem instance is run for 30 times with random starting solution and subsequently noting the best value found for each instance. The % Diff is the relative difference of the best value found by all other algorithms (TFT_{min}) against the proposed GA algorithm (TFT_{GA}) and is calculated by equation 3:

$$\frac{TFT_{\min} - TFT_{GA}}{TFT_{\min}} \times 100 \tag{3}$$

Positive values indicate that the proposed GA found better results as compared to all other previous algorithms, while negative values indicate worse results.

Problem Set 1

Problem set 1 consists of eight problem instances adapted from Carlier (1978). The results produced by GA have been compared with the following algorithms:

A-1: Grouping harmony search algorithm (Gao *et al.*, 2011b)

A-2: Discrete differential evolution algorithm (Gao *et al.*, 2011b)

A-3: Improved harmony search algorithm (Gao *et al.*, 2011b)

A-4: Particle swarm optimisation algorithm (Dong *et al.*, 2010)

A-5: Differential evolution algorithm (Dong *et al.*, 2010)

A-6: Hybrid differential evolution algorithm (Dong *et al.*, 2010)

The proposed GA approach found better results for six problems, while same results for two problems. Comparative results for total flowtime values of algorithms A1 – A6 and the proposed GA algorithm are presented in Table 2.

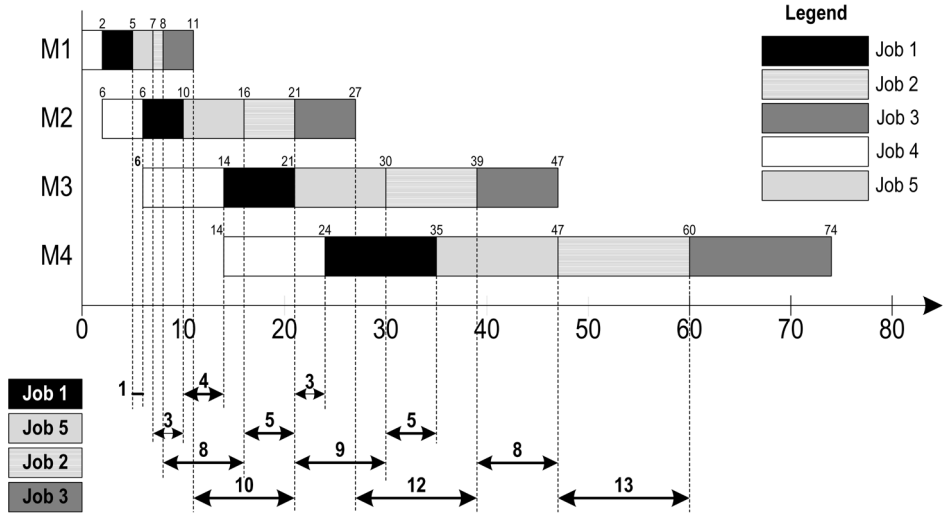

Figure 2. Gantt chart for job sequence 4-1-5-2-3 with waiting times between the jobs.
Source: Authors

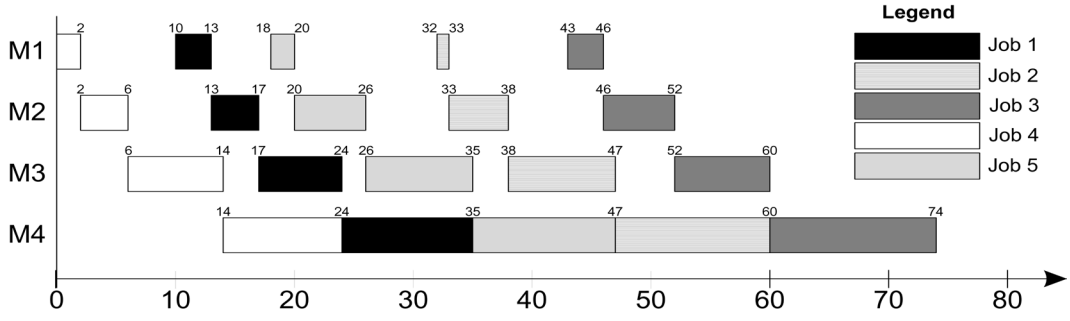

Figure 3. Gantt chart for job sequence 4-1-5-2-3 with no-waiting times between successive operations.
Source: Authors

Table 2. Total flowtime comparison for algorithms A-1 to A-6 for Carlier (1978) data set

Instance	*n* x *m*	A-1	A-2	A-3	A-4	A-5	A-6	Proposed GA	
								Best	% Diff
car1	11 × 5	56 209	53 339	52 641	54 245	55 955	53 951	**52,353**	0,550
car2	13 × 4	65 199	56 833	55 717	61 638	68 768	58 968	**55 541**	0,317
car3	12 × 5	69 157	63 328	62 432	65 508	65 199	62 432	**61 965**	0,754
car4	14 × 4	81 882	81 040	74 565	79 348	79 604	75 716	**74 093**	0,637
car5	10 × 6	61 619	60 497	59 040	60 304	60 497	60 160	**58 445**	1,018
car6	8 × 9	56 004	52 946	52 946	53 470	52 946	52 946	**52 798**	0,280
car7	7 × 7	38 578	36 869	**36 534**	36 534	37 061	**36 534**	**36 534**	0
car8	8 × 8	54 273	52 912	**52 703**	53 175	52 912	**52 703**	**52 703**	0

Source: Authors

Problem Set 2

Problem set 2 consists of twenty-one problem instances adapted from Reeves (1995), ranging from 20-jobs 5-machines to 75-jobs 20-machines. Apart from the algorithms mentioned for Problem Set 1, the performance of the proposed algorithm was also compared with three more algorithms, as mentioned below:

A-7: Harmony search algorithm (Gao *et al.*, 2010)

A-8: Differential evolution algorithm (Gao *et al.*, 2010)

A-9: Grouping harmony search algorithm (Gao *et al.*, 2010)

Comparative results of the proposed approach with nine other algorithms, i.e. from A-1 to A-9, for minimisation of total flowtime are given in Table 3. From Table 3, we can

Table 3. Total flowtime comparison for algorithms A-1 to A-9 for Reeves (1995) data set

Instance	*n* x *m*	A-1	A-2	A-3	A-4	A-5	A-6	A-7	A-8	A-9	Min	Proposed GA		% Diff
												Best	Avg	
rec01	20 x 5	20 029	17 874	17 874	19 556	19 938	17 594	21 063	20 873	20 289	17 594	**17 187**	17 508,30	2,368
rec03	20 x 5	18 163	15 248	15 098	17 417	17 869	16 235	19 615	19 689	18 358	15 098	**14 682**	14 919,60	2,833
rec05	20 x 5	19 034	17 785	17 793	19 210	19 055	17 910	20 554	20 261	19 149	17 785	**17 142**	17 409,50	3,751
rec07	20 x 10	28 914	26 045	25 647	28 407	28 841	24 978	28 914	28 841	26 912	**24 978**	25 105	25 770,53	-0,506
rec09	20 x 10	27 229	24 347	24 347	26 796	29 254	26 234	29 355	29 254	25 965	24 347	**23 861**	24 088,10	2,037
rec11	20 x 10	25 657	23 248	22 706	25 362	25 657	23 324	27 466	27 619	25 510	22 706	**22 218**	22 469,90	2,196
rec13	20 x 15	37 755	34 382	33 136	36 669	35 091	33 279	38 668	38 307	35 091	33 136	**32 524**	33 016,80	1,882
rec15	20 x 15	35 753	34 286	33 066	35 905	35 035	32 451	37 200	38 240	35 035	32 451	**32 218**	32 760,35	0,723
rec17	20 x 15	36 709	31 956	31 901	35 215	35 563	33 178	38 084	37 626	33 847	31 901	**31 528**	31 678,30	1,183
rec19	30 x 10	58 866	52 564	51 080	59 231	62 458	53 609	61 578	62 458	56 667	51 080	**50 395**	50 900,10	1,359
rec21	30 x 10	58 925	50 364	48 935	57 782	60 206	51 234	61 195	60 206	55 279	48 935	**47 733**	48 884,50	2,518
rec23	30 x 10	55 056	51 981	47 921	56 316	57 992	47 901	55 060	57 992	55 056	47 901	**45 935**	47 588,80	4,280
rec25	30 x 15	77 467	70 280	65 926	76 201	78 315	66 566	79 310	78 315	72 610	65 926	**64 805**	65 913,60	1,730
rec27	30 x 15	73 564	65 425	63 788	73 432	74 699	66 679	76 868	74 699	69 739	63 788	**62 792**	63 735,20	1,586
rec29	30 x 15	74 560	59 655	59 655	-	-	-	80 378	79 649	69 178	59 655	**58 221**	59 608,60	2,463
rec31	50 x 10	153 276	120 133	118 184	-	-	-	156 544	160 666	151 279	118 184	**117 368**	121 406,20	0,695
rec33	50 x 10	157 020	131 960	125 914	-	-	-	165 615	166 772	161 474	125 914	**123 601**	128 132,50	1,871
rec35	50 x 10	157 527	125 474	124 035	-	-	-	171 974	177 408	160 466	124 035	**123 667**	127 157,90	0,298
rec37	75 x 20	464 985	355 803	**344 797**	-	-	-	472 305	471 108	466 048	344 797	368 785	378 189,60	-6,505
rec39	75 x 20	486 774	370 643	**356 681**	-	-	-	488 338	487 011	483 443	356 681	378 596	385 217,70	-5,788
rec41	75 x 20	487 457	369 798	**355 808**	-	-	-	498 551	493 196	492 006	355 808	383 363	393 873	-7,188

Source: Authors

see that the proposed approach produced better results for 17 instances out of a total of 21. The proposed approach could not find better results for instance 'rec07', where the percentage error was 0,506%, as compared to the best-known value, i.e. to algorithm A-6. Furthermore, the performance of the proposed approach was also worse for problem size 75 × 20, i.e. instances rec37, rec39 and rec41, where the percentage errors were 6,505%, 5,788% and 7,188%, respectively, compared to the best-known value among algorithms A1 to A-9. Only for algorithms A-2 and A-3, the results were superior to the proposed approach for problem size 75 × 20. For all other problems, the results obtained by the proposed approach were superior to all other nine algorithms (A-1 to A-9). The best values found for each instance by various algorithms is marked in bold.

Problem Set 3

Problem set 3 consists of two problem instances adapted from Heller (1960). The first problem instance is a large sized problem with 100 jobs and 10 machines, while the second instance is a small sized problem with 20 jobs and 10 machines. For Problem Set 3, comparison of the GA was also done with nine algorithms (A-1 to A-9), as mentioned previously. The proposed GA was able to find superior results compared to all nine previous algorithms for problem instance 20 × 10, while the performance was worse only in algorithms A6 and A9 for problem instance 100 × 10. The comparative total flowtime values are presented in Table 4. The best values for each of the two instances are marked in bold.

Table 4. Tomtal flowtime comparison for algorithms A-1 to A-9 for Heller (1960) data set

Instance	hel1	hel2
n x *m*	100 x 10	20 x 10
A-1	54 683	2 466
A-2	54 833	2 476
A-3	54 216	2 384
A-4	54 216	2 459
A-5	39 693	2 236
A-6	**37 285**	2 105
A-7	54 168	2 373
A-8	54 833	2 476
A-9	39 422	2 201
Proposed GA	39 455	**2 070**
% Diff	-5,500	1,691

Source: Authors

Problem Set 4

Problem set 4 consists of sixty problem instances adapted from Taillard (1993). mThe set consists of six subsets of problems with n x m combination of 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10 and 50 × 20. Each set consists of 10 instances. The following heuristics were used for the comparison of results with the proposed GA algorithm:

A-10: Improved std dev heuristic proposed by Gao *et al.* (2011a)

A-11: Job insertion based heuristic algorithm proposed by Bertolissi (2000)

A-12: Constructive heuristic, based on the idea of job insertion, proposed by Laha *et al*. (2008)

A-13: Heuristic algorithms proposed by Aldowaisan *et al*. (2004)

A-14: ISDH algorithm with local search by Gao *et al*. (2013)

A-15: IBH with local search algorithm by Gao *et al*. (2013)

A-16: ISDH algorithm with an iteration operator by Gao *et al*. (2013)

A-17: IBH algorithm with iteration operator by Gao *et al*. (2013)

Table 5 to Table 10 give the comparative results for the total flowtimes found by various algorithms for the flow shop instances proposed by Taillard (1993). The best values among all instances are marked in bold.

From the preceding tables, we can see that the proposed GA approach was able to find better solution for 33 instances out of a total of 60 problem instances solved, while for 27 instances the performance was worse. For smaller size problems, i.e. for n = 20 (total of 30 instances), the proposed GA approach produced superior results for 24 instances, while worse only for six instances. For only two problem instances, i.e. *tail2* and *tail3*, the **%Diff**

Table 5. Total flowtime comparison for algorithms A-10 to A-17 for 20 x 5 data set from (Taillard, 1993)

Instance	A-10	A-11	A-12	A-13	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail1	16 553	16 562	16 421	16 357	16 414	16 230	16 381	16 302	**15 674**	3,4258
tail2	16 749	16 435	16 551	16 268	**16 164**	16 172	16 220	16 230	17 250	-6,7186
tail3	15 160	15 197	14 959	15 258	**14 943**	15 024	15 051	15 018	15 855	-6,1032
tail4	18 989	18 864	19 048	18 644	18 732	18 679	18 788	18 782	**17 970**	3,6151
tail5	17 293	16 587	16 570	16 353	16 684	16 475	16 385	16 467	**15 317**	6,3352
tail6	16 268	15 841	15 974	15 669	16 109	15 832	15 620	15 841	**15 501**	0,7618
tail7	16 302	16 533	16 538	16 116	15 990	15 898	16 117	16 312	**15 693**	1,2895
tail8	17 836	17 509	17 277	17 528	17 403	17 499	17 340	17 421	**15 955**	7,6518
tail9	16 802	17 096	17 186	16 760	16 551	16 736	16 802	16 588	**16 394**	0,9486
tail10	15 693	15 897	15 776	15 688	15 785	**15 051**	15 208	15 373	15 329	-1,8471

Source: Authors

Table 6. Total flowtime comparison for algorithms A-10 to A-17 for 20 x 10 data set from (Taillard, 1993)

Instance	**A-10**	**A-11**	**A-12**	**A-13**	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail11	27 043	25 664	26 431	25 410	26 582	25657	25410	25 664	**25 319**	0,3581
tail12	26 976	27 037	26 794	26 847	26 748	26774	26773	26 586	**26 363**	0,8388
tail13	25 033	24 509	24 856	24 377	24 230	24509	24260	24 277	**22 910**	5,4478
tail14	23 323	23 353	23 284	22 905	22 976	23120	22905	23 138	**22 243**	2,8902
tail15	24 056	24 185	23 824	23 779	23 611	23838	24056	23 998	**23 191**	1,7788
tail16	23 503	23 416	23 319	23 743	23 187	23016	23503	23 380	**22 011**	4,3665
tail17	24 371	24 236	24 574	24 344	24 264	23967	24372	24 500	**21 939**	8,4616
tail18	24 614	24 416	24 878	24 294	24 294	24315	24294	24 294	**24 265**	0,1194
tail19	24 947	25 128	25 535	25 799	25 040	24663	24771	25 107	**23 522**	4,6264
tail20	26 688	25 638	25 966	26 243	25 864	25703	25704	25 638	**24 605**	4,0292

Source: Authors

Table 7. Total flowtime comparison for algorithms A-10 to A-17 for 20 x 20 data set from (Taillard, 1993)

Instance	A-10	A-11	**A-12**	**A-13**	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail21	41 278	40 426	40 080	40 207	40 929	39 522	41 464	39 688	**38 697**	2,0874
tail22	38 537	38 780	38 880	38 791	38 524	38 400	38 509	38 268	**37 571**	1,8214
tail23	40 972	40 439	39 807	39 845	39 564	39 911	40 327	40 037	**38 312**	3,1645
tail24	38 015	38 300	**37 157**	38 562	37 251	37 300	37 295	37 376	38 829	-4,4998
tail25	39 798	40 711	39 811	39 750	39 761	40 360	39 593	39 680	**39 071**	1,3184
tail26	38 900	38 667	39 372	38 652	38 419	38 787	38 900	38 660	38 620	-0,5232
tail27	40 556	39 865	40 663	39 902	40 170	39 849	40 183	39 902	**39 718**	0,3287
tail28	37 983	37 685	38 579	37 389	37 979	37 128	37 304	37 295	**37 000**	0,3448
tail29	38 294	38 616	38 669	**38 145**	38 649	38 555	38 230	38 616	39 228	-2,8392
tail30	38 400	38 406	37 956	38 404	38 362	38 297	38 479	38 033	**37 953**	0,0079

Source: Authors

between the proposed approach and the best solution by earlier approaches was more than 6%, while for the rest of the four problems the %Diff was 1,85%, 4,50%, 0,52% and 2,84% for problem instances *tail10*, *tail24*, *tail26* and *tail29*, respectively.

As mentioned earlier, the performance worsened for large sized problem instances. For $n = 50$, a total of 30 instances were solved, but the GA found a better solution only for 9 of them. However, for the 21 instances where proposed GA approach was not able to find a better solution than the best solution value among earlier approaches, the maximum **%Diff** was less than 5% with maximum **%Diff** being 4,48%. It may be noted here that the best value found by the proposed GA algorithm was not worse than all the previous algorithms under discussion. The proposed approach did find better solution compared to some of the earlier algorithms.

Although the proposed approach was not able to find better solutions for all the instances, the performance of the algorithm can be considered robust. The general-purpose nature and the ability to handle any objective function without changing the basic GA routine makes it a

Table 8. Total flowtime comparison for algorithms A-10 to A-17 for 50 x 5 data set from (Taillard, 1993)

Instance	A-10	**A-11**	**A-12**	**A-13**	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail31	82 183	81 613	80 843	79 569	79 471	78 746	**78 675**	79 562	80 701	-2,5752
tail32	90 846	91 092	89 181	89 391	88 454	87 771	88 192	86 395	**86 105**	0,3357
tail33	82 738	83 096	82 351	81 796	82 218	80 939	82 108	81 122	**80 561**	0,4670
tail34	86 173	83 711	84 422	83 572	84 250	**82 681**	82 807	83 257	84 991	-2,7939
tail35	87 367	88 054	85 446	86 504	85 680	**83 558**	84 430	85 763	86 789	-3,8668
tail36	89 192	87 431	88 293	87 577	85 739	84 831	**84 653**	86 354	84 781	-0,1512
tail37	85 884	85 001	82 610	84 657	82 335	83 210	82 063	83 010	**81 998**	0,0792
tail38	85 103	85 607	87 387	83 344	83 365	82 538	84 816	84 082	**81 934**	0,7318
tail39	80 444	80 683	82 794	79 804	79 978	78 646	77 996	77 992	**77 916**	0,0974
tail40	88 675	87 376	86 849	87 237	85 946	85 878	84 389	**84 142**	85 670	-1,8160

Source: Authors

Table 9. Total flowtime comparison for algorithms A-10 to A-17 for 50 x 10 data set from (Taillard, 1993)

Instance	**A-10**	**A-11**	**A-12**	**A-13**	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail41	12 0090	11 7480	11 7234	11 6704	11 6969	**11 5561**	11 5753	11 6122	11 7654	-1,8112
tail42	11 8203	11 6111	11 6199	11 4548	11 3873	11 3447	11 3481	**11 2619**	11 7445	-4,2852
tail43	11 7403	11 7158	11 4350	11 5547	11 4235	11 4242	11 4754	11 4880	**11 0999**	2,8328
tail44	12 2769	12 0536	12 0652	11 7684	11 8586	11 7617	**11 5956**	11 6836	11 7599	-1,4169
tail45	12 0773	12 3084	12 2743	11 9960	12 0242	11 9692	11 9953	**11 9499**	12 0528	-0,8611
tail46	12 0201	11 8519	11 9088	11 8942	11 6570	11 6549	11 8320	11 8467	**11 6090**	0,3938
tail47	12 2457	12 3182	12 3595	12 2566	11 9751	11 9805	**11 8958**	12 0075	12 2151	-2,6841
tail48	11 6975	11 7187	11 5611	11 6316	11 6003	11 5041	**11 4624**	11 5720	11 8636	-3,5001
tail49	11 8063	11 6116	11 7939	11 5975	11 6323	11 6036	**11 4759**	11 6140	11 5648	-0,7747
tail50	12 1804	11 9112	12 1418	11 8504	11 8031	11 8225	11 7610	**11 6781**	11 8053	-1,0892

Source: Authors

Table 10. Total flowtime comparison for algorithms A-10 to A-17 for 50 x 20 data set from (Taillard, 1993)

Instance	**A1-0**	**A-11**	**A-12**	**A-13**	**A-14**	**A-15**	**A-16**	**A-17**	**GA**	**% Diff**
tail51	17 8954	17 2365	17 4116	17 2570	17 3683	17 2252	**17 1545**	17 2254	17 8630	-4,1301
tail52	16 9880	17 0373	17 0720	16 7220	**16 6390**	16 9428	16 8870	16 6792	16 6887	-0,2987
tail53	17 5244	17 3685	17 5598	17 0515	17 2739	17 1590	17 0143	17 0554	**16 7089**	1,7950
tail54	17 2895	17 2186	17 1659	17 2193	16 8989	17 1063	16 8895	17 1055	**16 7904**	0,5868
tail55	17 2514	17 1821	16 8248	17 1365	**16 6783**	16 7471	16 8437	16 9655	17 3415	-3,9764
tail56	17 2492	17 2528	17 0262	17 1498	16 9714	**16 8527**	17 1708	17 0539	16 8755	-0,1353
tail57	17 7382	17 6812	17 7987	17 6985	17 1602	17 2083	**17 1442**	17 4218	17 3165	-1,0050
tail58	16 9268	16 9049	17 3768	16 7918	16 5782	16 6297	16 5887	**16 5601**	17 3020	-4,4800
tail59	17 4213	17 1749	17 2095	17 3293	**16 9524**	16 9937	17 0904	17 1078	17 2826	-1,9478
tail60	17 8270	17 4981	17 5283	17 4576	17 3446	**17 3288**	17 4594	17 3635	17 5483	-1,2667

Source: Authors

truly general purpose scheduling approach. Furthermore, arrangement of data and schedule in familiar spreadsheet environment also makes it easy to use in shop floor environment. The general-purpose nature and robustness of the algorithm to address a large number of problems has been the key advantage of the proposed approach.

Conclusions

In this paper, a no-wait flow shop scheduling problem was considered where the objective was to minimise total flowtime. The problem has practical applications in process industries and is considered to be NP-hard even for 3-machine cases (Hans, 1984).

Though the performance of the proposed approach though was inferior in some cases, it found solutions that were equal or better than those of previous studies in a wide range of problems. The empirical analysis shows that the proposed approach can solve large sized flow shop problems with reasonable accuracy. The %Diff was calculated between the solution found by the proposed approach and the best value among all the previous solution techniques. For problem set 1, the proposed GA found better solution for six instances out of eight problem instances, while the same solution for the remaining two. For problem set 2, out of twenty-one instances, the proposed approach found better solutions for 17 instances and worse for four of them with maximum **% Diff** less than 10%. Problem set 3 consisted of only two problems. The proposed approach found a better solution for the small-sized problem, while for large-sized problem the **% Diff** was 5,50%. For problem set 4, the proposed approach found better solutions for 33 instances out of sixty problems and worse for the remaining 27 with a maximum **% Diff** of 6,72%. It may be noted here that the best value found by the proposed GA algorithm was not worse to all the previous algorithms under discussion. The proposed approach did find better solution compared to some of the earlier algorithms.

It was demonstrated that the proposed algorithm is simple to implement and easily customisable to include additional jobs or machines. The proposed GA approach has been implemented in a familiar spreadsheet interface and has the ability to generate Gantt chart, thus presenting a graphical representation of the schedules which is easily understandable by shop floor managers.

Acknowledgements

This research is supported by the Deanship of Academic Research at University of Hail by a grant for Project Number 160769.

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*(5-8), 1181-1188. DOI: 10.1007/s00170-013-5351-9.

Aldowaisan, T., & Allahverdi, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time. *Omega*, *32*(5), 345-352. DOI: 10.1016/j.omega.2004.01.004.

Astaiza A, L. G. (2005). A practical approach to scheduling examinations. *Ingeniería e Investigación*, *25*(3), 92-100.

Bertolissi, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology*, *107*(1–3), 459-465. DOI: 10.1016/S0924-0136(00)00720-2.

Bewoor, L., Chandra Prakash, V., & Sapkal, S. (2017a). Evolutionary Hybrid Particle Swarm Optimization Algorithm for Solving NP-Hard No-Wait Flow Shop Scheduling Problems. *Algorithms*, *10*(4), 121. DOI: 10.3390/a10040121.

Bewoor, L. A., Prakash, V. C., & Sapkal, S. U. (2017b). Comparative Analysis of Metaheuristic Approaches for m-Machine No Wait Flow Shop Scheduling for minimizing Total Flow Time with Stochastic Input. *International Journal of Engineering and Technology*, *8*(6), 3021-3026. DOI: 10.21817/ijet/2016/v8i6/160806265.

Bewoor, L. A., Prakash, V. C., & Sapkal, S. U. (2018). Production scheduling optimization in foundry using hybrid Particle Swarm Optimization algorithm. *Procedia Manufacturing*, *22*, 57-64. DOI: 10.1016/j.promfg.2018.03.010.

Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *R.A.I.R.O. Recherche operationelle/Operations Research*, *12*(4), 333-350. DOI: 10.1051/ro/1978120403331.

Chaudhry, I. A., Ahmed, R., & Khan, A. M. (2014). Genetic Algorithm to minimize flowtime in a no-wait flowshop scheduling problem. *IOP Conference Series: Materials Science and Engineering*, *65*(1), 1-6. DOI: 10.1088/1757-899X/65/1/012007.

Chaudhry, I. A., & Elbadawi, I. A. Q. (2017). Minimisation of total tardiness for identical parallel machine scheduling using genetic algorithm. *Sadhana - Academy Proceedings in Engineering Sciences*, *42*(1), 11-21. DOI: 10.1007/s12046-016-0575-7.

Chaudhry, I. A., & Khan, A. M. (2012). Minimizing makespan for a no-wait flowshop using genetic algorithm. *Sadhana - Academy Proceedings in Engineering Sciences*, *37*(6), 695-707. DOI: 10.1007/s12046-012-0105-1.

Davis, L. (1985). *Job Shop Scheduling with Genetic Algorithms*. Paper presented at the Proceedings of the 1st International Conference on Genetic Algorithms.

Delgado, E., Rodríguez, C. J. C., & Velasco, Ó. G. D. (2005). Applying genetic algorithms for programming manufactoring cell tasks. *Ingeniería e Investigación*, *25*(2), 24-31.

Díaz Ramírez, J., & Huertas, J. I. (2018). A continuous time model for a short-term multiproduct batch process sche-

duling. *Ingeniería e Investigación*, *38*(1), 96-104. DOI: 10.15446/ing.investig.v38n1.66425.

Dong, B., Gao, K.-z., Pan, Q.-k., & Sun, Q.-q. (2010). Hybrid differential evolution optimization algorithm for no-wait flow shop problem with total flow time criterion. *Application Research of Computers*, *27*(8), 2875-2877. DOI: 10.1007/978-3-642-24728-6_81.

Framinan, J. M., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flow shops. *Omega - International Journal of Management Science*, *31*(4), 311-317. DOI: 10.1016/s0305-0483(03)00047-1.

Frutos, M., & Tohmé, F. (2012). Evolutionary multi-objective scheduling procedures in non-standardized production processes. *DYNA*, *79*(172), 101-107.

Gao, K.-Z., Pan, Q.-K., & Li, J.-Q. (2011a). Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. *International Journal of Advanced Manufacturing Technology*, *56*(5-8), 683-692. DOI: 10.1007/s00170-011-3197-6.

Gao, K.-Z., Pan, Q.-K., Li, J.-Q., & Jia, B.-X. (2011b). Improved Harmony Search Algorithm for No-Wait Flow Shop Schedule. *Computer Engineering*, *37*(8), 178-180. DOI: 10.3969/j.issn.1000-3428.2011.08.061.

Gao, K.-Z., Pan, Q.-K., Li, J.-Q., Wang, Y.-T., & Liang, J. (2012). A hybrid harmony search algorithm for the no-wait flowshop scheduling problems. *Asia-Pacific Journal of Operational Research*, *29*(2), 1250012/1250011-1250023. DOI: 10.1142/S0217595912500121.

Gao, K.-z., Pan, Q., Li, J., & He, Y. (2010). *A novel grouping harmony search algorithm for the no-wait flow shop scheduling problems with total flow time criteria*. Paper presented at the 2010 International Symposium on Computer Communication Control and Automation (3CA) Tainan, Taiwan. DOI: 10.1109/3CA.2010.5533729.

Gao, K.-z., Pan, Q., Suganthan, P. N., & Li, J. (2013). Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. *International Journal of Advanced Manufacturing Technology*, *66*(9-12), 1563-1572. DOI: 10.1007/s00170-012-4440-5.

Gilmore, P. C., & Gomory, R. E. (1964). Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem. *Operations Research*, *12*(5), 655-679. DOI: 10.1287/opre.12.5.655.

Guang, X., & Junqing, L. (2012). *Evolved Discrete Harmony Search Algorithm for Multi-objective No-wait Flow Shop Scheduling Problem*. Paper presented at the 2nd International Conference on Computer Application and System Modeling, Taiyuan Institute of Science and Technology, Taiyuan, Shanxi, China. DOI: 10.2991/iccasm.2012.200.

Gupta, J. N. D., & Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, *169*(3), 699-711. DOI: 10.1016/j.ejor.2005.02.001.

Hall, N., & Sriskandarajah, C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research*, *44*(3), 510-525. DOI: 10.2307/171711.

Hans, R. (1984). The Three-Machine No-Wait Flow Shop is NP-Complete. *Journal of the Association for Computing Machinery*, *31*(2), 336-345. DOI: 10.1145/62.65.

Heller, J. (1960). Some Numerical Experiments for an M × J Flow Shop and Its Decision - Theoretical Aspects. *Operations Research*, *8*(2), 178-184. DOI: 10.1287/opre.8.2.178.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Huang, R.-H., Yang, C.-L., & Liu, S.-C. (2015). No-Wait Flexible Flow Shop Scheduling with Due Windows. *Mathematical Problems in Engineering*, 9 pages. DOI: 10.1155/2015/456719.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61-68. DOI: 10.1002/nav.3800010110.

Laha, D., & Chakraborty, U. K. (2008). A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, *41*(1-2), 97-109. DOI: 10.1007/s00170-008-1454-0.

Laha, D., Gupta, J. N. D., & Sapkal, S. U. (2014a). A penalty-shift-insertion-based algorithm to minimize total flow time in no-wait flow shops. *Journal of the Operational Research Society*, *65*(10), 1611-1624. DOI: 10.1057/jors.2013.118.

Laha, D., & Sapkal, S. U. (2011). *An Efficient Heuristic Algorithm for m-Machine No-wait flow shops*. Paper presented at the International MultiConference of Engineers and Computer Scientists, Hong Kong.

Laha, D., & Sapkal, S. U. (2014b). An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop. *Computers & Industrial Engineering*, *67*, 36-43. DOI: 10.1016/j.cie.2013.08.026.

Miyata, H. H., Nagano, M. S., & Gupta, J. N. D. (2018). Incorporating preventive maintenance into the m-machine no-wait flow-shop scheduling problem with total flow-time minimization: a computational study. *Engineering Optimization*, 1-19. DOI: 10.1080/0305215X.2018.1485903.

Nagano, M. S., Miyata, H. H., & Araújo, D. C. (2015). A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. *Journal of Manufacturing Systems*, *36*, 224–230. DOI: 10.1016/j.jmsy.2014.06.007.

Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega - International Journal of Management Science*, *11*(1), 91-95. DOI: 10.1016/0305-0483(83)90088-9.

Qi, X., Wang, H., Zhu, H., Zhang, J., Chen, F., & Yang, J. (2016). Fast local neighborhood search algorithm for the no-wait flow shop scheduling with total flow time minimization. *International Journal of Production Research*, *54*(16), 4957-4972. DOI: 10.1080/00207543.2016.1150615.

Rajendran, C., & Chaudhuri, D. (1990). Heuristic algorithms for continuous flow-shop problem. *Naval Research Logistics*, *37*(5), 695-705. DOI: 10.1002/1520-6750(199010)37:5<695::AID-NAV3220370508>3.0.CO;2-L.

Reddi, S. S., & Ramamoorthy, C. V. (1972). On the Flow-Shop Sequencing Problem with No Wait in Process. *Journal of the Operational Research Society*, *23*(3), 323-331. DOI: 10.1057/jors.1972.52.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, *22*(1), 5-13. DOI: 10.1016/0305-0548(93)E0014-K.

Sapkal, S. U., & Laha, D. (2013). A heuristic for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, *68*(5-8), 1327-1338. DOI: 10.1007/s00170-013-4924-y.

Shafaei, R., Moradinasab, N., & Rabiee, M. (2011). Efficient meta heuristic algorithms to minimize mean flow time in no-wait two stage flow shops with parallel and identical machines. *International Journal of Management Science and Engineering Management*, *6*(6), 421-430. DOI: 10.1080/17509653.2011.10671192.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285. DOI: 10.1016/0377-2217(93)90182-M.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P. N., & Buyukdagli, O. (2013). A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Computers & Operations Research*, *40*(7), 1729-1743. DOI: 10.1016/j.cor.2013.01.005.

Tyagi, N., Varshney, N. G., & Chandramouli, A. B. (2013). Six decades of flowshop scheduling research. *International Jouranal of Scientific & Engineering Research*, *4*(9), 854-864.

Whitley, D., & Kauth, K. (1988). *GENITOR: A different genetic algorithm*. Paper presented at the Proceedings of the 1988 Rocky Mountain Conference on Artificial Intelligence.

Ying, K.-C., Lin, S.-W., & Wu, W.-J. (2016). Self-adaptive ruin-and-recreate algorithm for minimizing total flow time in no-wait flowshops. *Computers & Industrial Engineering*, *101*(C), 167-176. DOI: 10.1016/j.cie.2016.08.014.

Zhu, X., & Li, X. (2015). Iterative search method for total flow-time minimization no-wait flowshop problem. *International Journal of Machine Learning and Cybernetics*, *6*(5), 747–761. DOI: 10.1007/s13042-014-0312-7.