

HEURISTIC FOR THE M-MACHINE NO-WAIT FLOW SHOP SCHEDULING PROBLEM WITH HIERARCHICAL OBJECTIVE

Fernando Siqueira de Almeida

University of São Paulo, School of Engineering of São Carlos, Department of Production
Engineering
Av. Trabalhador São-Carlense 400, 13566-590 São Carlos, São Paulo, Brazil
fernando.siqueira.almeida@usp.br

Marcelo Seido Nagano

University of São Paulo, School of Engineering of São Carlos, Department of Production
Engineering
Av. Trabalhador São-Carlense 400, 13566-590 São Carlos, São Paulo, Brazil
drnagano@usp.br

ABSTRACT

This paper addresses the m -machine no-wait flow shop scheduling problem. Two hierarchical objective functions are considered: (1) minimizing makespan subject to the constraint that mean completion time is less than or equal to a given value; and (2) minimizing total completion time subject to the constraint that makespan is less than or equal to a given value. For each objective, an Iterated Greedy with Local Search (IG_{LS}) algorithm is proposed. Computational experiments were performed to evaluate the performance of IG_{LS} against other heuristics in the literature. The results show that the IG_{LS} outperform the other methods in effectiveness and efficiency.

Keywords: No-wait flowshop. Makespan. Mean completion time.

1. Introduction

In a m -machine flow shop scheduling problem, n jobs have to be processed. There are m machines available. Each job requires m operations and each operation requires a different machine. A machine can process at most one job at a time. All jobs are processed on the same order of machines. An unlimited buffer can be assumed so that the jobs can wait in a queue between successive machines. All queues operate under the *First In First Out* (FIFO) discipline; therefore, jobs cannot overtake other jobs and only permutation schedules are possible [Emmons and Vairaktarakis 2013; Pinedo 2016]. The problem consists of find the best sequence of jobs that optimize some performance measure.

The no-wait flow shop is a case in which jobs can be in the system only in process. In other words, once the processing of a job begins, it must flow without any interruption through every machine until the last operation is completed. The no-wait flow shop can occur for null waiting space, technological or process requirement, unavailability of resources and many other particular reasons. Several cases are found in the metal, chemical and food industries, where controlling some parameters of the work in process requires that each operation follows the previous one immediately. [Emmons and Vairaktarakis 2013]. For example, during deformation processes in manufacturing industries, the work material is often heated and delays are not allowed because they can result in cooling [Groover 2014]. Similarly, the problem can be found in semiconductor manufacturing (Chien et al. 2008; Ritzo et al. 2011) and electroplating lines [Che and Chu 2007]. In addition, modern environments such as flexible manufacturing systems, just-in-time and agile manufacturing can also be modeled as a no-wait scheduling problem [Bertolissi 2000].

Minimizing makespan and total completion time (or the equivalent mean completion time) are two common objectives in the no-wait flow shop problem, both of which are intended to make as small as possible. Minimizing makespan is used when the goal is the efficient utilization of resources; that is, reduce equipment idle time. This is important when the prompt delivery of a complete batch is required. Minimizing total completion time means, in practice, increasing the processing rate and is appropriate when it is intended to ship each job as soon as it is completed. This is important when the goal is to decrease the work-in-process inventory and provide rapid response to demands. [Aydilek and Allahverdi 2012; Maccarthy and Liu 1993].

In addition to cases with a single objective, a scheduling problem can involve multiple objectives. When two performance parameters, A and B , are considered, the objective function can be formulated as hierarchical objective ($A \mid B$), composite objective ($\alpha A + \beta B$) or bicriteria objective (A, B). Hierarchical objective minimizes A subject to a constraint on B . Composite objective minimizes a linear combination of the two parameters, for relative weights α and β . Bicriteria objective finds the schedules with the best pairs of A and B , which should be used as options for a final selection [Emmons and Vairaktarakis 2013]. These problems are considered intractable (NP-Hard), which means that the proposed methods to solve them do not guarantee optimum solution. Therefore, the development of heuristics to obtain good solutions at a reasonable computational cost represents an important topic of study [Nagano e Miyata 2016].

Over the last years, researchers have begun studying the application of algorithms to multi-objective scheduling problems. Allahverdi and Aldowaisan [2002] considered the no-wait flow shop problem with the objective of minimizing a weighted sum of makespan and the total completion time. Framinan and Leisten [2006] considered a regular flow shop with the objective of minimizing makespan subject to the constraint that the maximum tardiness should be less than or equal to a given value. Minimizing makespan subject to the constraint that mean completion time should not be greater than or equal to a given value was addressed by Aydilek and Allahverdi [2012] for the same problem. Allahverdi and Aydilek [2013] addressed the flow shop with the objective of minimizing total completion time subject to the constraint that makespan should be less than or equal to a given value. Recently, Allahverdi et al. [2018] proposed an algorithm to minimize the total tardiness subject to the constraint that makespan should not be greater than a given value. Other publications in this area can be found in the studies reviews and

surveys presented by Hall and Sriskandarajah (1996), Nagano and Miyata (2016) and Allahverdi [2016].

In this paper, the m -machine no-wait flow shop problem is addressed. Two hierarchical objectives are considered: (1) minimizing makespan subject to the constraint that mean completion time is less than or equal to a given value; and (2) minimizing total completion time subject to the constraint that makespan is less than or equal to a given value. The Iterated Greedy with Local Search (IG_{LS}), proposed by Ruiz and Stützle [2007b], is adapted for both objectives. For Objective (1), the IG_{LS} is compared with the heuristic HH1, proposed by Aydilek and Allahverdi [2012]. For Objective (2), the IG_{LS} is evaluated against the algorithm PA20, proposed by Allahverdi and Aydilek [2013]. The remainder of this paper is organized as follow. Section 2 is dedicated to the problem definition. The algorithm used to generate the constraints and initial solutions are presented in Section 3. In Section 4, the proposed heuristic is described. Section 5 is devoted to the computational experiment. Finally, the conclusions are presented in Section 6.

2. Problem Definition

It should be noted that the objectives addressed for the no-wait flow shop problem in this paper consider two equivalent parameters: mean completion time and total completion time. This performance measures differ only by a constant factor. However, both are used to facilitate the comparison of the proposed heuristics with those selected in literature.

Let $C_{\max}$, MCT and TCT denote makespan, mean completion time and total completion time, respectively. Further, let $C_{\max}(\pi)$, $MCT(\pi)$, and $TCT(\pi)$ represent makespan, mean completion time and total completion time of a given sequence π . The problems can be defined as: (1) $Fm | nwt (perm) | (C_{\max} | MCT \leq M)$ and (2) $Fm | nwt (perm) | (TCT | C_{\max} \leq K)$.

In expression (1), the objective is to find a sequence of jobs that minimizes makespan such that the mean completion time is less than or equal to a M value. In expression (2), the objective is to find a sequence that minimizes total completion time such that the makespan is less than or equal to a K value. M and K are upper bounds on the value of makespan and mean completion time, respectively. For a given problem, these values should be given by scheduler. If M or K value is not given, they can be obtained by an algorithm like the one presented in the next section.

3. Algorithm-R

The Algorithm-R can be used to generate the initial solutions with a relatively tight upper bound on the value of MCT or $C_{\max}$. In essence, Algorithm-R is a compilation of the Algorithm-M presented by Aydilek and Allahverdi [2012] and the Algorithm-K presented by Allahverdi and Aydilek [2013]. The only difference is that there is a generic function $F(\pi)$ that can represent $MCT(\pi)$ or $C_{\max}(\pi)$. The complete algorithm is presented in Fig. 1.

Algorithm-R

```

for  $p := 1$  to  $n$  do
     $\pi_p :=$  random sequence;
     $R_p := F(\pi_p)$ ;
    for  $h := 1$  to  $n - 1$  do
         $\pi_0 :=$  swap jobs  $h$  and  $h+1$  in  $\pi_p$ ;
        if  $F(\pi_0) < R_p$  then
             $\pi_p := \pi_0$ ;
             $R_p := F(\pi_0)$ ;
        endif
    endfor
endfor
 $R := \min(R_1, R_2, \dots, R_n)$ ;
 $\pi := \min_R(\pi_1, \pi_2, \dots, \pi_n)$ ;
return  $\pi, R$ 
end

```

Fig. 1 Algorithm-R

4. Iterated Greedy with Local Search

The IG_{LS} heuristic performs, from an initial solution, a sequence of iterations in the search for a solution candidate. The process occurs in two phases: destruction and construction. During destruction, d random elements are removed from the previous solution. Then, in the construction phase, the removed elements are reinserted to construct a new sequence through the NEH constructive heuristic of Nawaz et al. [1983]. A local search process is also added to improve the solution found. Finally, an acceptance criterion is applied using a parameter T , similar to the Simulated Annealing criterion [Yang 2010, 182]. The entire process is repeated until some stopping criterion is satisfied, such as computational time or a number of iterations [Nagano et al., 2015].

In the proposed adaptation, only sequences that respect the constraint by which they are subject (M or K) are accepted. Two neighborhood local searches are implemented – insertion and interchange – based on Nagano et al. [2015]. The stopping criterion is given by a number of iterations. For convenience, the proposed heuristic is denoted by GL where L represent the number of iterations. The algorithm of the complete adaptation is illustrated in Fig. 2.

IG_{LS} has three input parameters: d , T and L . In order to calibrate the parameter d , values 2, 4, 6, 8, 10 and 12 for small problems and values 5, 10, 15, 20, 25 and 30 for large problems were tested. It is obvious that a large number for d increase the performance of the heuristic. However, different from what was expected, the increase in CPU Time was not significant due to the increase of d . Then we set $d = 12$ for small problems and $d = 30$ for large problems. In addition, we set $T = 0.5$ which is the best value found for a regular flowshop scheduling problem in the experiments of Ruiz and Stützle [2007b]. For the parameter L , five values (1, 5, 10, 15 and 20) were set to generate five versions GL (G1, G5, G10, G15 and G20).

GL heuristic

```

Initialize:  $L, T_{emp}, d$ ;
 $\pi :=$  initial solution (from Algorithm-R);
 $\pi_b := \pi$ ;
for  $i := 1$  to  $L$  do
     $\pi' := \pi$ ;
     $\pi_d :=$  empty sequence
    for  $i := 1$  to  $d$  do           # Destruction phase
        remove one job at random from  $\pi'$  and insert it in  $\pi_d$ ;
    endfor
    for  $i := 1$  to  $d$  do           # Construction phase
         $\pi' :=$  best feasible sequence obtained by inserting job  $\pi_d(i)$  in all possible positions of  $\pi'$ ;
    endfor
     $\pi' :=$  best feasible sequence of  $(n - 1)^2$  permutations by insertion neighborhood # Local search
     $\pi' :=$  best feasible sequence of  $\binom{n}{2}$  permutations by interchange neighborhood # Local search
    if  $F(\pi') < F(\pi)$  then           # Acceptance Criterion
         $\pi := \pi'$ ;
        if  $F(\pi') < F(\pi_b)$  then       # Check if new best sequence
             $\pi_b := \pi'$ ;
        endif
    elseif  $(exp(-\Delta/T_{emp}) \geq r)$  then
         $\pi := \pi'$ ;
    endif
endfor
return  $\pi_b$ 
end

```

Fig. 2 Algorithm for IG_{LS} heuristic. Adapted from Ruiz and Stützle (2007).

$$\Delta = -\frac{F(\pi'') - F(\pi)}{F(\pi)} \quad (1)$$

$$T_{emp} = T \frac{\sum_{i=1}^m \sum_{j=1}^n t_{ij}}{m \times n \times 10}, \quad 0 < T < 1. \quad (2)$$

5. Computational Experiment

The heuristics HH1, PA20 and GL were implemented in C. The five versions of GL (G1, G5, G10, G15, G20) were adapted for the two optimization objectives addressed. The computer used was a PC with Intel Core i5-4200U 1.60 GHz CPU with Turbo Impulse for 2.30 GHz, 6 GB of RAM and operating under the Windows 10 operating system.

Table 1 ARPD of heuristics (Objective 1)

Heuristic		HH1	G1	G5	G10	G15	G20
n	m						
50	10	1.231	7.341	1.526	0.991	0.503	0.415
	20	1.900	8.323	1.899	0.998	0.405	0.313
	30	1.498	7.402	1.503	0.915	0.601	0.355
	40	1.499	7.652	1.811	0.856	0.430	0.235
100	10	1.369	19.154	4.005	1.427	0.491	0.329
	20	1.595	20.441	5.211	1.553	0.652	0.261
	30	2.123	21.629	5.437	1.813	0.984	0.061
	40	2.124	22.321	5.396	2.011	0.688	0.165
150	10	3.456	24.588	7.000	2.637	0.982	0.152
	20	4.250	27.483	8.724	2.846	0.943	0.190
	30	3.917	27.772	8.778	3.168	0.929	0.098
	40	4.537	28.678	9.489	3.355	1.509	0.100
200	10	5.156	27.221	9.830	3.343	1.358	0.060
	20	6.223	29.787	11.329	4.289	1.447	0.054
	30	5.990	30.668	11.372	4.369	1.664	0.019
	40	6.326	31.021	11.778	4.499	1.752	0.000

Table 2 ARPD of heuristics (Objective 2)

Heuristic		PAL	G1	G5	G10	G15	G20
	n	m					
50	10	7.001	8.319	1.853	0.905	0.668	0.211
	20	6.541	7.720	1.575	0.851	0.674	0.241
	30	6.082	6.209	1.178	0.601	0.627	0.322
	40	6.066	6.461	1.528	0.799	0.372	0.249
100	10	16.729	20.028	4.792	1.433	0.613	0.260
	20	16.851	19.760	4.550	1.721	0.797	0.254
	30	15.771	18.762	3.919	1.440	0.289	0.381
	40	16.786	19.564	4.418	1.657	0.424	0.501
150	10	21.954	25.443	8.093	2.810	1.045	0.064
	20	23.123	26.893	7.408	2.688	0.692	0.227
	30	22.886	26.169	7.327	2.602	0.929	0.018
	40	23.696	27.237	8.050	2.518	0.884	0.054
200	10	25.935	29.657	10.171	3.776	1.378	0.075
	20	27.345	30.710	10.448	3.981	1.433	0.021
	30	26.937	31.062	10.611	3.686	1.056	0.000
	40	27.774	31.747	10.801	3.925	1.238	0.000

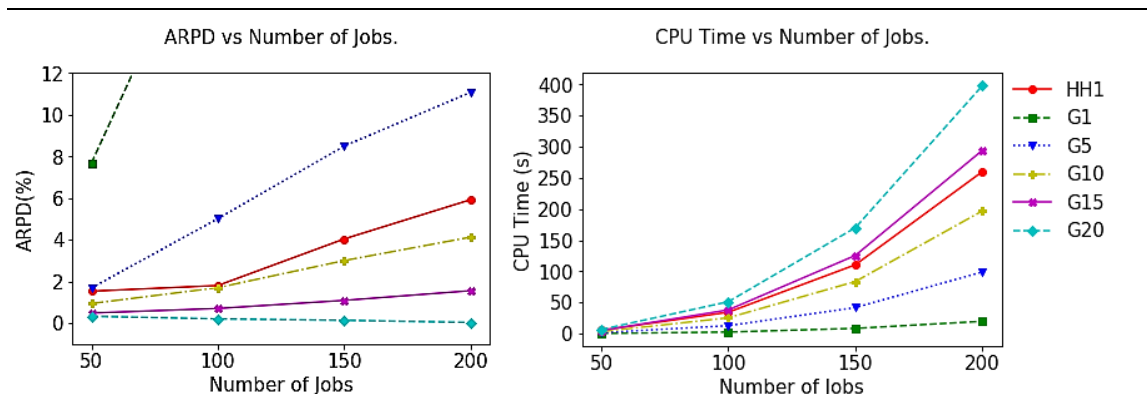


Fig. 3 ARPD and CPU Time vs Number of jobs (Objective 1).

The heuristics were tested using two sets of instances from the database of Ruiz et al. [2007a]. The set called *small* consist of all combinations of $n \in [15, 20, 25, 30] \times m \in [2, 3, 4, 5, 6]$. The set called *large* consist of all combinations of $n \in [50, 100, 150, 200] \times m \in [10, 20, 30, 40]$. Each combination of m and n consists of 25 different problems, totaling 500 for the *small* set and 400 for the *large* set. Processing times have a uniformly random discrete distribution $U(1,100)$. From this material, another database was created for each of the two objectives, but this time with the initial solutions and the constraints of all problems, through the implementation of the Algorithm-R. Finally, using these databases, all heuristics were implemented.

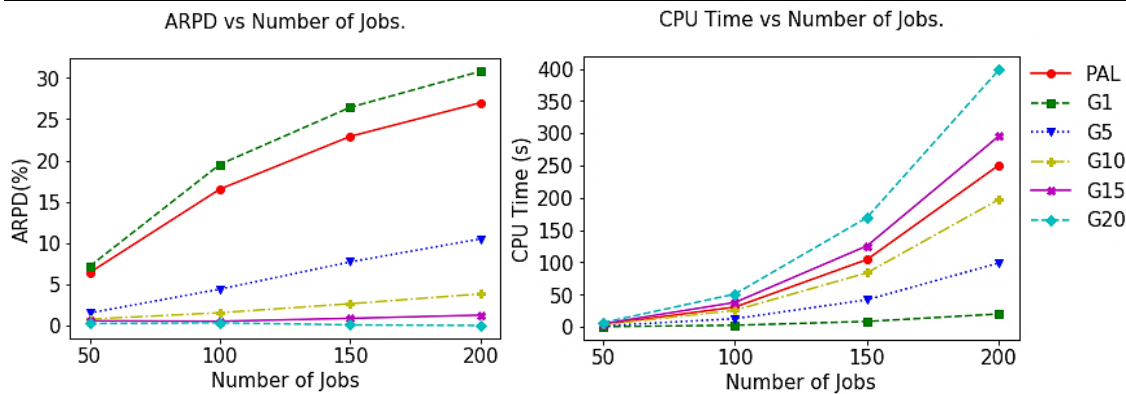


Fig. 4 ARPD and CPU Time vs Number of jobs (Objective 2).

Table 3 Results of Tukey' honestly significant difference tests.

Heuristics		Mean Difference (I-J)	Std. Error	Sig.	Heuristics		Mean Difference (I-J)	Std. Error	Sig.
I	J				I	J			
G1	G5	14,774575*	,289985	,000	G1	G5	14,9386532*	,3683876	,000
	G10	18,900681*	,289985	,000		G10	18,7717824*	,3683876	,000
	G15	20,383893*	,289985	,000		G15	20,1638823*	,3683876	,000
	G20	21,167110*	,289985	,000		G20	20,8038620*	,3683876	,000
	HH1	18,017820*	,289985	,000		PA20	2,7664446*	,3683876	,000
G5	G1	-14,774575*	,289985	,000	G5	G1	-14,9386532*	,3683876	,000
	G10	4,126107*	,289985	,000		G10	3,8331292*	,3683876	,000
	G15	5,609318*	,289985	,000		G15	5,2252291*	,3683876	,000
	G20	6,392535*	,289985	,000		G20	5,8652088*	,3683876	,000
	HH1	3,243245*	,289985	,000		PA20	-12,1722086*	,3683876	,000
G10	G1	-18,900681*	,289985	,000	G10	G1	-18,7717824*	,3683876	,000
	G5	-4,126107*	,289985	,000		G15	1,3920999*	,3683876	,002
	G15	1,483211*	,289985	,000		G20	2,0320796*	,3683876	,000
	G20	2,266428*	,289985	,000		G5	-3,8331292*	,3683876	,000
	HH1	-,882861*	,289985	,028		PA20	-16,0053378*	,3683876	,000
G15	G1	-20,383893*	,289985	,000	G15	G1	-20,1638823*	,3683876	,000
	G5	-5,609318*	,289985	,000		G10	-1,3920999*	,3683876	,002
	G10	-1,483211*	,289985	,000		G20	,6399797	,3683876	,507
	G20	,783217	,289985	,075		G5	-5,2252291*	,3683876	,000
	HH1	-2,366072*	,289985	,000		PA20	-17,3974377*	,3683876	,000
G20	G1	-21,167110*	,289985	,000	G20	G1	-20,8038620*	,3683876	,000
	G5	-6,392535*	,289985	,000		G5	-5,8652088*	,3683876	,000
	G10	-2,266428*	,289985	,000		G10	-2,0320796*	,3683876	,000
	G15	-,783217	,289985	,075		G15	-,6399797	,3683876	,507
	HH1	-3,149290*	,289985	,000		PA20	-18,0374174*	,3683876	,000
HH1	G1	-18,017820*	,289985	,000	PA20	G1	-2,7664446*	,3683876	,000
	G5	-3,243245*	,289985	,000		G5	12,1722086*	,3683876	,000
	G10	,882861*	,289985	,028		G10	16,0053378*	,3683876	,000
	G15	2,366072*	,289985	,000		G15	17,3974377*	,3683876	,000
	G20	3,149290*	,289985	,000		G20	18,0374174*	,3683876	,000

* The mean difference is significant at the 0.05 level.

As similar results were obtained in the computational experiments performed with small and large problems, the focus was given to the large problems. The performances of the heuristics were evaluated by average relative percentage deviation (ARPD) and CPU time. ARPD is defined as $(\bar{F}(H') - \bar{F}(H)) / \bar{F}(H)$ where $\bar{F}(H')$ and $\bar{F}(H)$ are the averages of the heuristic solution and the best solution, respectively. The results are presented in Table 1 and 2.

To justify the results statistically, the performances were compared using Tukey's honestly significant difference (HSD) test. A summary is given in Table 3 where is displayed the compared heuristics, the mean difference between their ARPD values, the standard error and the significance. Significant results at a level of 0.05 are denoted by *. The homogeneous subsets of

the heuristics are presented in Table 4 and 5. The ARPD comparisons can also be observed by investigating the confidence intervals of the means (Fig. 5).

Table 4 ARPD at confidence level of 95%
(Objective 1)

Heuristic	N	Subset for alpha = 0.05				
		1	2	3	4	5
G20	400	,175				
G15	400	,959				
G10	400		2,442			
HH1	400			3,325		
G5	400				6,568	
G1	400					21,342
Sig.		,075	1,000	1,000	1,000	1,000

Means for groups in homogeneous subsets.
Harmonic Mean Sample Size = 400.

Table 5 ARPD at confidence level of 95%
(Objective 2)

Heuristic	N	Subset for alpha = 0.05				
		1	2	3	4	5
G20	400	,1799				
G15	400	,8198				
G10	400		2,212			
G5	400			6,045		
PA20	400				18,217	
G1	400					20,984
Sig.		,507	1,000	1,000	1,000	1,000

Means for groups in homogeneous subsets.
Harmonic Mean Sample Size = 400.

As expected, the CPU time of GL gets larger as L increases, since it represents the number of iterations of the heuristic. It can be observed from Fig. 3 that, for Objective (1), G10 outperforms HH1 in both ARPD and CPU time. The difference between G10 and HH1 is significant at confidence level of 95%, according to the subsets of the Tukey' HSD test (Table 4). G15 and G20 also have better ARPD results but with the cost of a higher CPU Time. For Objective (2), as illustrated in Fig. 4, G5 and G10 easily outperform PA20 in ARPD even with shorter CPU times. The difference is confirmed statistically by the Tukey' HSD test (Table 5). Best ARPD results can be obtained in G15 and G20 with the cost of having higher CPU times.

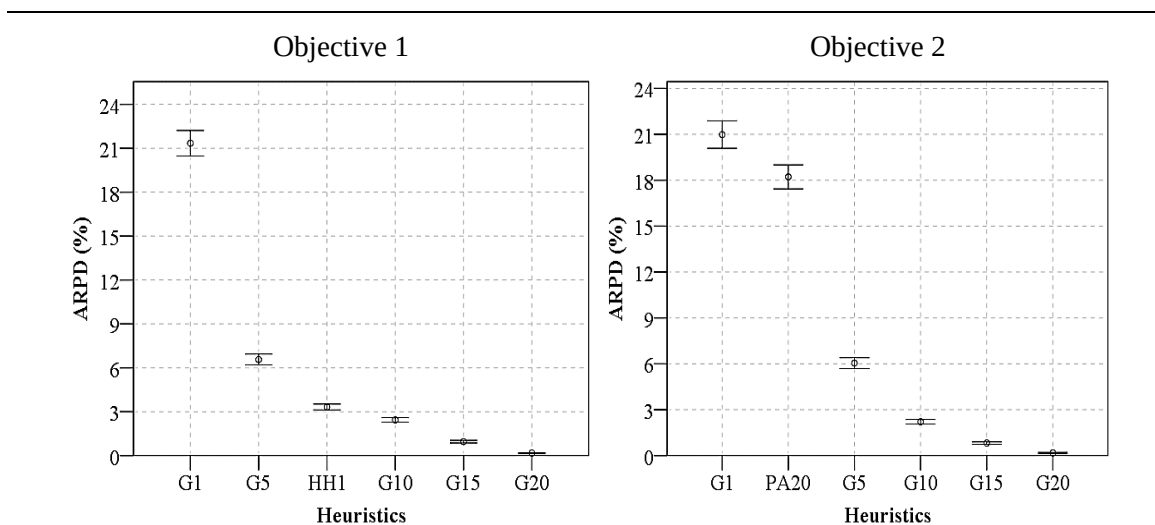


Fig. 5 ARPD (%) of heuristics (Objective 1 and 2).

Table 2 Overall performance for *small* problems

Objective 1			Objective 2		
Heuristic	ARPD	Time	Heuristic	ARPD	Time
HH1	0.81	0.45	PAL	2.69	0.18
G1	5.14	0.02	G1	20.98	0.02
G5	1.16	0.08	G5	6.05	0.08
G10	0.65	0.15	G10	2.21	0.15
G15	0.38	0.22	G15	0.82	0.22
G20	0.25	0.29	G20	0.18	0.30

Table 3 Overall performance for *large* problems

Objective 1			Objective 2		
Heuristic	ARPD	Time	Heuristic	ARPD	Time
HH1	3.32	102.64	PAL	18.22	97.46
G1	21.34	7.77	G1	20.98	7.85
G5	6.57	38.81	G5	6.05	38.86
G10	2.44	77.37	G10	2.21	77.62
G15	0.96	115.73	G15	0.82	116.12
G20	0.18	156.54	G20	0.18	156.39

Tables 2 and 3 summarize the overall ARPD and CPU Time of the heuristics for *small* and *large* problems, respectively. The versions of GL that outperform the heuristic of literature (red) in both performance measures are highlighted in blue.

6. Conclusions

In this paper, the no-wait flowshop scheduling problem was addressed with the objective of (1) minimizing makespan such that mean completion time is less than or equal to a given value and (2) minimizing the total completion time such that makespan is less than or equal to a given value. An algorithm was presented to get an initial solution and an upper bound for the restrictive parameter in case it is not known. Five versions of the heuristic GL (G1, G5, G10, G15 and G20) were proposed for the two objectives. Computational experiments with large instances indicated that, for Objective (1), G10 outperform HH1 in ARPD and CPU time. For Objective (2), the experiments also indicated the superiority of G5 and G10 over PA20 in both performance measures. Similar results were obtained for small instances. This clearly indicates that the heuristic GL performs better than the other heuristics.

It is important to note that between the three input parameters of GL (d , T and L), only variations of d and L were tested. Even that was defined the best value of T for a regular flowshop problem [Ruiz and Stützle 2007b], it is possible that this value is not the best for the no-wait case. In addition, the experiments indicated a gain of performance by increasing d without great cost of CPU time. These means that it is possible to explore variations of T and use values of d very close to the maximum number of jobs to improve performance even more. Another suggested extension is to consider setup and maintenance times. In this work, these times were ignored or assumed to be included in processing times. However, this approach may not be appropriate for some processing environments, and it is necessary to treat these constraints separately.

Acknowledgements

This research was supported by the National Council for Scientific and Technological Development (CNPq), Brazil.

References

- Allahverdi, A., H. Aydilek, e A. Aydilek. 2018. “No-wait flowshop scheduling problem with two criteria; total tardiness and makespan”. *European Journal of Operational Research* 269(2): 590–601. <https://doi.org/10.1016/j.ejor.2017.11.070>.
- Allahverdi, A, e T Aldowaisan. 2002. “No-wait flowshops with bicriteria of makespan and total completion time”. *Journal of the Operational Research Society* 53(9): 1004–15. <https://www.tandfonline.com/doi/full/10.1057/palgrave.jors.2601403> (24 de outubro de 2018).
- Allahverdi, Ali. 2016. “A survey of scheduling problems with no-wait in process”. *European Journal of Operational Research* 255(3): 665–86.
- Allahverdi, Ali, e Harun Aydilek. 2013. “Algorithms for no-wait flowshops with total completion time subject to makespan”. *The International Journal of Advanced Manufacturing Technology* 68(9–12): 2237–51. <http://link.springer.com/10.1007/s00170-013-4836-x> (21 de março de 2018).
- Aydilek, Harun, e Ali Allahverdi. 2012. “Heuristics for no-wait flowshops with makespan subject to mean completion time”. *Applied Mathematics and Computation* 219(1): 351–59. <http://dx.doi.org/10.1016/j.amc.2012.06.024>.
- Bertolissi, Edy. 2000. “Heuristic algorithm for scheduling in the no-wait flow-shop”. *Journal of Materials Processing Technology* 107(1–3): 459–65. <http://linkinghub.elsevier.com/retrieve/pii/S0924013600007202> (23 de outubro de 2018).
- Che, Ada, e Chengbin Chu. 2007. “Cyclic hoist scheduling in large real-life electroplating lines”. *OR Spectrum* 29(3): 445–70. <https://link.springer-com.ez67.periodicos.capes.gov.br/content/pdf/10.1007%2Fs00291-006-0040-9.pdf> (23 de

outubro de 2018).

Chien, Chen Fu et al. 2008. "Modeling and analysis of semiconductor manufacturing in a shrinking world: Challenges and successes". In *Proceedings - Winter Simulation Conference*, , 2093–99.

Emmons, Hamilton, e George Vairaktarakis. 2013. *Flow Shop Scheduling Theoretical Results, Algorithms, and Applications*. Springer Science & Business Media.

Framinan, Jose M, e Rainer Leisten. 2006. "A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness". *Production* 99: 28–40.
<https://www.sciencedirect.com/science/article/pii/S0925527304004499> (24 de outubro de 2018).

Hall, Nicholas G., e Chelliah Sriskandarajah. 1996. "A survey of machine scheduling problems with blocking and no-wait in process". *Operations Research* 44(3): 510(16).

Maccarthy, B. L., e Jiyin Liu. 1993. "Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling". *International Journal of Production Research* 31(1): 59–79.

Mikell P. Groover. 2014. *Igarss 2014 Fundamentals of Modern Manufacturing: Materials, Processes, and Systems*.

Nagano, Marcelo Seido, e Hugo Hissashi Miyata. 2016. "Review and classification of constructive heuristics mechanisms for no-wait flow shop problem". *International Journal of Advanced Manufacturing Technology* 86(5–8): 2161–74. <http://dx.doi.org/10.1007/s00170-015-8209-5>.

Nagano, Marcelo Seido, Hugo Hissashi Miyata, e Daniella Castro Araújo. 2015. "A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times". *Journal of Manufacturing Systems* 36: 224–30. <https://www-sciencedirect-com.ez67.periodicos.capes.gov.br/science/article/pii/S0278612514000739?via%3Dihub> (17 de maio de 2018).

Nawaz, Muhammad, E Emory Enscore, e Inyong Ham. 1983. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". *Omega* 11(1): 91–95.
<https://www.sciencedirect.com/science/article/pii/0305048383900889?via%3Dihub> (20 de outubro de 2018).

Pinedo, Michael. 2016. *Scheduling. Theory, Algorithms, and Systems*. 5º ed Springer Science & Business Media.

Ritzo, Ching Hua Chen, Sugato Bagchi, Lindsay E. Burns, e Steven C. Catlett. 2011. "Experiences in implementing simulation-based support for operational decision making in semiconductor manufacturing". *European J. of Industrial Engineering* 5(3): 272.
<http://www.inderscience.com/link.php?id=41617> (23 de outubro de 2018).

Ruiz, Rubén, e Ali Allahverdi. 2007. "Some effective heuristics for no-wait flowshops with setup times to minimize total completion time". *Ann Oper Res* 156: 143–71. <https://link-springer-com.ez67.periodicos.capes.gov.br/content/pdf/10.1007%2Fs10479-007-0227-8.pdf> (8 de abril de 2019).

Ruiz, Rubén, e Thomas Stützle. 2007. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem". *European Journal of Operational Research* 177(3): 2033–49. <https://www.sciencedirect.com/science/article/pii/S0377221705008507?via%3Dihub> (17 de maio de 2018).

Yang, Xin-She. 2010. *Engineering optimization : An introduction with metaheuristic applications*. John Wiley & Sons. www.wiley.com.