Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-10 (2022) 187–191

Permutation flowshop scheduling problem with Total Core Idle Time Minimization [★]

Sanchez-de-los-Reyes, Paula [*] Perez-Gonzalez, Paz [*]
Framinan, Jose M. [**]

[*] *Industrial Management Research Group, Universidad de Sevilla, 41092 Seville, Spain (e-mail: {psdelosreyes, pazperez}@ us.es).*
[**] *Industrial Management Research Group and the Laboratory of Engineering for Energy and Environmental Sustainability, Universidad de Sevilla, 41092 Seville, Spain (e-mail: framinan@us.es).*

Abstract: In this paper, we present a deterministic permutation flowshop scheduling problem with a new objective function, the total core idle time. The interest of this objective is related to reduce the energy consumption of the system, taking into account that the energy needed during the processing times is constant, and that machines are switched off during the front and back idle times. Therefore, the energy consumption is dependent on the time where machines are in stand-by mode, i.e during the idle time of machines between jobs, named as core idle times. Constructive heuristics and metaheuristics are adapted from the permutation flowshop scheduling literature for classical objectives as makespan and total completion time. Additionally, a new variant of one of the metaheuristic is proposed, the VBIH-P. An experimental evaluation has been carried out to analyse the performance of all the methods. The results show an excellent performance of the VBIH-P compared to the adapted methods.

Keywords: Production scheduling, metaheuristic, permutation flowshop, optimization.

1. INTRODUCTION

The permutation flowshop scheduling problem (PFSP) has been widely studied considering different objective functions as makespan or total completion time, see e.g. Fernandez-Viagas et al. (2017, 2020). The total core idle time is defined as the machines idle times between jobs, avoiding the front/back idle times generated by the first/last job in a given schedule (see Figure 1). Maassen et al. (2020) show that total core idle time minimization provides schedules with good makespan values whereas the opposite does not occur (i.e. schedules provided by makespan minimization are far from optimal for total core idle time). Therefore, considering total core idle time, the (hard) no-idle constraint is relaxed guaranteeing good values for makespan.

In this work, the justification of considering total core idle time as objective is due to energy efficiency reasons. During front/back idle times machines can be switched off, while during the core idle times machines should be on (or at least in a stand-by mode). Therefore, this problem is closely related to the minimization of the energy consumed by machines during the job processing, being constant when machines are processing the jobs (since the processing times are considered deterministic), but variable during the idle times.

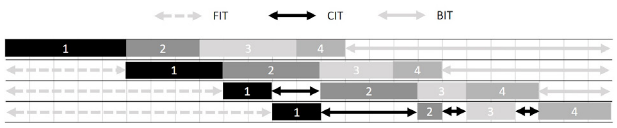


Fig. 1. Idle times in the PFSP: FIT front idle time; CIT: core idle time; BIT: back idle time

The PFSP with total core idle time (TCIT) is NP-hard in the strong sense (Maassen et al., 2020), so approximate methods should be developed to solve realistic sizes instances in reasonable computational times.

As, to the best of our knowledge, approximate methods have not been applied to the PFSP with TCIT minimization, in this paper, the main idea is to test methods proposed in the literature for related problems, in order to check their efficiency for this objective. Related problems can be PFSP with makespan and total completion time, due to the relation between these objectives and machine idle times (Maassen et al., 2020). We have adapted approximate methods in the literature developed for makespan to TCIT: the constructive heuristic Nawaz-Enscore-Ham (NEH) Nawaz et al. (1982), and the metaheuristics Iterate Greedy (IG) (Ruiz and Stützle, 2007), a variant named IGALL (Dubois-Lacoste et al., 2017) and Variable Block Insertion Heuristic (VBIH) (Tasgetiren et al., 2016). It has not been possible to include in this work the state-of-the-art method by Fernandez-Viagas and Framinan (2019), due to the high computational effort needed by this method in the adaptation, since the accelerations and critical path methods applied to the makespan

[★] This work is supported by the projects Ibsos (AT17_5920_US), Quid Pro Quo (AT17_5967_US), EFECTOS (US-1264511) and DEMAND (P18-FR-1149) funded by Junta de Andalucia, and the project Assort (PID2019-108756RB-I00) funded by the Spanish Government.

are not adaptable to the TCIT with similar complexity to the total completion time, see e.g. Fernandez-Viagas et al. (2020). Additionally, we have adapted approximate methods developed for total completion time: the version of the Liu-Reeves (LR) heuristic, named LR-NEH, since, according to Pan and Ruiz (2013), it is the best constructive heuristic with respect to the quality of the solution. Finally, taking into account that, the idle time is part of the Total Energy Consumption (TEC) objective (see e.g. Öztop et al., 2020a for details), those methods proposed by Öztop et al. (2020a) for the PFSP with total completion time and TEC as objectives, and Öztop et al. (2020b) for the hybrid flowshop scheduling problem with makespan and TEC have been adapted: two versions of the NEH named PFH-NEH and NEH-M as constructive heuristics, and IG, IGALL, VBIH as metaheuristics.

Therefore, seven methods have been adapted: four constructive heuristics (Section 2) and three metaheuristics (Section 3). Additionally a variant of one of the adapted metaheuristic (the VBIH) is proposed, and labelled as VBIH-P. Results are presented in Section 4. Among the adapted methods, the best results are provided by the adapted constructive heuristic NEH-M, and the adapted metaheuristic VBIH. Finally the variant proposed, VBIH-P gives the best results among all the tested methods.

2. CONSTRUCTIVE HEURISTICS

In this section we describe the constructive heuristics NEH, NEH-M, PFH-NEH and LR-NEH, previously mentioned in Section 1, and adapted for the TCIT objective.

In the NEH (Nawaz et al., 1982), the jobs are initially sorted in descending order of their total processing times. The first job is selected to obtain a partial solution with size one. Then, each job of the initial sequence is inserted into all possible positions of the new partial sequence to find the best position, until all jobs are scheduled.

The NEH-M(X) heuristic (Öztop et al., 2020a,b) takes into account the strong impact of the first job of the sequence in the objective function. Initially, the jobs are sorted in the same way than in the NEH. Then, X solutions are generated by choosing different jobs as the first job. In the h-th iteration, the job in position h in the initial sequence is chosen as the first job of the partial solution. Once the first job of this h-th solution is selected, the NEH procedure is applied. Finally, the best among the X solutions is selected as the final solution.

In the LR-NEH(X) heuristic (Pan and Ruiz, 2013), d jobs are selected to generate a partial solution by the LR method (Liu and Reeves, 2001). The LR(X) constructs the solution by appending each job based on an index function (the sum of the weighted total machine idle time and the artificial total flow time). The job with the minimum value of this index function is selected. Ties are broken by selecting the one with the minimum weighted total machine idle time. Then, the NEH is applied to the partial solution of size d by inserting the $n - d$ jobs. Due to the impact of the first job into the objective function, this procedure is repeated X times, choosing in the h-th iteration, the job in position h in the initial sequence by LR as the first job of the partial solution.

The PFH-NEH(X) (Öztop et al., 2020a,b) sorts the jobs in ascending order of a function computing the sum of the front delays and total processing times of each job. Then, X solutions are generated choosing different jobs as the first job. In the h-th iteration, the job number h of the initial sequence is chosen as the first job of the partial solution. Once the first job of this h-th solution is selected, the method PF (Profile Fitting by McCormick et al. (1989)) is applied, constructing the solution by appending each job based in an index function, computed as the sum of the blocking and idle times on the machines. Once the complete sequence is obtained, the NEH and LS (described in the next section) are applied.

3. METAHEURISTICS

In this section, IG, IGALL and VBIH are presented as the metaheuristics, adapted for our problem only when computing the objective function, and the local search applied. In this paper, the local search used in all the methods of this section has been selected by testing among the following local search methods: adjacent swap, general swap, insertion combined with first improvement and best improvement strategies. Finally, the local search based on general swap with first improvement, named LS, has been selected. Additionally, our proposal, denoted as VBIH-P is described too. Note that, in the adaptation, all the constructive heuristics previously described have been tested as initial solutions for the metaheursitics.

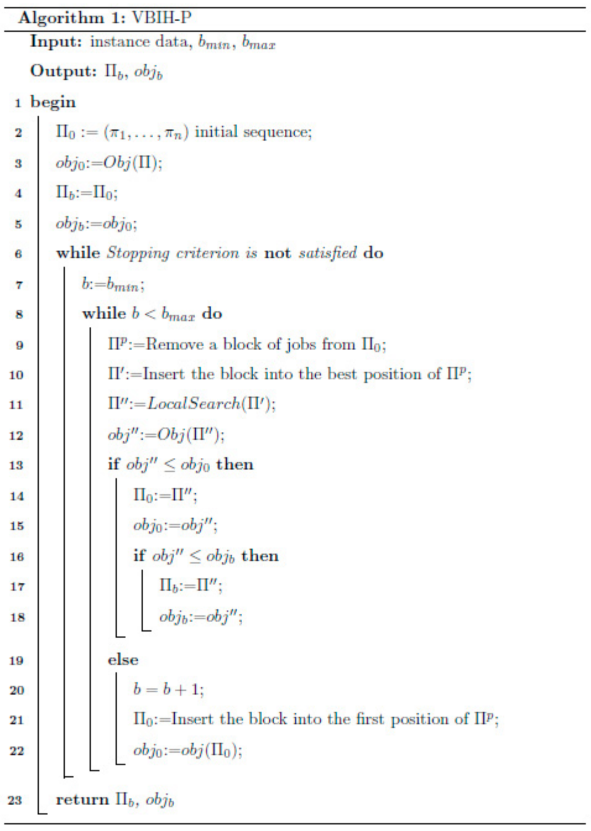
The original Iterated Greedy (IG) algorithm starts with an initial solution obtained by the NEH heuristic. In the destruction phase, d jobs are removed from the current sequence and stored in a partial sequence. In the construction phase, the d jobs are reinserted one by one in the best position. LS is applied to the complete solution. After LS, it should be decided whether or not the new sequence is accepted as the initial solution for the next iteration. A simple simulated annealing-like acceptance criterion with a constant temperature (parameter denoted TP) is employed.

The IGALL algorithm is similar to the IG, with the difference that the IGALL applies LS to each partial solutions after the destruction. It uses the same parameters d and TP.

The VBIH algorithm originally developed by Tasgetiren et al. (2016) for the PFSP with total completion time and blocking constraint, is similar to the IG and IGALL. Öztop et al. (2020a,b) consider the PFH-NEH as initial sequence. The difference is that the destruction procedure removes from the current solution a block of jobs, with an increasing size $b \in [bmin, bmax]$ in each iteration, applying LS. Afterwards, the block is inserted in all possible positions, choosing the best one. Again, LS is applied to the complete solution. The same simple simulated annealing-like acceptance criterion with parameter TP is employed.

For more detail of the original methods IG, IGALL and VBIH, the reader is referred to the cited papers.

The VBIH-P is the method proposed in this study (the pseudocode is presented in Figure 2). The VBIH-P follows the same procedure of the VBIH previously described with some differences: the local search applied to the partial



```
Algorithm 1: VBIH-P
Input: instance data, $b_{min}$, $b_{max}$
Output: $\Pi_b$, $obj_b$
begin
    $\Pi_0 := (\pi_1, \ldots, \pi_n)$ initial sequence;
    $obj_0 := Obj(\Pi)$;
    $\Pi_b := \Pi_0$;
    $obj_b := obj_0$;
    while Stopping criterion is not satisfied do
        $b := b_{min}$;
        while $b < b_{max}$ do
            $\Pi^p$ := Remove a block of jobs from $\Pi_0$;
            $\Pi'$ := Insert the block into the best position of $\Pi^p$;
            $\Pi'' := LocalSearch(\Pi')$;
            $obj'' := Obj(\Pi'')$;
            if $obj'' \leq obj_0$ then
                $\Pi_0 := \Pi''$;
                $obj_0 := obj''$;
                if $obj'' \leq obj_b$ then
                    $\Pi_b := \Pi''$;
                    $obj_b := obj''$;
            else
                $b = b + 1$;
                $\Pi_0$ := Insert the block into the first position of $\Pi^p$;
                $obj_0 := obj(\Pi_0)$;
    return $\Pi_b$, $obj_b$
```

Fig. 2. Pseudocode for the metaheuristic VBIH-P

sequence is not included, revealing a good performance of the destruction and construction for the TCIT objective. Additionally, the simulated annealing-like acceptance criterion is removed. In the proposal, if the new solution is better than the current one, the new solution is always accepted as the incumbent solution. Otherwise, the incumbent solution will become the last solution explored in the construction process. In our procedure, the last solution explored is the result of inserting the block in the first position of the partial sequence. Note that this metaheuristic only has $bmin$ and $bmax$ as parameters.

4. COMPUTATIONAL EVALUATION

4.1 Performance Measure and Testbed

To compare the methods, the Relative Deviation Index (RDI) has been selected as performance measure. It is computed by the following equation:

$$RDI_i = \frac{TCIT_{im} - TCIT_{imin}}{TCIT_{imax} - TCIT_{imin}} \quad (1)$$

with $TCIT_{im}$ the objective value for instance i obtained by method m, $TCIT_{imax}$ the maximum value obtained for instance i across all the methods, and $TCIT_{imin}$ the minimum.

For all the methods, the well known Taillard testbed Taillard (1993) has been used with the following sizes, being n the number of jobs, and m the number of machines : $n \times m \in \{20\times5, 20\times10, 20\times20, 50\times5, 50\times10, 50\times20, 100\times5, 100\times10, 100\times20, 200\times10, 200\times20, 500\times20\}$, and the processing times uniformly generated between 1 and 99. Each size has 10 instances, being in total 120 instances.

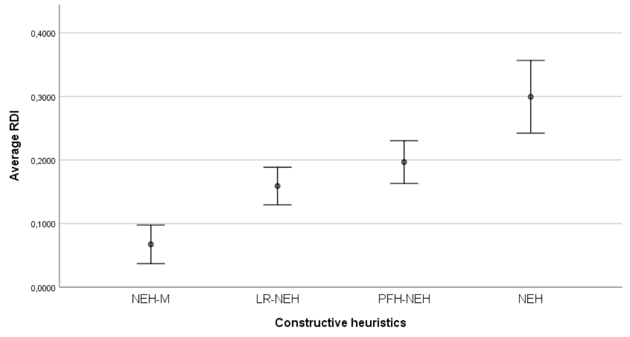


Fig. 3. 95% Confidence Intervals for Constructive Heuristics

4.2 Constructive heuristics

In this section, the constructive heuristics (NEH, NEH-M, LR-NEH and PFH-NEH) have been compared. The parameter X was calibrated and the best results were obtained for $X = n$, with n the number of jobs, when $n \leq 200$, or $X = 1$ otherwise. Table 1 shows the results in terms of ARDI (Average RDI) for each constructive heuristic, grouped by the values of n and m. The last row provides the total, showing that the best performance is obtained by the NEH-M. An analysis of the variance has been carried out, showing that NEH-M is significantly different to the rest of the methods by 95% confidence intervals. This can be observed in Figure 3 where the confidence intervals reveal the significance of the NEH-M compared to the other methods.

Table 1. Heuristics: ARDI

n	m	LR-NEH	NEH	NEH-M	PFH-NEH
20	5	0.1722	0.6945	0.3338	0.0831
	10	0.2054	0.4669	0.0652	0.2631
	20	0.2579	0.2491	0.0000	0.1429
50	5	0.0657	0.4770	0.1908	0.1640
	10	0.1343	0.3173	0.0405	0.1806
	20	0.2722	0.2379	0.0000	0.1838
100	5	0.1282	0.4876	0.0732	0.4174
	10	0.0744	0.3699	0.0643	0.1684
	20	0.2045	0.1104	0.0000	0.1640
200	10	0.1881	0.0766	0.0000	0.3019
	20	0.1700	0.0657	0.0000	0.1595
500	20	0.0356	0.0401	0.0401	0.1318
Total Average		0.1590	0.2994	0.0673	0.1967

4.3 Metaheuristics

Regarding the metaheuristics, IG, IGALL, VBIH and VBIH-P are compared. All the constructive heuristics have been tested as initial sequences. Due to space reasons, in this section we present only the results for the initial sequence providing the best results in the complete computational experiment, the NEH-M.

In this study, the parameters used are the following: IG and IGALL $d = 4$ and $TP = 0.4$, VBIH $bs \in [bmin = 2, bmax = 5]$ and $TP = 0.4$, and, finally, VBIH-P $bs \in$

Table 2. Metaheuristics: ARDI

n	m	IG	IGALL	VBIH	VBIH-P
20	**5**	0.4521	0.5974	0.3387	0.0091
	10	0.5740	0.6408	0.5203	0.0333
	20	0.4712	0.5838	0.4548	0.0243
50	**5**	0.3955	0.4933	0.3000	0.0000
	10	0.3774	0.5120	0.2789	0.1059
	20	0.4740	0.5209	0.3873	0.0529
100	**5**	0.1835	0.2407	0.1835	0.0333
	10	0.3823	0.4805	0.2830	0.0257
	20	0.3713	0.5370	0.2667	0.0798
200	**10**	0.3323	0.3309	0.2436	0.2012
	20	0.3956	0.4493	0.2896	0.1844
500	**20**	0.2930	0.6155	0.5902	0.4275
Total		0.3918	0.5002	0.3447	0.0981

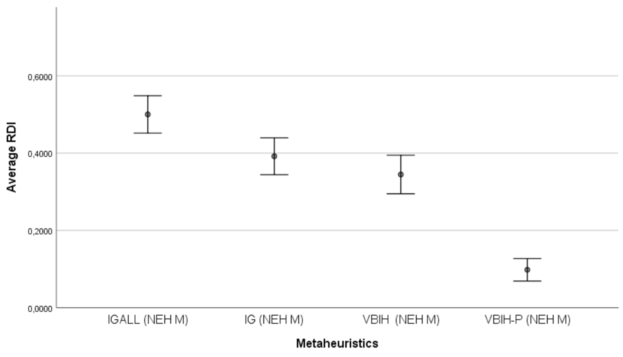


Fig. 4. 95% Confidence Intervals for Metaheuristics

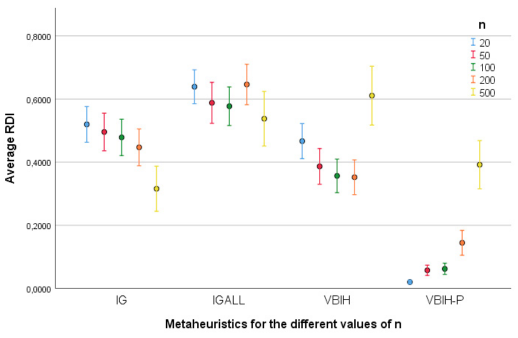


Fig. 5. 95% Confidence Intervals for Metaheuristics for different values of n

$[bmin = 2, bmax = 5]$. For all the methods, the stopping criterion has been $n \cdot m \cdot 30$ milliseconds. For each method, Table 2 provides the ARDI for each metaheuristic for the different values of n and m.

The results clearly indicate that the best method in terms of ARDI is the VBIH-P. In statistical analysis carried out shows that VBIH-P is better than IG, IGALL and VBIH. This can be observed in Figure 4, where the confidence intervals show the good performance of the VBIH, with statistical differences. However, it should be noticed that the VBIH-P gets worse as n increases, being outperformed by IG when $n = 500$ (see Table 2), but without statistical differences, as it can be seen in Figure 5, where the results are disaggregated by the number of jobs. Therefore, the results show that our proposal outperforms the other efficient metaheuristics for the problem.

5. CONCLUSION

In this paper, the permutation flowshop scheduling problem PFSP with total core idle time minimization is considered. Approximate methods have not been considered previously to solve this problem. Therefore, in this paper, the most efficient methods found in the related literature (PFSP with makespan and total completion time minimization) have been adapted. Additionally, a variant of one of the adapted metaheuristics is proposed. A computational experiment has been carried out, in order to determine the efficiency of the methods for the new objetive function. The results reveal a good performance of the classic constructive heuristics, being the best the NEH-M. Among the adapted metaheuristics, the VBIH shows the best performance, but among all the methods, the proposed VBIH-P, provides the best results with statistical differences compared to the rest of the adapted methods. As future research line, new methods including more intelligence of the problem can be proposed, and compared to the adapted methods in more extensive testbeds as the proposed by Vallada et al. (2015).

REFERENCES

Dubois-Lacoste, J., Pagnozzi, F., and Stützle, T. (2017). An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. *Computers and Operations Research*, 81, 160–166. doi:10.1016/j.cor.2016.12.021.

Fernandez-Viagas, V. and Framinan, J.M. (2019). A best-of-breed iterated greedy for the permutation flowshop scheduling problem with makespan objective. *Computers and Operations Research*, 112. doi: 10.1016/j.cor.2019.104767.

Fernandez-Viagas, V., Molina-Pariente, J.M., and Framinan, J.M. (2020). Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling. *European Journal of Operational Research*, 282(3), 858–872. doi:10.1016/j.ejor.2019.10.017.

Fernandez-Viagas, V., Ruiz, R., and Framinan, J.M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3), 707–721. doi: 10.1016/J.EJOR.2016.09.055.

Liu, J. and Reeves, C.R. (2001). Constructive and composite heuristic solutions to the $P||\sum C_i$ scheduling problem. *European Journal of Operational Research*, 132(2), 439–452. doi:10.1016/S0377-2217(00)00137-5.

Maassen, K., Perez-Gonzalez, P., and Günther, L.C. (2020). Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling. *Computers & Operations Research*, 104965. doi:10.1016/j.cor.2020.104965.

McCormick, S., Pinedo, M.L., Shenker, S., and Wolf, B. (1989). Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*.

Nawaz, M., Enscore, E., and Ham, I. (1982). A Heuristic Algorithm for the m-Machine , n-Job Flow-shop Sequencing Problem. *Omega*, 11(1), 91–95. doi: 10.1016/0305-0483(83)90088-9.

Öztop, H., Tasgetiren, M.F., Eliiyi, D.T., Pan, Q.K., and Kandiller, L. (2020a). An energy-efficient permutation flowshop scheduling problem. *Expert Systems with Applications*, 150, 113279. doi:10.1016/j.eswa.2020.113279.

Öztop, H., Tasgetiren, M.F., Kandiller, L., Eliiyi, D.T., and Gao, L. (2020b). Ensemble of metaheuristics for energy-efficient hybrid flowshops: Makespan versus total energy consumption. *Swarm and Evolutionary Computation*, 54(September 2019). doi: 10.1016/j.swevo.2020.100660.

Pan, Q.K. and Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research*, 40(1), 117–128. doi:10.1016/J.COR.2012.05.018.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3), 2033–2049. doi: 10.1016/j.ejor.2005.12.009.

Taillard, E.D. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64, 278–285.

Tasgetiren, M.F., Pan, Q.K., Kizilay, D., and Gao, K. (2016). A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion. *Algorithms*, 9(4), 71. doi: 10.3390/A9040071.

Vallada, E., Ruiz, R., and Framinan, J.M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3), 666–677. doi:10.1016/j.ejor.2014.07.033. URL `http://dx.doi.org/10.1016/j.ejor.2014.07.033`.