Total Tardiness Minimization in a Flow Shop with Blocking Using an Iterated Greedy Algorithm

Nouri Nouha and Ladhari Talel

Abstract We highlight in this paper the competitive performance of the Iterated Greedy algorithm (IG) for solving the flow shop problem under blocking. A new instance of IG is used to minimize the total tardiness criterion. Basically, due to the NP-hardness of this blocking problem, we employ another variant of the NEH heuristic to form primary solution. Subsequently, we apply recurrently constructive methods to some fixed solution and then we use an acceptance criterion to decide whether the new generated solution substitutes the old one. Indeed, the perturbation of an incumbent solution is done by means of the destruction and construction phases. Despite its simplicity, the IG algorithm under blocking has shown its effectiveness, based on Ronconi and Henriques benchmark, when compared to state-of-the-art meta-heuristics.

Keywords Blocking · Flow shop · Total tardiness · IG

1 Problem Definition

We are engaged in this research with generating an efficient optimization technique for the subsequent scheduling problem. Each of N jobs from the job set $J = 1; 2; \ldots; N$ has to be processed on m consecutive machines from the machine set $M = (j = 1, 2, \ldots, m)$ during a p_{ij} time units. The process of each job on each machine is exactly the same and once the process is started it may not be broken down. A job i can have a given due date D_i corresponding to the perfect time that it

N. Nouha (✉)
Ecole Supérieure des Sciences Economiques et Commerciales de Tunis,
University of Tunis, Tunis, Tunisia
e-mail: nouri.nouha@yahoo.fr

L. Talel
College of Business, Umm Al-Qura University, Mecca, Saudi Arabia
e-mail: talel_ladhari2004@yahoo.fr

R. Silhavy et al. (eds.), *Artificial Intelligence Perspectives in Intelligent Systems*,
Advances in Intelligent Systems and Computing 464,
DOI 10.1007/978-3-319-33625-1_9

should be completed. Besides, each job can be processed only on one machine at a time and each machine can process at most one job at a time. No passing is allowed.

Considering the above assumptions, we are facing some blocking constraints: buffers between consecutive pair of machines are stated with zero capacity. That is a current machine j may be blocked by the job it has processed if the next machine is not discharged. This environment is completely different from the No-wait Flow Shop setting where when a job is started on the first machine, it must be constantly processed till its achievement on the last machine without interruption.

This research deals with the Blocking Flow Shop Scheduling Problem (BFSP) to minimize the total tardiness of jobs, denoted as $Fm|block|\sum T_j$ according to the notation proposed in [1]. This variant of flow shop problems has important impact in manufacturing systems since when a job is not finished by its due date then supplementary costs are incurred.

The tardiness is defined as the maximum time between zero and the lateness of a job settled as the difference between the completion time of a job and its fixed due date. In general, the $Fm|block|C_{max}$ is strongly NP-hard [2]. This is an immediate consequence of the NP-hardness of the $Fm||C_{max}$. The case of two machines ($m = 2$) may be easily solved using Gilmore and Gomory's scheme [3].

Formally, the BFSP may be formulated using the following equations [4], where $C_{\pi_i,M} = d_{\pi_i,M}$ is the completion time of job π_i on machine M, $d_{\pi_i,j}$ ($i = 1, 2, \ldots, N; j = 0, 1, 2, \ldots, M$) represents the departure time of job π_i on machine j, and $\Pi := (\pi_1, \pi_2, \ldots, \pi_N)$ is a solution for the problem.

$$d_{\pi_1,0} = 0 \tag{1}$$

$$d_{\pi_1,j} = \sum_{k=1}^{j} p_{\pi_{1,k}} \quad j = 1, 2, \ldots, M-1 \tag{2}$$

$$d_{\pi_i,0} = d_{\pi_{i-1},1} \quad i = 2, \ldots, N \tag{3}$$

$$d_{\pi_i,j} = max\{d_{\pi_i,j-1} + p_{\pi_i,j}, d_{\pi_{i-1},j+1}\} \quad i = 2, \ldots, N; j = 1, 2, \ldots, M-1 \tag{4}$$

$$d_{\pi_i,M} = d_{\pi_i,M-1} + p_{\pi_i,M} \quad i = 1, 2, \ldots, N \tag{5}$$

$$TT(\Pi) = \sum_{i=1}^{n} (max\{0, (C_{\pi_i} - D_i\}) \tag{6}$$

The literature regarding a BFSP is not extensive. Indeed, the tardiness criterion has been studied fewer than the makespan and total flow time criteria. We rapidly review the related literature.

As a constructive heuristics, we refer to the Profile Fitting (PF) [5], the Nawaz-Enscore-Ham heuristic (NEH) [6], the MinMax (MM) and combination of MM and NEH (MME) and combination of PF and NEH (PFE) [7] techniques. In [8, 9] the NEH-WPT heuristic and a constructive and a GRASP-based heuristics for the BFSP were introduced, respectively.

Concerning meta-heuristics, we refer to the Genetic Algorithm (GA) proposed in [10], the (Ron) algorithm developed in [11], and the Tabu Search (TS) and the enhanced TS techniques used in [12]. Meanwhile, we cite the Hybrid Discrete Differential Evolution (HDDE) algorithm introduced in [13] which was compared to the Hybrid Differential Evolution (HDE) algorithm developed in [14], and the Iterated Greedy (IG) method in [15].

Thus far, it was proven that RON, HDDE, and IG algorithms give competitive results and may be considered as top techniques for the BFSP under makespan.

Now, under the total flow time criterion, we refer to the hybrid modified global-best Harmony Search (hmgHS) algorithm and the Discrete Artificial Bee Colony algorithm (DABC_D) technique presented in [8, 16], respectively, and the Greedy Randomized Adaptive Search Procedures (GRASP) hybridized with the Variable Neighbourhood Search (VNS) technique in [17].

Besides, we cite the effective hybrid discrete artificial bee colony algorithms proposed in [18], and the Revised Artificial Immune Systems (RAIS) technique in [19]. A three-phase algorithm under C_{max} is presented in [20] and a Discrete Particle Swarm Optimization algorithm with self-adaptive diversity control was treated in [21].

Also, we cite the Memetic Algorithm (MA) in [22], the chaos-induced discrete self organizing migrating algorithm in [23], the Iterated Local Search algorithm (ILS) coupled with a Variable Neighbourhood Search (VNS) in [24], and the Blocking Genetic Algorithm (BGA) and Blocking Artificial Bee Colony (BABC) algorithms in [25].

Under tardiness measure, few papers were found. Basically, the Tabu Search method proposed in [26], the NEH-based method called (FPDNEH) and the Greedy Randomized Adaptive Search Procedure(GRASP) developed in [9], and the Iterated Local Search algorithm (ILS) hybridized with the Variable Neighbourhood Search (VNS) technique in [24].

Therefore, it is interesting to intensify research to develop simple algorithms which are easy to adapt and implement in practical applications. In this paper we propose an Iterated Greedy algorithm (IG) to minimize the tardiness of scheduled jobs in a flow shop environment with blocking. We restrict our attention solely to permutation schedules.

Following this brief definition, the paper is structured as follows. The IG algorithm under blocking is explained in Sect. 2. Section 3 presents the numerical experiments, and Sect. 4 summarizes the conclusions.

2 Iterated Greedy Algorithm for the Blocking Flow Shop Problem

The iterated greedy algorithm is very well adapted to solve combinatory problems, and especially various flow shop instances [27]. In an iterative way, the greedy method uses constructive techniques to generate new solution based on some other

fixed solution, and then decides if it will be accepted and replace the old one based on some acceptance criterion.

The destruction and construction stages are employed to obtain a sequence of solution. The destruction stage deletes some elements from the designated solution. After that, a new sequence is obtained by reforming a whole solution based on constructive heuristic, which rearranges the removed elements in some order to form a final sequence. Optionally, the obtained sequence (final) may be subjected to local search stage for enhancement.

One important point to note is that IG is closely related to the Iterated Local Search (ILS). The difference between them is that in IG the perturbation of an incumbent solution is done by means of the destruction and construction phases, whereas in ILS the perturbation is done just for escaping from local optimum.

Now, details of the algorithms are presented below.

2.1 Seed Sequence

To yield the initial sequence, the PF-NEH(x) heuristic proposed in [25] has been used. Nevertheless, instead of generating x solutions at the end of the heuristic, we select only the permutation with the minimum tardiness value. With a probability P_{ls}, we have also used a local search technique based on the insertion operator to generate neighboring solution. A fixed job is removed from its first position and reinserted in all feasible places. Then, the new obtained sequence is recorded only when there is an enhancement in the objective value. The final permutation Π^s thus generated forms the seed sequence.

2.2 Destruction and Construction Phases

The destruction stage is started based on the initial seed sequence generated as explained earlier. Iteratively, the current stage starts with a whole sequence Π^s and then removes $[q * \Pi^s]$ randomly jobs from Π^s. The degree of destruction q is drawn in the range [0,1]. This yields two sub-solutions: the first one models the removed jobs Π^r, and the second represents the rest of the sequence after deleting jobs Π^s.

In the construction stage, a final solution Π^c is then reformed by replacing the previously extracted jobs in the order in which they were removed.

The procedures of the destruction and construction stages are as in Tables 1 and 2.

Table 1 Procedure destruction stage (Π^s, q)

Begin
Stage 1: Set Π^r empty
Stage 2: Let $\Pi^q \leftarrow \Pi^s$
Stage 3: ***For*** $i = 1$ ***to*** $(q * \lvert\Pi^q\rvert)$ ***Do***
$\Pi^q \leftarrow$ Remove a randomly selected job from Π^q
$\Pi^r \leftarrow$ Include the removed job in Π^r
End

Table 2 Procedure construction stage (Π^q, Π^r)

Begin
Stage 1: Let $\Pi^c \leftarrow \Pi^q$
Stage 2: ***For*** $j = 1$ ***to*** $\lvert\Pi^r\rvert$ ***Do***
$\Pi^c \leftarrow$ Best permutation obtained after inserting job π_j^r in all possible positions of Π^c
End

2.3 *Acceptance Measure and Final IG Algorithm Under Blocking*

An acceptance measure is applied to decide wether the generated sequence will be accepted or not. As in [27, 28], we have used the Simulated Annealing (SA) acceptance measure to approve 'bad' solutions with some fixed probability.

This acceptance criterion is employed with a certain temperature value depending on the number of jobs, machines, and on other tractable parameter λ:

$$Tempt = \lambda * \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} p_{ij}}{10 * M * N} \tag{7}$$

Let $TT(\Pi^s)$ and $TT(\Pi^c)$ be respectively the total tardiness values of the current incumbent solution and the new reconstructed solution. Also, let rand() be a function returning a random number sampled from a uniform distribution between 0 and 1.

If $TT(\Pi^c) \geq TT(\Pi^s)$, Then Π^c is accepted as the new incumbent solution if:

$$rand() \leq exp\{TT(\Pi^c) - TT(\Pi^s)/Tempt\} \tag{8}$$

Considering all previous subsections, the proposed IG algorithm under blocking goes as in Table 3.

Table 3 IG algorithm under blocking

Begin
Stage 1: Set the parameters: P_{ls}, q, λ and MCN.
Stage 2: Obtain the initial sequence using the PF-NEH(x) heuristic. Depending on the local probability rate P_{ls}, improve this solution using the local search technique. Let the final permutation Π^s be the seed sequence.
Stage 3: Let $\Pi^* = \Pi^s$
Stage 4:
While termination condition is not met ***Do***
• Π^q = Destruction-phase (Π^s, q)
• Π^c = Construction-phase (Π^q)
• $\Pi^{c'}$ = Local-phase (Π^c, P_{ls})
• If $TT(\Pi^{c'}) < TT(\Pi^s)$ Then
– $\Pi^s := \Pi^{c'}$
– If $TT(\Pi^s) < TT(\Pi^*)$ Then
∗ $\Pi^* := \Pi^s$
• Else If $(rand() \leq exp\{TT(\Pi^s) - TT(\Pi^{c'})/Tempt\})$ Then
– $\Pi^s := \Pi^{c'}$
Stage 5: Return the best solution found Π^*
End

3 Numerical Experiments

This section focuses on computational experience with the proposed IG algorithm under blocking. Its performance obtained by comparing the resulting solutions (total tardiness) with respect to one competitive algorithm from the literature was investigated. We performed experiments on a well-known set of benchmark instances [9]. These instances are composed of 5 groups which are a combination of 20, 50, 100, 200 and 500 jobs with 5, 10 and 20 machines. The processing times of jobs and due dates are uniformly distributed between [29, 99] and $P(1 - T - R/2)$ and $P(1 - T + R/2)$, respectively. T is the tardiness factor, R is the due-date range [30], and P is a valid LB [31].

As well, the Relative Percentage Deviation (RPD) is numbered once an instance is launched (with 10 replications) and calculated according to the following recursion. TT^A is the tardiness value obtained using the proposed technique and TT^{Min} represents the minimum tardiness value obtained among the two compared algorithms.

$$RPD(A) = \frac{(TT^A - TT^{Min}) \times 100}{TT^{Min}} \tag{9}$$

The IG algorithm under blocking is coded in Visual C++ and run on an Intel Pentium IV 2.4 GHz PC with 512 MB of memory. Our technique establishes a relatively simple mechanism where there are basically four parameters which must be properly established. These are the MCN, P_{ls}, λ, and q. We analyzed these parameters using one generated instance with $n = 150$ and $m = 10$. The due dates of the jobs were generated following the TWK rule [32]. We fix this size since it represents a challenge given the number of jobs and machines that must be satisfied (large instance). For each analysis, we vary only the parameter of interest to study its impact on the final solution and on the convergence rate of the IG. After extensive testing, parameters were set to the following values: $MCN = 100$, $P_{ls} = 0.2$, $\lambda = 2$, and $q = 3$.

3.1 Evaluation of the IG Algorithm Under Blocking

To prove the good quality of our presented IG for the BFSP, we wanted to compare our technique to other meta-heuristics that are already used in the literature for dealing with our problem. Our IG was compared against the obtained results by the GA algorithm based on the path relinking technique (GA_PR) in [33]. This technique has shown higher efficiencies in solving benchmark for large scale instances. Therefore, this technique was selected for comparison with the IG under blocking. In fact, Table 4 summarizes the computational results of the two compared techniques for all combination of jobs and machines, and where the total tardiness solutions were averaged (comparisons were made based on the ARPD metric).

By analyzing Table 4, it can be seen that IG algorithm presents better average results than the *GA_PR* in the different scenarios. For all test instances, with $N * M$ varying from (20*5) up to (500*20), the greedy technique enhanced all results. The tested algorithm outperforms the *GA_PR* algorithm in 97 % of the classes, and considering the number of superior results, our method outperformed the *GA_PR* in 398 of the 480 test-problems.

4 Final Remarks

Different from other sophisticated techniques, this greedy algorithm has few parameters to be adjusted, which makes it more simpler to be implemented and used to solve the BFSP. Also, the IG under blocking presented a significant improvement in all test instances. Its superiority against GA_PR algorithm should be attributed to the smart combination of greedy stages (destruction and construction), local search, as well as to the use of new version of the PF-NEH heuristic. Future work involving the tardiness criterion could include designing another procedure for comparing algorithms. We also expect to apply the IG for multi-objective scheduling problems.

Table 4 ARPD values under total tardiness measure

N	M	Scenario 1		Scenario 2		Scenario 3		Scenario 4		All Scenario	
		IG (%)	GA_PR (%)	IG (%)	GA_PR (%)	IG (%)	GA_PR (%)	IG (%)	GA_PR (%)	IG (%)	GA_PR (%)
20	5	0,000	3,596	0,000	1,083	0,000	1,037	0,000	0,861	0,000	1,644
20	10	0,000	1,415	0,000	1,190	0,000	0,537	0,000	0,594	0,000	0,934
20	20	0,000	0,461	0,000	0,374	0,000	0,195	0,000	0,291	0,000	0,330
50	5	0,000	4,743	0,000	2,620	0,000	1,096	0,000	0,920	0,000	2,345
50	10	0,000	5,888	0,000	9,010	0,000	1,443	0,000	2,716	0,000	4,764
50	20	0,000	4,500	0,000	2,899	0,000	1,064	0,000	0,914	0,000	2,344
100	5	0,000	5,140	0,000	0,625	0,000	1,561	0,000	3,985	0,000	2,828
100	10	0,000	4,430	0,000	11,103	0,000	1,145	0,000	1,373	0,000	4,513
100	20	0,000	8,475	0,000	17,152	0,000	1,781	0,000	1,910	0,000	7,330
200	10	0,000	11,678	0,000	20,482	0,000	7,304	0,000	9,915	0,000	12,345
200	20	0,000	13,835	0,000	25,910	0,000	6,763	0,000	5,895	0,000	13,101
500	20	0,000	18,072	0,000	63,177	0,000	16,799	0,000	34,073	0,000	33,030
Average		0,000	6,853	0,000	12,969	0,000	3,394	0,000	5,287	0,00	7,13

References

1. Graham, R., Lawler, E., Lenstra, J., Rinnooy, K.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discret. Math. **5**, 287–362 (1979)
2. Hall, N., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Oper. Res. **44**, 25–510 (1996)
3. Gilmore, P., Gomory, R.: Sequencing a one state variable machine: a solvable case of the traveling salesman problem. Oper. Res. **5**, 79–655 (1964)
4. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Pretice Hall, U.A.S (2008)
5. McCormick, S., Pinedo, M., Shenker, S., Wolf, B.: Sequencing in an assembly line with blocking to minimize cycle time. OR **37**, 925–935 (1989)
6. Nawaz, M., Enscore, J., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega **11**, 91–95 (1983)
7. Ronconi, D.: A note on constructive heuristics for the flowshop problem with blocking. Int. J. Prod. Econ. **87**, 39–48 (2004)
8. Wang, L., Pan, Q., Tasgetiren, M.: Minimizing the total flow time in a flowshop with blocking by using hybrid harmony search algorithms. Expert Syst. Appl. **12**, 7929–7936 (2010)
9. Ronconi, D.P., Henriques, L.R.S.: Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. Omega **2**, 272–81 (2009)
10. Caraffa, V., Ianes, S., Bagchi, T.P., Sriskandarajah, C.: Minimizing makespan in a flowshop using genetic algorithms. Int. J. Prod. Econ. **2**, 15–101 (2001)
11. Ronconi, D.: A branch-and-bound algorithm to minimize the makespan in a flowshop problem with blocking. Ann. Oper. Res. **1**, 53–65 (2005)
12. Grabowski, J., Pempera, J.: The permutation flowshop problem with blocking. A tabu search approach. Omega **3**, 11–302 (2007)
13. Wang, L., Pan, Q., Suganthan, P., Wang, W., Wang, Y.: A novel hybrid discrete differential evolution algorithm for blocking flowshop scheduling problems. Comput. Oper. Res. **3**, 20–509 (2010)
14. Qian, B., Wang, L., Huang, D.X., Wang, W.L., Wang, X.: An effective hybrid DE-based algorithm for multi-objective flowshop scheduling with limited buffers. Comput. Oper. Res. **1**, 3–209 (2009)
15. Ribas, I., Companys, R., Tort-Martorell, X.: An iterated greedy algorithm for the flowhsop scheduling problem with blocking. Omega **3**, 293–301 (2011)
16. Deng, G., XU, Z., Gu, X.: A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling. Chin. J. Chem. Eng. **20**, 1067–1073 (2012)
17. Ribas, N., Companys, R.: Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. Comput. Ind. Eng. (2015). doi:10.1016/j.cie.2015.04.013
18. Han, Y.-Y., Liang, J., Pan, Q.-K., Li, J.-Q., Sang, H.-Y., Cao, N.: Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem. Int. J. Adv. Manuf. Technol. **67**, 397–414 (2013)
19. Lin, S., Ying, K.: Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega **41**, 383–389 (2013)
20. Pan, Q., Wang, L.: Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. Omega **2**, 218–29 (2012)
21. Wang, X., Tang, L.: A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. Appl. Soft Comput. **12**, 652–662 (2012)
22. Pan, Q., Wang, L., Sang, H., Li, J., Liu, M.: A high performing memetic algorithm for the flowshop scheduling problem with blocking. IEEE Trans. Autom. Sci. Eng. **10**, 741–756 (2013)
23. Davendra, D., Bialic-Davendra, M., Senkerik, R., Pluhacek, M.: Scheduling the flow shop with blocking problem with the chaos-induced discrete self organising migrating algorithm. In: Proceedings 27th European Conference on Modelling and Simulation (2013)

24. Ribas, I., Companys, R., Tort-Martorell, X.: An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. Int. J. Prod. Res. **51**, 5238–5252 (2013)
25. Nouri, N., Ladhari, T.: Minimizing regular objectives for blocking permutation flow shop scheduling: heuristic approaches. GECCO, 441–448 (2015)
26. Armentano, V.A., Ronconi, D.P.: Minimizaçâo do tempo total de atraso no problema de flow-shop com buffer zero através de busca tabu. Gestao Produçao **7**(3), 352–363 (2000)
27. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **177**, 49–2033 (2007)
28. Ruiz, R., Stützle, T.: An iterated greedy heuristic for the sequence dependent setup times flow-shop problem with makespan and weighted tardiness objectives. Eur. J. Oper. Res. **187**, 1143–1159 (2008)
29. Johnson, S.M.: Optimal two- and three-stage production schedules with setup time included. Naval Res. Logist. Q. **1**, 8–61 (2013)
30. Potts, C.N., Van Wassenhove, L.N.: A decomposition algorithm for the single machine total tardiness problem. Oper. Res. Lett. **1**, 81–177 (1982)
31. Taillard, E.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. **64**, 85–278 (1993)
32. Baker, K.R., Bertrand, J.W.M.: An investigation of due date assignment rules with constrained tightness. J. Oper. Manag. **3**, 109–120 (1984)
33. Januario, T.O., Arroyo, J.E.C., Moreira, M.C.O.: Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), pp. 153–164. Springer, Berlin (2009)