Scheduling of Parallel Print Machines with Sequence-Dependent Setup Costs: A Real-World Case Study

Manuel Iori[1], Alberto Locatelli[1](✉), and Marco Locatelli[2]

[1] Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy
{manuel.iori,alberto.locatelli}@unimore.it

[2] Department of Engineering and Architecture, University of Parma, Parma, Italy
marco.locatelli@unipr.it

Abstract. In the present work, we consider a real-world scheduling problem arising in the color printing industry. The problem consists in assigning print jobs to a heterogeneous set of flexographic printer machines, as well as in finding a processing sequence for the sets of jobs assigned to each printer. The aim is to minimize a weighted sum of total weighted tardiness and total setup times. The machines are characterized by a limited sequence of color groups and can equip additional components (e.g., embossing rollers and perforating rolls) to process jobs that require specific treatments. The process to equip a machine with an additional component or to clean a color group takes a long time, with the effect of significantly raising the setup costs. Furthermore, the time required to clean a color group between two different jobs depends directly on the involved colors. To tackle the problem, we propose a constructive heuristic followed by some local search procedures that are used one after the other in an iterative way. Extensive tests on real-world instances prove that the proposed algorithm can obtain very good-quality solutions within a limited computing time.

Keywords: Parallel machine scheduling · Flexographic printing process · Sequence-dependent setup costs · Food packaging industry

1 Introduction

Over the past years, the food packaging has undergone a remarkable evolution. The choice of materials and graphic design for the food packages has become more and more important to capture and establish a strong and lasting bond

We acknowledge financial support from Istituto Stampa s.r.l. (Italy). The work originates from the daily activity of Istituto Stampa s.r.l., a company whose headquarters is located in Reggio Emilia (Italy). The company operates in the field of packaging industry by producing and printing packaging materials for food products since 1933.

A. Dolgui et al. (Eds.): APMS 2021, IFIP AICT 631, pp. 637–645, 2021.
https://doi.org/10.1007/978-3-030-85902-2_68

with the customers (see, e.g., Paine and Paine [5]). As a result, the color printing industry has had to adapt to the increasingly demanding requirements from the food industry and its costumers. Flexography and rotogravure printing are the most used technologies for printing flexible food packages. Rotogravure printing used to guarantee a better quality, but now flexography has reached a comparable quality. In addition, flexography is less expensive, faster, and allows for printing on almost any material such as paper, plastic, aluminum foil, and cellophane.

A flexographic printer machine is characterized by a limited sequence of color groups, by a set of additional components (e.g., embossing rollers and perforating rolls) that can be mounted to process jobs that require specific treatments, and by its washing system that can be manual or automatic.

As reported by Schuurman and Van Vuuren [7], the printing process of a printing job consists of two phases: the *machine setup phase* and the *printing phase*. The first phase consists in preparing the machine for printing a new job and may require to remove material from the machine, to equip the machine with an additional component, and to wash and refill color groups. If two consecutively scheduled printing jobs require significantly different color overlays, then substantial down times are incurred. The process to wash a color group and to refill it with another ink takes a long time and depends directly on the specific colors involved. The time required for the *machine setup phase* is job dependent, machine dependent, and job sequence-dependent. Indeed, it depends on the colors and additional components required by the new printing job, on the colors and additional components available in the printing machine and required by the previous printing jobs, and by the washing system type of the machine. On the other hand, the printing phase is just job dependent and machine dependent. More specifically, it depends on print volumes and complexity of the job, on the type of printed material, and on the printing speed of the machine.

Motivated by a real-world application, we consider a scheduling problem consisting in assigning printing jobs to a heterogeneous set of parallel flexographic printer machines, as well as in finding a processing sequence for the sets of jobs assigned to each printer. In accordance with the company, the objective is to minimize the weighted sum of total weighted tardiness and total setup times. We refer to this scheduling problem as the *Parallel Print Machine with Setup Costs Problem* (PMSCP). The PMSCP is a variant of the unrelated parallel machines with sequence-dependent setup times (see Pinedo [6]), in which the optimization criterion is the minimization of a weighted sum of total weighted tardiness and total setup times. Using the three-field notation for scheduling problems by Graham et al. [2], the PMSCP can be denoted as the $R|ST_{sd}|\alpha WT + \beta TST$ (see Allahverdi [1]), where WT denotes the total weighted tardiness, TST the total setup time, and α and β are two input parameters. The PMSCP is strongly $\mathcal{NP}$-hard since it is a generalization of the weighted tardiness scheduling problem (see Lenstra et al. [3]). In this work, we solve the problem by means of a constructive heuristic followed by some local search procedures. The resulting algorithm is capable of finding good-quality solutions within a limited computing time on several instances derived from the real-world case study.

The remainder of the paper is organized as follows. In Sect. 2, we formally describe the PMSCP. In Sect. 3, we present the heuristic algorithm. In Sect. 4, we provide the outcome of our computational experiments. Finally, in Sect. 5, we draw some concluding remarks.

2 Problem Description

In this section, the PMSCP is formally described. The following sets are defined:

- $M = \{1, \dots, m\}$: set of printing machines;
- $J = \{1, \dots, n\}$: set of printing jobs;
- C: set of colors available;
- A: set of additional components (i.e., embossing rollers and perforating rolls) available to the printing company.

A printing job $j \in J$ is characterized by the following elements:

- $M_j \subseteq M$: subset of machines capable of printing j;
- n_j: number of colors of j;
- $C_j = \{c_1, \dots, c_{n_j}\} \subseteq C$: subset of colors required by j;
- $A_j \subseteq A$: subset of additional components required by j;
- v_j: length of j, measured in linear meters;
- d_j: due date of j;
- w_j: tardiness penalty assigned to j for each day of tardiness.

A flexographic printing machine $i \in M$ is characterized by:

- $J_i \subseteq J$: subset of jobs that can be printed by machine i;
- θ_i: average printing speed of machine i;
- p_{ij}: processing time of job j on machine i;
- g_i: number of color groups of machine i;
- ℓ_i: washing system type of machine i, automatic ($\ell_i = 1$) or manual ($\ell_i = 0$).

The state of machine $i \in M$, after having finished job $j \in J$, is characterized by:

- $A^i_j \subseteq A$: subset of additional components installed on i;
- d^i: availability date of machine i, i.e., the moment in which the machine will finish processing j.

The state of a machine i depends not only on the last processed job j, but also on all the jobs processed by i before j. For instance, if the job processed before j requires an additional component $a \in A$, whereas j does not require it, then the component might be simply left on the machine. If, instead, j requires a different component a', then a must be switched with a'. For this reason, the state of machine i, after having finished job j, has a great impact on s^i_{jk}, which is the setup time on machine $i \in M$, when processing job $k \in J$ after having finished job $j \in J$, and which might also depend on the sequence of jobs previously assigned to the machine. This complicates the computation of a heuristic solution and makes the PMSCP even more challenging to solve in practice.

Each machine $i \in M$ is available from Monday to Friday for 8 working hours per day. It can be simply paused at the end of each work shift and resumed the day after without any penalty cost (even during the *printing phase*).

A sequence of jobs $S^i = (j^i_1, \ldots, j^i_{n_i})$ is feasible for machine $i \in M$ if $i \in M_j$ $\forall j \in S^i$. A feasible sequence S^i of jobs defines a schedule for machine $i \in M$ and can be used to compute start and completion times of all jobs. Let the T_j denote the number of days of tardiness of a job j in S^i. Because of the sequence-dependent setups based on the colors and on the additional components, even computing T_j is NP-hard (see, Meunier and Neveu [4], for a similar problem). In our work, we decided to adopt a simple but quick and good-enough evaluation of the setup costs (and hence of the tardiness), as described below in the next section.

A set of feasible sequences $S = \{S^i : i \in M\}$ is a feasible schedule for J if it forms a partition of J. The PMSCP consists in finding a feasible schedule that minimizes $f = \alpha WT + \beta TST$, where $WT = \sum_{i \in M} \sum_{j \in S^i} w_j T_j$ and $TST = \sum_{i \in M} \sum_{h=1}^{n_i - 1} s^i_{j^i_h j^i_{h+1}}$. With a slight abuse of notation, given a feasible sequence of jobs $S^i = (j^i_1, \ldots, j^i_{n_i})$ for machine $i \in M$, we let $f(S^i)$ denote the value $\alpha \cdot \sum_{j \in S^i} w_j T_j + \beta \cdot \sum_{h=1}^{n_i - 1} s^i_{j^i_h j^i_{h+1}}$.

3 Heuristic Approach

Given a machine $i \in M$ and two jobs $j, k \in J_i$, the setup cost s^i_{jk} is evaluated as follows. A color washing time is added to s^i_{jk} for each color $c \in C_k \backslash C_j$ (in the case $\ell_i = 0$) or if $|C_k \backslash C_j| \geq 1$ (in the case $\ell_i = 1$). Indeed, if a printer machine equips an automatic washing systems, the color groups can be washed in parallel, otherwise they have to be washed one by one. Moreover, if there is a $c \in C_k \backslash C_j$ corresponding to white or a special varnish, an additional penalty time is considered. Indeed, the process to wash a color group and to refill it with white ink or a special varnish requires complete cleaning of the corresponding color group with the effect of a significantly raising of the setup costs. Finally, for each additional component $a \in A_k \backslash A^i_j$ (i.e., set of additional components required by k that are not mounted in i after the finish of job j), a component changing time is added to s^i_{jk}. On the other hand, the time required by machine i to print job $j \in J_i$ is calculated by multiplying v_j by θ_i. Given a feasible sequence of jobs S^i, the time required for the *machine setup phase* and the *printing phase* are calculated, for each job $j \in S^i$ as described above. Thus, since the working time of the machines is known and the jobs in S^i are performed one after another, start and completion times of all jobs in S^i can be directly calculated.

A constructive greedy heuristic algorithm (CGHA) to solve the PMSCP is presented in Algorithm 1. The proposed method assigns, at each iteration, a print job to a first available machine, generating, for each machine $i \in M$, a feasible sequence of jobs S^i. From Step 5 to Step 11, a set L_i of jobs is created by selecting from J_i the jobs behind schedule, i.e., $\{j \in J_i : d_j < d^i\}$. If there is no job behind schedule, L_i is created selecting from J_i the feasible jobs that can only be printed by machine i, i.e., $\{j \in J_i : |M_j| = 1\}$. If L_i is still empty, then we set $L_i = J_i$. At Step 13, the greedy phase of the algorithm adds to S^i the most convenient job $j \in L_i$. At step 16, the state of machine i is updated. At Step 18, the obtained output $S = \{S^i : i \in M\}$ is a feasible schedule for J by construction.

```
1  input: $M = \{1, \dots, m\}$, $J = \{1, \dots, n\}$;
2  $S^i = \emptyset$ $(i \in M)$;
3  while $J \neq \emptyset$ do
4      select a machine $i \in M$ such that $d^i = \min_{i' \in M}\{d^{i'} : J_{i'} \neq \emptyset\}$ and $J_i \neq \emptyset$;
5      $L_i := \{j \in J_i : d_j < d^i\}$;
6      if $(L_i = \emptyset)$ then
7          $L_i = \{j \in J_i : |M_j| = 1\}$;
8      end
9      if $(L_i = \emptyset)$ then
10         $L_i = J_i$;
11     end
12     let $S^i_j$ be the sequence obtained adding job $j$ at the end of sequence $S^i$;
13     select a job $j \in L_i$ such that $f(S^i_j) = \min_{j' \in L_i} f(S^i_{j'})$;
14     add $j$ at the end of sequence $S^i$;
15     remove job $j$ from $J$;
16     update $d^i, A^i_j$;
17 end
18 return $S = \{S^i : i \in M\}$.
```

Algorithm 1: Constructive greedy heuristic algorithm (CGHA)

In order to search for improved solutions, three local search procedures, namely LS1, LS2, and LS3, are applied to the initial solution S obtained by the CGHA.

LS1 is based on an intra-machine swap neighborhood, which selects a machine $i \in M$, two job positions h and g $(1 \leq h < g \leq n_i)$ in the sequence of jobs S^i, and then generates a new sequence $S^i_{(h,g)} = (j^i_1, \dots, j^i_g, \dots, j^i_h, \dots, j^i_{n_i})$ by switching job j^i_h with job j^i_g. The generated solution is better then the current one if $f(S^i_{(h,g)}) < f(S^i)$.

LS2 is based on an inter-machine insertion neighborhood, which selects two machines i, $i' \in M$ and two job positions h $(1 \leq h \leq n_i)$ and g $(1 \leq g \leq n_{i'})$ in the sequences of jobs S^i and $S^{i'}$, respectively, and, if $i' \in M_{j^i_h}$, removes job j^i_h from S^i (generating $S^i_{(j^i_h, h^-)} = (j^i_1, \dots, j^i_{h-1}, j^i_{h+1}, \dots, j^i_{n_i})$) and inserts j^i_h before the g-th position of $S^{i'}$ (generating $S^{i'}_{(j^i_h, g^+)} = (j^{i'}_1, \dots, j^{i'}_{g-1}, j^i_h, j^{i'}_g, \dots, j^i_{n_{i'}})$). The generated solution is better then the current one if $f(S^i_{(j^i_h, h^-)}) + f(S^{i'}_{(j^i_h, g^+)}) < f(S^i) + f(S^{i'})$.

Given a feasible sequence of jobs $S^i = (j^i_1, \dots, j^i_{n_i})$ and two job positions h and g $(1 \leq h < g \leq n_i)$ of S^i, a sub-sequence of consecutive jobs $S^i_{(h-g)} = (j^i_h, \dots, j^i_g)$ is t-maximal if $s^i_{j_{h-1} j_h} > t$ (or $h = 1$), $s^i_{j_g j_{g+1}} > t$ (or $g = n_i$), and $s^i_{j,j+1} \leq t$ (for each $j = h, \dots, g-1$). Stated in another way, a t-maximal sub-sequence is a maximal sequence of consecutive jobs to be executed on machine

i whose setup times do not exceed t. By definition, a t-maximal sub-sequence $S^i_{(h-g)} = (j^i_h, \ldots, j^i_g)$ of S^i is composed by at least of two jobs.

In LS3, firstly, for each $i \in M$, a list L_i of all t-maximal sub-sequences of S^i is created. LS3 is based on an intra-machine sub-sequence insertion neighborhood, which selects a machine $i \in M$, a t-maximal sub-sequence $S^i_{(h-g)}$ in L_i, a job position k ($1 \leq k < h$ or $g < k \leq n_i$), and then generates a new sequence $S^i_{k,(h-g)} = (j^i_1, \ldots, j^i_{k-1}, j^i_h, \ldots, j^i_g, j^i_k \ldots, j^i_{n_i})$ by inserting the sub-sequence $S^i_{(h-g)}$ before the k-th position of S^i. The generated solution is better then the current one if $f(S^i_{k,(h-g)}) < f(S^i)$.

All three local search procedures operate in a first-improvement manner. For each $i \in M$, all the possible moves are considered, evaluating all the solutions in a neighborhood. As soon as an improving solution is found, the current solution S is updated and the local search is re-executed on S. If, instead, no improving solution is found, then the local search returns the current local optimal solution.

All the described methods are combined in an improved single heuristic algorithm (ISHA). ISHA finds a first feasible solution S by means of CGHA. Then, after the constructive phase is completed, S is brought to a local minimum by sequentially invoking LS1, LS2, and LS3, in this order.

4 Computational Results

In this section, computational experiments are performed on various instances encountered in industry practice. In particular, the solutions obtained by means of the heuristic methods described in Sect. 3 are compared among them. All

Table 1. Computational results of CGHA and ISHA

Inst.	$\|J\|$	$\|M\|$	CGHA				ISHA			
			WT	*TST*	*f.o.*	*run t.*	*WT*	*TST*	*f.o.*	*run t.*
I01	61	10	4812	8460	7001	0.05	2587	7635	5616	4.61
I02	90	10	9322	12895	11466	0.07	5203	11725	9116	19.19
I03	87	10	19127	12620	15223	0.07	13632	10890	11987	13.42
I04	106	10	15987	15520	15707	0.11	9241	13680	11904	20.92
I05	96	11	18110	13530	15362	0.08	15375	12125	13425	12.66
I06	89	10	8112	12730	10883	0.08	3598	10555	7772	14.45
I07	89	10	18792	14125	15992	0.09	15187	11850	13185	13.04
I08	78	10	4312	10315	7914	0.09	2159	9625	6639	8.77
I09	94	10	15796	13510	14425	0.08	12436	12170	12277	17.54
I10	104	10	7260	14335	11505	0.10	5027	12380	9439	19.19
I11	95	10	26136	13940	18819	0.08	21327	12300	15911	14.16
I12	103	10	18898	13400	15599	0.09	14456	12190	13096	16.86
AVG			13889	12948	13324	0.08	10019	11427	10864	14.57
GAP							−33%	−12%	−19%	

the algorithms are coded in Python version 3.9.2 and run on a computer with Intel(R) Core (TM) i7-10510U with CPU 1.80 GHz and RAM 16 GB, using Windows 10 Pro 64-bit.

We consider 12 real-world instances, each corresponding approximately to a week of production. Some randomization was applied to all instances in order to meet the company privacy requirements. The number of jobs spans from 61 to 106, while the number of machines is 10 or 11. According to the company interests, the parameters α and β of the objective function are set to 0.4 and 0.6, respectively. The parameter t in the intra-machine sub-sequence insertion moves is set to 125, since it has been seen that, with this value, LS3 produces good-quality solutions. In Table 1, the solutions obtained by means of CGHA and ISHA are compared. Columns $|J|$ and $|M|$ refer to the size of the instances in terms of number of jobs and number of machines, respectively. Entries in columns WT, TST, and $f.o.$ exhibit the sum of total weighted tardiness, the total setup times, and the value of the objective function, respectively. Finally, entries in columns *run t.* exhibit the run-time required to solve the instances (expressed in seconds). The mean of the percentage gaps from the solution values obtained by means of CGHA are reported in the row GAP, while entries in row AGV exhibit the average values of each column. Table 1 shows that CGHA is able to find a solution in a very short time (on average, less than a tenth of a second) and, in terms of quality, it is similar to the solution produced by the company. ISHA is always able to significantly improve the quality of the initial solution found by means of CGHA. More precisely, the value of the objective function improves by an average of 19%, with an average percentage improvement of the weighted tardiness and of total setup times equal to 33% and 12%, respectively.

Table 2. Comparative evaluation of the different locals search procedures

Inst.	$\|J\|$	$\|M\|$	CGHA		CGHA+LS1		CGHA+LS2		CGHA+LS3		ISHA	
			f.o.	*run t.*	*f.o.*	*run t.*	*f.o.*	*run t.*	*f.o.*	*run t.*	*f.o.*	*run t.*
I01	61	10	7001	0.05	5809	1.46	6565	3.86	6784	0.41	5616	4.61
I02	90	10	11466	0.07	9585	7.81	10791	8.58	10838	1.06	9116	19.19
I03	87	10	15223	0.07	12656	8.47	14083	4.79	13741	3.11	11987	13.42
I04	106	10	15707	0.11	12438	9.84	14567	13.52	14058	1.26	11904	20.92
I05	96	11	15362	0.08	14073	2.42	14379	12.73	14556	0.67	13425	12.66
I06	89	10	10883	0.08	8304	5.64	9163	12.36	9916	0.51	7772	14.45
I07	89	10	15992	0.09	14229	3.64	14643	10.44	14656	0.78	13185	13.04
I08	78	10	7914	0.09	6998	3.12	7490	12.13	7751	0.28	6639	8.77
I09	94	10	14425	0.08	12695	12.04	13619	14.41	13583	0.50	12277	17.54
I10	104	10	11505	0.10	10217	4.76	10443	19.41	10686	0.45	9439	19.19
I11	95	10	18819	0.08	16397	4.03	13046	14.05	18029	0.34	15911	14.16
I12	103	10	15599	0.09	13708	5.96	14273	18.62	14491	0.51	13096	16.86
AVG			13324	0.08	11426	5.76	11922	12.08	12424	0.82	10864	14.57
GAP					−14%		−10%		−6%		−19%	

Table 2 highlights the results of the second round of experiments, which aims at analyzing the impact of the different local searches (described in Sect. 3) with respect to the initial solution computed by CGHA. In CGHA+LS1, CGHA+LS2, and CGHA+LS3, after the constructive phase is completed, the initial solution is brought to a local minimum by means of LS1, LS2, and LS3, respectively. The results demonstrate that LS1 is able to produce on average the largest improvement in therms of quality solution. Indeed, the value of the objective function improves by 14% with LS1, by 10% with LS2, and by 6% with LS3. This result reveals the strongly sequence-dependent nature of the problem. Indeed, thanks to the intra-machine swap moves of LS1, each sequence of jobs can be re-sorted, obtaining a large improvement of the initial solution. The slight improvements brought by LS3 are obtained in a very short time (on average, less than one second), thus LS3 is able to provide a good trade-off between the neighborhood size (hence time needed to explore it) and its effectiveness. On the other hand, LS2 is the most time-consuming local search, reflecting the fact that the LS2 neighborhood size is the largest one.

5 Conclusions

In this paper, we presented a local search approach for the PMSCP. Firstly, we proposed a constructive greedy procedure (CGHA) for the generation of a good initial solution and then a local search-based approach (ISHA) which combines CGHA with three different local search procedures. Experiments over real-world instances confirmed that the proposed methods are able to find good-quality solutions and can provide a quick support to the company on its weekly decisions. More precisely, the computational results show that CGHA is able to find solutions in a very short time, which, in therms of quality, are similar to the solutions produced by the company. Furthermore, ISHA is always capable to significantly improve the quality of the initial solution (by an average of about 20%) within small computing times (less than 20 s). These good results were confirmed in practice. Indeed, ISHA is currently used during the weekly production scheduling by the company, which attested that ISHA is capable of producing good quality results.

It is worthwhile to remark that the computing times of all the proposed approaches, including the full ISHA approach, are currently not a major issue. Indeed, taking into account that the planned activities cover about one week, larger computing times are feasible and further refinements of the proposed approaches are possible. In particular, as a possible topic for future research, we are interested in introducing some perturbation mechanism inside ISHA to help the search to escape from local optima solutions and to bring diversification into the search.

References

1. Allahverdi, A.: The third comprehensive survey on scheduling problems with setup times/costs. Eur. J. Oper. Res. **246**(2), 345–378 (2015)
2. Graham, R., Lawler, E., Lenstra, J., Kan, A.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**(C), 287–326 (1979)
3. Lenstra, J., Rinnooy Kan, A., Brucker, P.: Complexity of machine scheduling problems. Ann. Discrete Math. **1**(C), 343–362 (1977)
4. Meunier, F., Neveu, B.: Computing solutions of the paintshop-necklace problem. Comput. Oper. Res. **39**(11), 2666–2678 (2012)
5. Paine, F.A., Paine, H.Y.: A Handbook of Food Packaging. Springer, Boston (2012)
6. Pinedo, M.: Scheduling. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-2361-4
7. Schuurman, J., Van Vuuren, J.H.: Scheduling sequence-dependent colour printing jobs. S. Afr. J. Ind. Eng. **27**(2), 43–59 (2016)