Energy-efficient Flow-shop Scheduling in the Printing Industry using Memetic Algorithm

Ke Shen*, Fabian Heyse*, Toon De Pessemier*, Luc Martens*, Wout Joseph*
*Department of Information Technology, Ghent University/IMEC, Ghent, Belgium

***Abstract*—Facing the climate change and the energy crisis, plenty of challenges remain to achieving the carbon neutrality in the energy intensive industry. This work investigates a flow-shop production scheduling problem minimizing the energy cost without delaying any jobs. The research problem comes from a Belgium printing company with solar panels as its own energy source. Two sub-problems, the job sequence determination and the job-machine allocation, are tackled using a memetic algorithm. An energy-efficient local search heuristic is also designed to make best use of the self-generated electricity. Validated using realistic production data of 12 jobs in a 2 day planning horizon, the proposed method outperforms a standard binary-encoded genetic algorithm in both solution quality (having 27.17% lower energy cost) and the calculation time (saving 69.89%).**

***Index Terms*—memetic algorithm, production scheduling, energy efficiency**

I. Introduction

Climate change is more than ever a topic that gets the attention it deserves [1]. Many countries and regions have claimed to achieve the carbon neutrality in their long-term strategies [2]. From the power sector to industry, plenty of challenges and opportunities remain for the companies: they are encouraged to consume energy in the most efficient way, or even to install their own renewable and clean energy source [3]. If the self-generated power is more than required, the surplus will be injected into the electrical grid for the future consumption at a much lower price.

The energy-efficient production scheduling problem (EEPSP) has been extensively studied in recent studies because of the urgent demand. The complexity of EEPSP is proven as NP-hard on multi-state single machines in [4] and [5]. A few EEPSPs with more complicated machine environment (e.g. job-shop in [6]) or multi-objectives [7] are solved using mathematical programming and related approximation methods. Most of the complex EEPSPs are tackled using evolutionary algorithms (EAs) [8], since they can produce feasible and high-quality solutions within very short computation time: the modified genetic algorithms are reported in [9]–[13]; the multi-objective EAs are presented in [14]–[17]; other variants of EAs are introduced in [18] and [19]. In these works, there is no improved exploration for the energy efficiency: sometimes it is accomplished by achieving other objectives [9], or by subjective transformation to other indicators [13]. In this work, the energy efficiency is our main focus and optimization objective. Moreover, a memetic algorithm (MA, extension of the traditional genetic algorithm that uses local search [20]) is proposed as the approach to the research problem, where a specific local search heuristic for energy-efficient candidate solutions are introduced as well.

The remainder of this paper investigates the EEPSP from a Belgium printing company with solar panels as its own energy source. The contributions are as follows: (1) A generalized mixed integer programming (MIP) model is introduced to describe the research problem. (2) A memetic algorithm with an energy-efficient local search heuristic is proposed. (3) Numerical experiments are conducted using realistic production data to demonstrate the competitive performance of the proposed approach against a standard genetic algorithm. (4) An energy demand-response framework for industrial applications of the proposed approach is introduced.

II. Problem Definition

The investigated EEPSP is described using a generalized MIP model with the following notations. The production layout is a flow-shop: a set J of n independent jobs are to be processed by a set M of m machines in series.

Input parameters:

- n: number of jobs.
- j: index for jobs in the job set J, $j \in \{1, 2, ..., n\}$.
- m: number of machines (stages).
- i: index for machines in the machine set M, $i \in \{1, 2, ..., m\}$.
- p_j^i: processing time of job j on machine i.
- s_j^i: set up time of job j on machine i.
- l: index for jobs in a schedule, $l \in \{1, 2, ..., n\}$.
- E^i, power of machine i.
- H: planning horizon of the schedule.
- t: index of time periods in the planning horizon, $t \in \{1, 2, ..., H\}$.
- e_t, energy cost at period t.
- d_j, due date for job j.

Binary decision variables:

- w_{jl}, if job j is in the lth position of a sequence.
- z_{jt}^i, if job j starts on machine i at time period t.
- x_{jt}^i, if job j is executed on machine i during time period t.

The optimization objective is to minimize the energy cost C_E:

$$min \ \{C_E\}$$

The constraints are as follows:

$$\sum_{l=1}^{n} w_{jl} = 1 \quad \forall j \tag{1}$$

Final version DOI: 10.1109/ETFA52439.2022.9921519

$$\sum_{j=1}^{n} w_{jl} = 1 \quad \forall l \tag{2}$$

$$\sum_{t=1}^{H} x_{jt}^{i} = p_{j}^{i} \quad \forall j, i \tag{3}$$

$$z_{jt}^{i} = max\{x_{jt}^{i} - x_{jt-1}^{i}, 0\} \quad \forall t \geq 2, j, i, \tag{4}$$

$$\sum_{j=1}^{n} x_{jt}^{i} \leq 1 \quad \forall t, i \tag{5}$$

$$\sum_{t=1}^{H} z_{jt}^{i} = 1 \quad \forall j, i \tag{6}$$

$$\sum_{t=1}^{H}\sum_{i=1}^{m} z_{jt}^{i} = m \quad \forall j \tag{7}$$

$$\sum_{j=1}^{n}(\sum_{t=1}^{H}\sum_{i=1}^{m} x_{jt}^{i} - d_j) \leq 0 \tag{8}$$

$$C_E = \sum_{t=1}^{H}\sum_{j=1}^{n}\sum_{i=1}^{m} x_{jt}^{i} E^{i} e_t \tag{9}$$

The sequence (order) of jobs in a schedule is fixed during the processing (Constraints (1) and (2)). A job can only start when its previous stage is finished (Constraints (3) and (4)). Machines can process only one job at a time (Constraint (5)). No preemption is allowed (Constraint (6)). Each job contains exactly m stages (Constraint (7)). The ith stage of a job must be executed on the ith machine (Constraints (6) and (7)). No tardiness is allowed for any job (Constraint (8)).

Other case-specific characteristics (of the input parameters) include:

- A day shift is from 7 am to 11 pm.
- A job needs to finish the same day when it is started.
- Job types are determined by the paper or ink used in the label.
- The setup time between jobs will reduce if two adjacent jobs are of the same type.
- Machine states are simplified to on (when processing) and off (when idle).

These case-specific (not applicable to other EEPSPs) characteristics are not formulated in the presented generalized MIP model. Moreover, the conversion of the EEPSP to the MIP model might not be ideal [21] therefore this work does not use mathematical programming [6] or related approximation methods [7].

The NP-hardness of the investigated EEPSP can be inferred from the complexity of a fundamental problem: the flow-shop scheduling problem minimizing total tardiness (denoted by $F|prmu|\sum T_j$ using the three-field notation [22]), known as NP-hard with two or more stages in the flow [23].

III. Method Description

The pseudo-code of the proposed memetic algorithm with the energy-efficient local search heuristic (MA-ESH) tackling the investigated EEPSP is presented in Algorithm 1. The inputs include the problem information (P) and the algorithm parameters (Q). The output is the best $Solution$ including a job sequence ($Solution.seq$) and the corresponding job-machine allocation ($Solution.allc$) with the lowest energy cost ($Solution.fit$). Each individual $pop[h]$ represents a candidate solution.

Algorithm 1 The memetic algorithm with the energy-efficient local search heuristic (MA-ESH).

```
Input: $P : Problem, Q : Parameters$
Output: $Solution$
 1: $pop \leftarrow \emptyset$
 1: // Initialization:
 2: for $h = 1$ to $Q.popsize$ do
 3:     $t.seq \leftarrow GenerateSequence(P, Q)$
 4:     $t.allc \leftarrow GenerateAllocation(t.seq, P, Q)$
 5:     $t.fit \leftarrow CalculateFitness(t, P)$
 6:     $pop[h] \leftarrow t$
 7: end for
 8: repeat
 9:     for $h = 1$ to $Q.popsize$ do
 9:         // Reproduction:
10:         $p1, p2 \leftarrow Select(pop)$
11:         $c1.seq \leftarrow ReproducSequence(p1.seq, p2.seq, Q)$
12:         $c1.allc \leftarrow ReproductAllocation(p1.allc, p2.allc, Q)$
13:         $c1.fit \leftarrow CalculateFitness(c, P)$
13:         // Local Search:
14:         $c2.seq \leftarrow c1.seq$
15:         $c2.allc \leftarrow ESH(c1.seq, P, Q)$
16:         $c2.fit \leftarrow CalculateFitness(c2, P)$
17:         if $c1.fit \leq c2.fit$ then
18:             $pop[h] \leftarrow c1$
19:         else
20:             $pop[h] \leftarrow c2$
21:         end if
22:     end for
23: until $TerminationCriterion(Q)$
24: $Solution \leftarrow FindBest(pop)$
25: return $Solution$
```

A. The Memetic Algorithm (MA)

In the initialization stage, the individuals are initialized by first ranking the jobs according to their due dates and then performing permutations between jobs with the same due dates (line 3 in Algorithm 1). Such modified earliest-due-date rule [24] is applied to minimize the delay of jobs, and to reduce the solution space size. For job-machine allocation, the start time of job l on machine i is denoted as S_{li}. A job-machine allocation ($t.allc$) is represented using a $n \times m$ matrix whose elements indicate S_{li}. For each sequence ($t.seq$), the corresponding job-machine allocations are generated by adding random time intervals between the jobs (line 4 in Algorithm 1), during which the machine states are off but the solar panels are still working. The fitness (energy cost) can be calculated for each individual (line 5 in Algorithm 1).

In the reproduction stage, for each individual two parents ($p1$ and $p2$) are selected from the current population using tournament selection [25] (line 11 in Algorithm 1). The generation of the children ($c1$) includes the reproduction of the new sequence and the corresponding job-machine allocation. The reproduction of the new sequence (line 12 in Algorithm 1) conducts first the one-point-crossover [26], which has a high chance to preserve the the same type of the adjacent jobs. Afterwards, with a certain mutation rate (in Q), a sub sequence of jobs having the same due date are permutated. The reproduction of the job-machine allocation (line 13 in Algorithm 1) using a modified one-point crossover is demonstrated as follows:

$$p1 = \begin{bmatrix} p1_{1,1:m} \\ ... \\ \underline{p1_{k,1:m}} \\ ... \\ p1_{n,1:m} \end{bmatrix} \quad p2 = \begin{bmatrix} p2_{1,1:m} \\ ... \\ \underline{p2_{k,1:m}} \\ ... \\ p2_{n,1:m} \end{bmatrix} \quad \to \quad c1 = \begin{bmatrix} p1_{1,1:m} \\ ... \\ \underline{p1_{k,1:m}} \\ ... \\ p2_{n,1:m} \end{bmatrix}$$

For each parent a random job k is chosen as the cut point (the underlined kth row). The start time of the jobs before the cut point of $p1$ ($p1_{1,1:m}$ to $p1_{k,1:m}$), and after the cut point of $p2$ ($p2_{k,1:m}$ to $p2_{n,1:m}$), are copied to the children. The mutation is performed as follows: for each a_{li}, there is a probability (decided by a biased coin toss [28]) for it to be shifted earlier or later of a small time interval. Since planning a job later has a larger impact than planning it earlier (the subsequent jobs could be influenced by the late shift of the current job), the probability for the earlier shift is higher than the later shift. After reproduction the new job-machine allocation should not violate the following constraints:

$$S_{li} \geq S_{li-1} + s_l^{i-1} + p_l^{i-1} \tag{10}$$

The start time of a job l on machine i should be equal or later than its finish time on the previous machine $i-1$.

$$S_{li} \geq S_{l-1i} + s_{l-1}^{i} + p_{l-1}^{i} \tag{11}$$

The start time of a job l should be equal or later than the finish time of the previous job $l-1$ on the same machine. Otherwise, the reproduction operation is re-conducted. Afterwards, an energy-efficient local search heuristic is performed and the details are presented in the next section.

When the stop criterion (line 23 in Algorithm 1, e.g. a maximum number of iterations) is reached, the best individual with the lowest fitness (energy cost) is selected from the final population and returned as the $Solution$.

B. The Energy-efficient Local Search Heuristic (ESH)

Local search is the most important feature of the MA [20] to find better individuals in each iteration of the algorithm and to prevent the premature convergence. In our method, the ESH is designed to pursue the schedule to use as much the surplus of the self-generated energy as possible, to achieve a lower energy cost and carbon emission.

Following notations are introduced to present the ESH: the electricity that is needed to process job l on machine i is denoted as P_{li}. The surplus (net injected electricity into the grid, which is the total self-generated energy minus the total consumed energy) at the moment t is denoted as I_t. Apart from the constraints (10) and (11), the S_{li} has to comply with an additional constraint:

$$I_{S_{li}} \geq \beta_{li} \times P_{li} \tag{12}$$

The surplus at S_{li} should be equal or higher than the required energy, which is calculated using P_{li} multiply a scale factor β_{li}.

Algorithm 2 The energy-efficient local search heuristic (ESH).

```
Input: $seq$, $P : Problem$, $Q : Parameters$
Output: $allc$
 1: $\beta_{li} \leftarrow 1$ // All $\beta_{li}$ are set to 1
 2: $i \leftarrow 1$, $l \leftarrow 1$
 3: while $i \leq m$ do
 3:   // For each machine
 4:   while $l \leq n$ do
 4:     // For each job
 5:     $S_{li} \leftarrow Calculate(S_{li})$ // According to (12)
 6:     $k \leftarrow 1$
 7:     while $S_{li} + s_l^i + p_l^i > d_l$ and $k \leq 5$ do
 7:       // Overdue:
 8:       $\beta_{li} \leftarrow Adjust(\beta_{li})$
 9:       $S_{li} \leftarrow Calculate(S_{li})$ // According to (12)
10:       $k \leftarrow k + 1$
11:     end while
12:     if $S_{li} + s_l^i + p_l^i > d_l$ then
12:       // Overdue:
13:       $B \leftarrow Adjust(B)$
14:       $l \leftarrow 1$, $i \leftarrow 1$ // Re-organize all jobs
15:     else
16:       $l \leftarrow l + 1$
17:     end if
18:   end while
19:   $i \leftarrow i + 1$
20: end while
21: return $allc$
```

The procedure of ESH is introduced in Algorithm 2. Since the tardiness of jobs is often not allowed in realistic industrial applications, a self-adaptive scale factor β_{li} is designed to cope with the due dates (d_j) of jobs. If all β_{li} are set to 1, each job has to wait until enough surplus is generated to start the processing. The energy cost will be the lowest (as zero), but obviously many jobs will be overdue. An intuitive idea to this issue is that β_{li} should be lowered (line 9 in Algorithm 2) for the late-released jobs (when l is relatively large and close to n). However, it can happen that the β_{ki} of a late-released job k is already very small and the job is planned right after its previous job $k-1$. In this case, job k can not be planned earlier and still remains overdue (line 12 in Algorithm 2)). To prevent this phenomenon, B should be lowered to shift all jobs earlier in the planning horizon.

As an example, suppose there are 4 jobs to be planned on 1 machine. The B is initialized as $[1, 1, 1, 1]$. In case job 3 is overdue, β_{31} is adapted by being lowed with a fixed setup (from Q, e.g., 0.05). Assuming that job 3 is still overdue after 5 adaptations, the B will be updated from $[1, 1, 0.75, 1]$ to $[0.9, 0.9, 0.9, 0.9]$. If job 3 is still late, β_{31} will again be lowered by 0.05. The ESH repeats such adjustment until it finds a feasible schedule.

By making better use of the surplus, the ESH enables the MA with a great chance to find better individuals ($c2$ in Algorithm 1) than the reproduced ones from the parents ($c1$ in Algorithm 1) in each iteration.

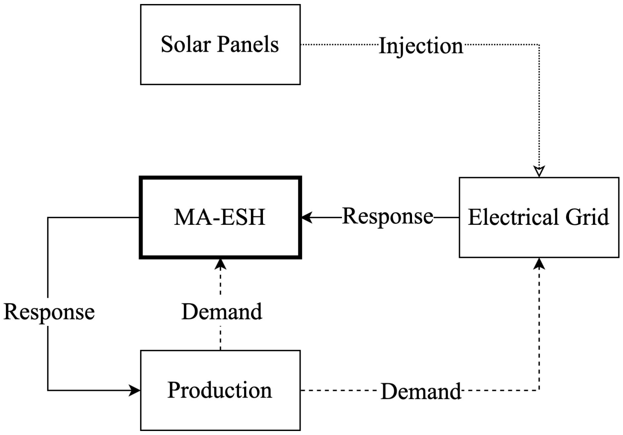


Fig. 1: The energy demand-response framework for the MA-ESH application.

IV. Numerical Experiments

A. The Energy Demand-response Flow

Shown in Figure 1, where the demands are in dashed lines and the responses are in solid lines, the production processes demand the energy from the electricity grid and the production plan from the managing department using the MA-ESH. Seeking an energy-efficient schedule also demands the floating consumption and injection tariff from the grid. The electricity generation forecast (the dotted line from the solar panels to the electrical grid) could be predicted using machine learning methods (often influenced by the weather condition), which is beyond the scope of this work. In our experiments, the power profiles of machines are known from the company. The electricity generation data and the tariff from the power generation market are provided by the solar panel manufacturer.

The production layout is a flow-shop consisting of 3 machines. The first machine (the pre-printing stage, 6kW) is responsible for making the printing forms. Its setup time for a job depends on the number of colors used in the product. The second machine (the printing stage, 24kW) is for conventional press and the printing speed for each job is different. The last machine (the post-printing stage, 6kW) is for inspection, cutting, and rewinding, whose processing speed is synchronized with the second machine.

There are 12 label printing jobs to be processed in a 2 day horizon, the most important characteristics are listed in Table I. The due dates are represented using minutes since the beginning of the planing horizon. The processing time of the jobs on each machine are calculated as the division of the order quantity and the machine speed.

B. Parameter Settings

Monte Carlo simulations are performed for the best parameter settings of the MA-EHS. A trade-off has to be made between the solution quality (optimality) and calculation time. Considering the hardware capacity of our laptop (Intel® i7 8565U, 16 GB 2400MHz RAM), the population size is set to 80, the stop criterion is that the algorithm will stop after 300 generations, or the best individual remains the same for 30 iterations. As shown in Figure 2, the crossover and mutation probabilities are decided considering the saved energy cost (compared with a non-optimized empirical schedule) and the calculation time, where the good values are marked using green. The algorithm has the lowest calculation time when there are least operations to perform (with the lowest crossover and mutation probability). A high mutation probability makes the algorithm converge slowly since the good individuals are more likely to be destroyed. Such effect is weakened by an increased crossover probability which makes the algorithm more actively explore the solution space near the good individuals of the current generation. The best settings ensure that the delivered objective value is (near-) optimal with a reasonable calculation time. Therefore, the crossover and mutation probabilities are set as 0.4 and 0.2 respectively.

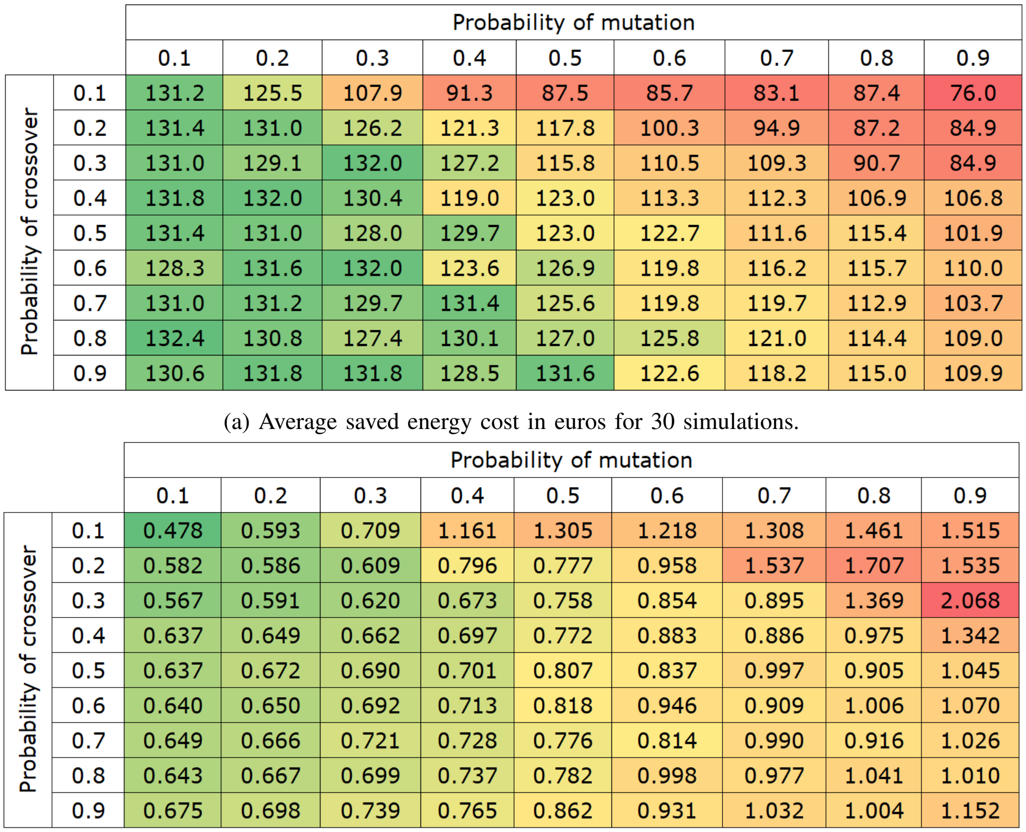
Other parameters are also decided using the Monte Carlo simulations and presented as follows: in the mutation operation for the job-machine allocations, the biased coin toss for an earlier shift is 0.55 and for a later shift is 0.45, the small correction interval is 15 minutes. In the ESH, the adjustment step for the β_{li} and B is set as 0.05 and 0.1 respectively.

C. Results and Discussions

Shown in Table II, the MA-ESH is compared with a standard genetic algorithm (SGA) for 40 experiments in terms of the objective value and the calculation time. In our implementation, the SGA shares the same procedures as those of the MA-ESH except for the local search (line 14-16 in Algorithm 1). Instead, it replaces the local search using the microbial elitism selection [29]. The average result of 40 experiments shows that the SGA takes 2.79 seconds to provide a solution with the energy cost of 43.18 euros, while the MA-ESH takes 0.84 seconds to find a better solution with the energy cost of 31.45 euros. Therefore, the MA-ESH requires only 30.11% of the SGA's calculation time to achieve a 27.17% better energy cost.

Job No.	Due date	Type	Machine 1		Machine 2		Machine 3	
	[min]		Setup [min]	Processing [min]	Setup [min]	Processing [min]	Setup [min]	Processing [min]
1	960	2	18	22	15	49.5	24	49.9
2	1920	1	16	4	13	88.9	15	89.5
3	1920	2	18	22	15	28.2	24	28.3
4	1920	2	18	4	15	4	24	4
5	1920	2	18	8	15	17.7	24	17.8
6	960	1	16	14	13	86.6	15	87.2
7	960	1	16	8	13	28.9	15	29.1
8	1920	3	28	10	16	104.6	25	105.3
9	1920	2	18	12	15	20	24	20.1
10	1920	1	16	2	13	98.4	15	99.1
11	960	2	18	14	15	163.6	24	164.7
12	1920	3	28	4	16	65.8	25	66.3

TABLE I: Order information for 12 jobs to be processed.



		Probability of mutation								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Probability of crossover	0.1	131.2	125.5	107.9	91.3	87.5	85.7	83.1	87.4	76.0
	0.2	131.4	131.0	126.2	121.3	117.8	100.3	94.9	87.2	84.9
	0.3	131.0	129.1	132.0	127.2	115.8	110.5	109.3	90.7	84.9
	0.4	131.8	132.0	130.4	119.0	123.0	113.3	112.3	106.9	106.8
	0.5	131.4	131.0	128.0	129.7	123.0	122.7	111.6	115.4	101.9
	0.6	128.3	131.6	132.0	123.6	126.9	119.8	116.2	115.7	110.0
	0.7	131.0	131.2	129.7	131.4	125.6	119.8	119.7	112.9	103.7
	0.8	132.4	130.8	127.4	130.1	127.0	125.8	121.0	114.4	109.0
	0.9	130.6	131.8	131.8	128.5	131.6	122.6	118.2	115.0	109.9

(a) Average saved energy cost in euros for 30 simulations.

		Probability of mutation								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Probability of crossover	0.1	0.478	0.593	0.709	1.161	1.305	1.218	1.308	1.461	1.515
	0.2	0.582	0.586	0.609	0.796	0.777	0.958	1.537	1.707	1.535
	0.3	0.567	0.591	0.620	0.673	0.758	0.854	0.895	1.369	2.068
	0.4	0.637	0.649	0.662	0.697	0.772	0.883	0.886	0.975	1.342
	0.5	0.637	0.672	0.690	0.701	0.807	0.837	0.997	0.905	1.045
	0.6	0.640	0.650	0.692	0.713	0.818	0.946	0.909	1.006	1.070
	0.7	0.649	0.666	0.721	0.728	0.776	0.814	0.990	0.916	1.026
	0.8	0.643	0.667	0.699	0.737	0.782	0.998	0.977	1.041	1.010
	0.9	0.675	0.698	0.739	0.765	0.862	0.931	1.032	1.004	1.152

(b) Average calculation time in minutes for 30 simulations.

Fig. 2: Monte Carlo simulations to find the best crossover and mutation probabilities.

Simulation No.	Energy Cost [€]		Calculation Time [s]	
	MA-ESH	SGA	MA-ESH	SGA
1	28.41	43.78	0.85	2.54
2	43.77	43.70	0.88	2.12
...	...	...	...	...
39	29.39	41.20	0.72	2.01
40	46.19	42.89	0.88	3.75
Average	31.45	43.18	0.84	2.79

TABLE II: Comparison of GA and MA-ESH for 40 experiments.

Algorithms	Best Sequence	Energy cost [€]	Calculation time [s]
MA-ESH	$[7, 6, 1, 11, 9, 5, 3, 4, 12, 8, 2, 10]$	27.54	0.87
SGA	$[7, 6, 1, 11, 9, 3, 5, 4, 12, 8, 10, 2]$	27.62	2.28

TABLE III: The best sequence and the lowest energy cost found by GA and MA-ESH in 40 experiments.

Range (%) From the Known Best Solution	Executions [#]		Executions [%]	
	MA-ESH	SGA	MA-ESH	SGA
1%	11	1	27.5%	2.5%
3%	16	1	40%	2.5%
5%	22	3	55%	7.5%
10%	35	7	87.5%	17.5%

TABLE IV: Solution quality for GA and the MA-ESH in 40 experiments.

The performances of the MA-ESH and the SGA are further investigated in terms of the solution quality. The best solutions provided by the two algorithms in 40 experiments are further investigated in Table III. The lowest energy cost found by the SGA (27.62 euros) is very close (within the 1% range) to that of the MA-ESH (27.54 euros), which is the known best solution in 40 experiments. Therefore, it is concluded that the SGA can still find the (near-)optimal solution, although it requires more calculation time. The Gantt chart for such best schedule found by the MA-ESH is shown in Figure 3, where the jobs are distinguished with different colors. The convergence plot of the MA-ESH in 40 experiments is provided in Fig. 4. It is noted that some experiments stopped in less than 300 generations since the other termination condition is met (no better individual is found over 30 iterations).

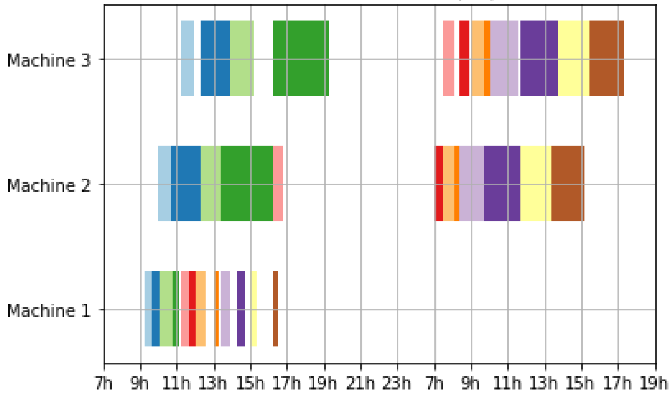


Fig. 3: Gantt chart of the best schedule found by MA-ESH.

The quality and the distribution of both algorithm's solutions are summarized in Table IV. In most (87.5%) experiments, the MA-ESH can converge to a solution within the 10% range of the known best schedule. In 27.5% of the experiments, it can approach the known best schedule within the 1% range. While for the SGA, only 2.5% of the 40 experiments can reach within the 1% range of the known best schedule, and 82.5% of the experiments provide outcomes out of the 10% range. Therefore, the MA-ESH outperforms the SGA in the solution quality and the calculation time.

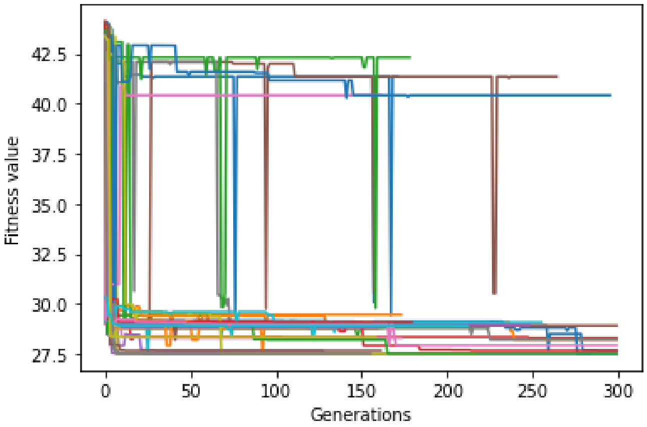


Fig. 4: Convergence plot of MA-ESH in 40 experiments.

V. Conclusion

In this work, an energy efficient flow-shop production scheduling problem from the printing industry is investigated by solving the two sub-problems: the job sequence determination and the job-machine allocation based on that sequence. The optimization objective is to minimize the energy cost of a schedule without delaying any jobs. A memetic algorithm is proposed to find the near-optimal solutions, with an energy-efficient local search heuristic to make best use of the self-generated electricity. In numerical experiments based on realistic production data, our method outperforms a standard genetic algorithm in both solution quality (27.17% lower energy cost) and the calculation time (69.89% shorter).

For future research, one direction is to improve the solution quality of the MA-ESH since it can not always converge to the best known schedule. Another is to consider the load balancing problem for the self-generated electricity to avoid the overload of the electrical grid.

References

[1] X. Zhao, X. Ma, B. Chen, Y. Shang, M. Song, "Challenges toward carbon neutrality in China: Strategies and countermeasures," *Resources, Conservation and Recycling,* vol. 176, no. 105959, Jan. 2022

[2] X. Li, T. Damartzis, Z. Stadler, S. Moret, B. Meier, M. Friedl, F. Maréchal, "Decarbonization in Complex Energy Systems: A Study on the Feasibility of Carbon Neutrality for Switzerland in 2050," *Frontiers in Energy Research,* vol. 8, 2020

[3] A. Qazi, F. Hussain, N. A.Rahim, G. Hardaker, D. Alghazzawi, K. Shaban, K. Haruna, "Towards Sustainable Energy: A Systematic Review of Renewable Energy Sources, Technologies, and Public Opinions," *IEEE Access,* vol. 7, pp. 63837-63851, 2019

[4] M. Aghelinejad, Y. Ouazene, A. Yalaoui, "Complexity analysis of energy-efficient single machine scheduling problems," *Operations Research Perspectives,* vol. 6, no. 100105, 2019

[5] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, C. Liu, "Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 10, pp. 4548-4556, Oct. 2018

[6] H. Golpîra, S. A. R. Khan, Y. Zhang, "Robust Smart Energy Efficient Production Planning for a general Job-Shop Manufacturing System under combined demand and supply uncertainty in the presence of grid-connected microgrid," *Journal of Cleaner Production,* vol. 202, pp. 649-665, Nov. 2018

[7] J. Xu, L. Wang, "A Feedback Control Method for Addressing the Production Scheduling Problem by Considering Energy Consumption and Makespan", *Sustainability,* vol. 9, pp. 1185, Jul. 2017

[8] K. Gao, Y. Huang, A. Sadollah, L. Wang, "A review of energy-efficient scheduling in intelligent production systems", *Complex & Intelligent Systems,* vol. 6, pp. 237-249, Sept. 2019

[9] D. Min, T. Dunbing, G. Adriana, S. Miguel A., "Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints," *Robotics and Computer-Integrated Manufacturing,* vol. 59, pp. 143-157, Oct. 2019

[10] K. Shen, T. De Pessemier, X. Gong, L. Martens, W. Joseph, "Genetic optimization of energy-and failure-aware continuous production scheduling in pasta manufacturing," *Sensors,* vol. 19, pp. 297, Jan. 2019

[11] R. Zhang, R. Chiong, "Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *Journal of Cleaner Production,* vol. 112, pp. 3361-3375, Jan. 2016

[12] M. A. Salido, J. Escamilla, A. Giret, F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," *The International Journal of Advanced Manufacturing Technology volume,* vol. 85, pp. 1303-1316, Nov. 2015

[13] Z. Liu, J. Yan, Q. Cheng, C. Yang, S. Sun, D. Xue, "The mixed production mode considering continuous and intermittent processing for an energy-efficient hybrid flow shop scheduling," *Journal of Cleaner Production,* vol. 246, no. 119071, Feb. 2020

[14] F. Zhao, X. He, L. Wang, "A Two-Stage Cooperative Evolutionary Algorithm With Problem-Specific Knowledge for Energy-Efficient Scheduling of No-Wait Flow-Shop Problem," *IEEE Transactions on Cybernetics,* vol. 51, pp. 5291-5303, Nov. 2021

[15] W. Li, L. He, Y. Cao, "Many-Objective Evolutionary Algorithm With Reference Point-Based Fuzzy Correlation Entropy for Energy-Efficient Job Shop Scheduling With Limited Workers," *IEEE Transactions on Cybernetics,*

[16] E. Jiang, L. Wang, Z. Peng, "Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition," *Swarm and Evolutionary Computation,* vol. 58, no. 100745, Nov. 2020

[17] Z. Pan, D. Lei, L. Wang, "A Bi-Population Evolutionary Algorithm With Feedback for Energy-Efficient Fuzzy Flexible Job Shop Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,*

[18] T. Jiang, C. Zhang, H. Zhu, G. Deng, "Energy-efficient scheduling for a job shop using grey wolf optimization algorithm with double-searching mode," *Mathematical Problems in Engineering,* 2018

[19] T. Jiang, C. Zhang, H. Zhu, G. Deng, "Energy-efcient scheduling for a job shop using an improved whale optimization algorithm," *Mathematics,* vol. 6, no. 11, pp. 220, Oct. 2018

[20] J. D. Knowles, D. W. Corne, "M-PAES: a memetic algorithm for multiobjective optimization," *Proceedings of the 2000 Congress on Evolutionary Computation,* vol. 1, pp. 325-332, 2000

[21] A. Keha, K. Khowala, J. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Computers & Industrial Engineering,* vol. 56, no. 1, pp. 357-367, 2009

[22] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics,* vol. 5, pp. 287-326, 1979

[23] Q. C. Ta, J. Billaut, J. Bouquard, "Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem", *Journal of Intelligent Manufacturing,* vol. 29, pp. 617-628, 2018

[24] S. Roychowdhury, T. T. Allen, N. B. Allen, "A genetic algorithm with an earliest due date encoding for scheduling automotive stamping operations," *Computers & Industrial Engineering,* vol. 105, pp. 201-209, Mar. 2017

[25] B. Miller, D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems,* vol. 9, pp. 193-212, 1995

[26] K. Deb, K. Sindhya, T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," *Proceedings of the 9th annual conference on Genetic and evolutionary computation,* pp. 1187-1194, 2007

[27] H. M. Pandey, A. Chaudhary, D. Mehrotra, "A comparative review of approaches to prevent premature convergence in GA," *Applied Soft Computing,* vol. 24, pp. 1047-1077, Nov. 2014

[28] N. Gupta, M. Khosravy, N. Patel, N. Dey, R. G. Crespo, "Lightweight Computational Intelligence for IoT Health Monitoring of Off-Road Vehicles: Enhanced Selection Log-Scaled Mutation GA Structured ANN," *EEE Transactions on Industrial Informatics,* vol. 18, no. 1, pp. 611-619, Jan. 2022

[29] I. Harvey, "The microbial genetic algorithm," *European Conference on Artificial Life*, Berlin, Heidelberg, pp.126-133, 2009