Multi-objective Genetic Algorithm to Reduce Setup Waste in a Single Machine with Coupled-Tasks Scheduling Problem

Corentin Le Hesran, Anne-Laure Ladier, Valérie Botta-Genoulaz

Multi-objective genetic algorithm to reduce setup waste in a single machine with coupled-tasks scheduling problem*

Corentin Le Hesran[1], Anne-Laure Ladier[1][0000−0002−4201−4430], and Valérie Botta-Genoulaz[1][0000−0003−2565−6690]

Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Univ Lumière Lyon 2, DISP, EA4570, 69621 Villeurbanne, France. anne-laure.ladier@insa-lyon.fr

Abstract. This article studies a single-machine scheduling problem involving coupled-tasks and hard due dates. A genetic algorithm based on the Non-dominated Sorting Genetic Algorithm (NSGA) II model is proposed to carry out a bi-objective optimization of both holding cost and setup-related waste generation. Results show that the multi-objective genetic algorithm outperforms the previous approaches regarding both computation time and objective functions, showing that a reduction of setups of 36% is possible at the expense of an 11% increase in inventory with acceptable computation times. It also highlights the importance of multi-objective optimization for decision-making in case of conflicting objective functions.

Keywords: Multi-objective scheduling · Genetic algorithm · Waste prevention.

1 Introduction

Reducing the environmental impact of industrial production is currently a pressing challenge. At the operational level, new machining techniques and better operations scheduling improve environmental performance, although these largely focus on the energy consumption aspect [8]. Alternatively, recent work has appeared on the reduction of material waste rather than energy consumption and CO2 emissions, enabling better resource usage and lower waste generation through adequate scheduling [11]. Limiting waste generation is complementary with recycling and reuse approaches.

Such a case was studied in Le Hesran et al. [10], through the optimization of both inventory levels and setup-induced waste in the painting line of a hubcap manufacturing plant. Only one painting line is available, making it a single-machine scheduling problem. A passage into the painting line is referred to as an operation, while the set of operations required for completion of an order is called a job. Different options exist for a hubcap going through the painting line. If it is unicolor, it is painted once and can go directly to the finished

* Supported by the Auvergne Rhône-Alpes region.

product inventory to await shipping. If it is bicolor, it receives its first coating and is sent to dry in the intermediary inventory for a minimum period L, then receives its second coating and is sent to the finished product inventory. Shipping must occur before a given due date, since the Make-To-Order setting allows no lateness. This particular problem is called a coupled-tasks scheduling problem [17] and has been proven to be NP-Hard [14]. Blazewicz et al. [3] provide a survey of research on coupled-tasks scheduling problems, as well as a list of important results for the most common variants and subproblems. The computation times being too large in real-life situations, metaheuristics such as PSO and SA [13], tabu-search [4,12], as well as various heuristics [1,5] have been used to solve coupled-tasks scheduling problems. Genetic algorithms have also been extensively used to solve scheduling problems, including problems involving reentrance characteristics which are similar to the coupled-tasks problems.

The objective is to optimize the daily schedule to minimize both the quantity held in inventory and the environmental impact of the paint sludge generated in the painting line when the color changes, represented by the number of setups. To deal with these two conflicting objectives, Le Hesran et al. [10] propose the use of a Genetic Algorithm with a weighted-sum method to obtain a Pareto front of alternative solutions. Results show that this algorithm has difficulty obtaining the entirety of the Pareto front : drastic improvements are possible regarding both the objective functions optimization and computing time required.

To this end, the contribution of this paper is a new multi-objective GA based on the NSGA-II framework [6] to solve the single machine couple-tasks scheduling problem described above. Its structure and mechanisms are described in section 2, and numerical experiments are carried out in section 3 followed by conclusions and perspectives for future work.

2 Multi-objective GA based on NSGA-II

Figure 1 shows the global structure of the proposed multi-objective GA based on NSGA-II.

2.1 Chromosome representation

A chromosome represents a sequence of operations, its size being equal to the number of jobs times the maximum number of operations per job. Since not all jobs have the same number of operations, dummy operations with processing time zero are added to keep the chromosome size constant. A gene's position corresponds to the job it belongs to and its order within this job. The value of a gene represents its rank in the global operations sequence. Figure 2 shows an example solution on a Gantt chart, with i the job and j the operation. As an example, operation 1 of job 9 is processed first, while operation 2 of job 1 is processed sixth, and operation 2 of job 8 is a dummy operation. Table 1 shows the corresponding chromosome, Seq giving the operations sequence.

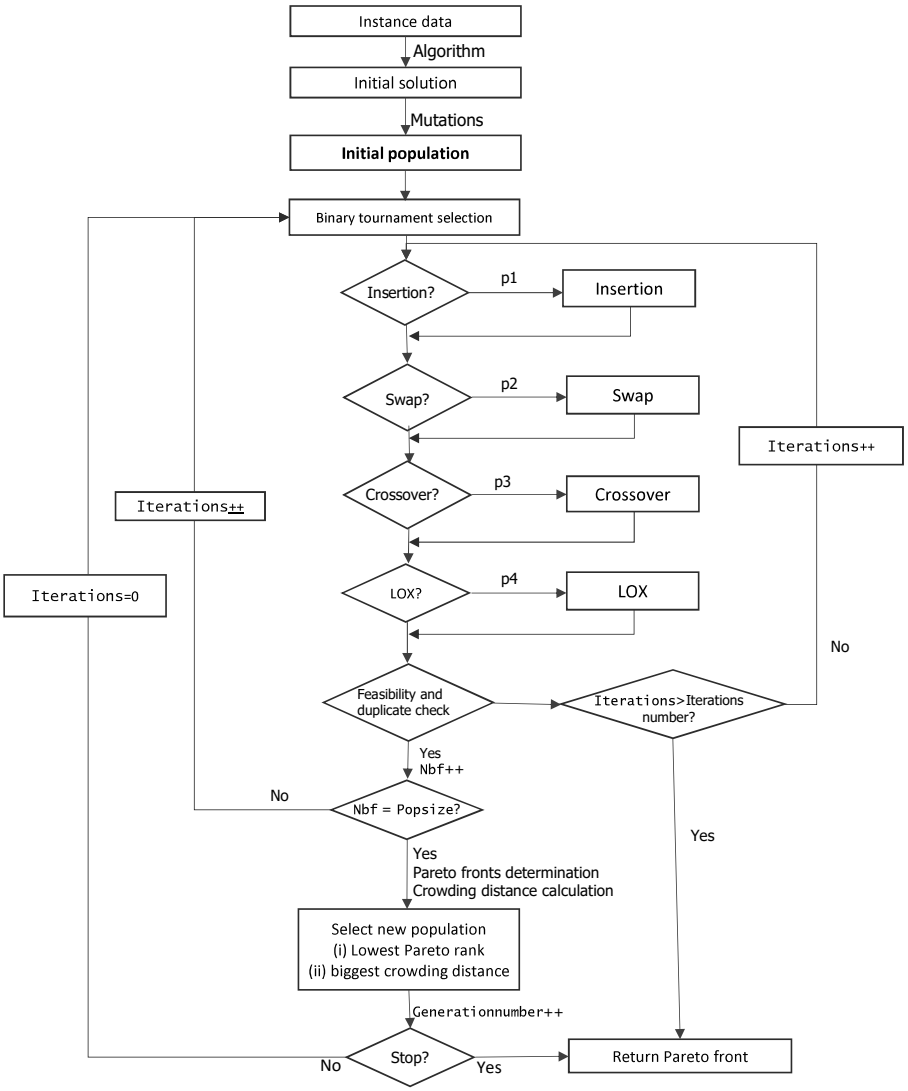

Fig. 1: NSGA-II structure representation

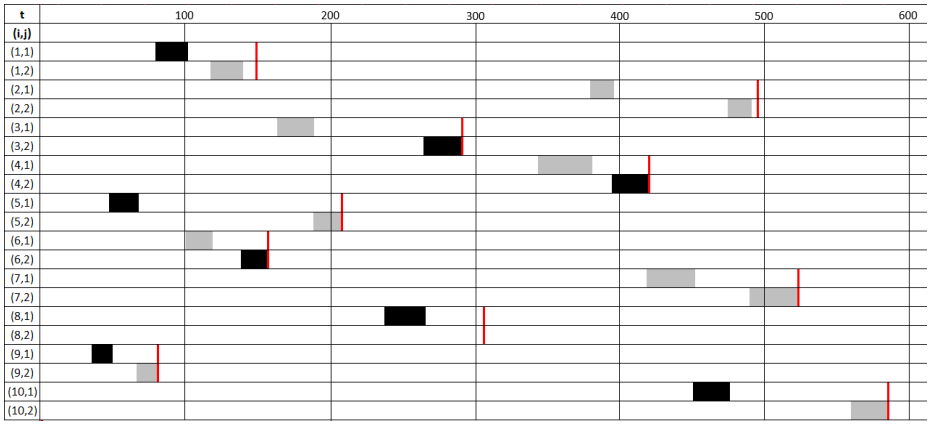

Fig. 2: Gantt chart of a schedule with ten jobs

Table 1: Associated chromosome sequence

i	1		2		3		4		5		6		7		8		9		10	
j	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Seq	4	6	13	17	8	11	12	14	2	9	5	7	15	18	10	20	1	3	16	19

Ranks are obtained using the fast non-dominated sorting algorithm [7] for all candidate solutions. In this strategy, the Pareto dominance relationship is used to assign each solution a rank based on a domination counter. All solutions are compared, and all the non-dominated ones are assigned rank 1. They are then removed from the current population, and the process is repeated with an incremented rank number, until all solutions have been assigned a rank. This provides a set of Pareto fronts $\mathcal{F}$, where all solutions of front $\mathcal{F}_k$ dominate the solutions of front $\mathcal{F}_{k+1}$.

In order to avoid the clustering of solutions, a crowding-distance comparison method is used. This crowding distance $\mathcal{I}[i]$ of a solution i is based on the neighboring points surrounding it, according to the different objectives. It is calculated as: $\mathcal{I}[i] = \sum_{o \in \mathcal{O}} = \frac{z^{\mathrm{o}}_{i+1} - z^{\mathrm{o}}_{i-1}}{z^{\mathrm{o}}_{\max} - z^{\mathrm{o}}_{\min}}$ where $\mathcal{O}$ is the set of objectives, z^{o}_{i+1} and z^{o}_{i-1} the objective value of both neighboring solutions for the o^{th} objective, and $z^{\mathrm{o}}_{\max}$ and $z^{\mathrm{o}}_{\min}$ the maximum and minimum values for objective $o \in \mathcal{O}$. A crowded comparison operator is then used to discriminate between different solutions with the following logic: if a solution is ranked lower than another, it is preferred to its counterpart. If two solutions have the same rank, the one with the biggest crowding distance is preferred.

2.2 Initialization

Based on the instance data, a single initial solution is created. An algorithm sorts the jobs by increasing due date. The operations of jobs with the lowest due dates are scheduled first, and operations of other jobs can be introduced whenever the job with the lowest due date is in the drying inventory. Once the initial solution is created, two mutation operators are applied in order to generate a sufficient number of new offspring. Any unfeasible solution generated (where tasks exceed their due dates) is immediately discarded and another one created to replace it. These constitute the initial population introduced into the GA.

2.3 Iterations

A pair of chromosomes is selected, and has a probability p_1 of being subjected to the insertion operator, that picks a random gene and inserts it somewhere else in the chromosome. It means that any given pair of chromosome can be subjected to either zero, one or two mutations. It is then subjected to the swap operator [16] (the swap picks two random genes within the chromosome and exchanges them; each chromosome is mutated independently) with a probability p_2. The resulting chromosomes then have a probability p_3 of being subjected to a standard two-point crossover [16] (as parents), followed by a probability p_4 of being subjected to the Linear Order Crossover (LOX) [15]. The standard two-point crossover chooses two random genes in the first parent and swaps them with the corresponding genes of the second parent. The LOX operator also chooses two random genes as crossover points: the partial sequence contained between them is transmitted to the offspring, the rest being filled with the missing genes from the other parent starting from the beginning of the chromosome. This operator has the merit of keeping a part of the first parent intact, as well as the relative order from the second one, which is important in a problem where due dates severely constrain the ordering possibilities.

If those new chromosomes are feasible, they are kept in the offspring generation, and a counter called `Nbf` is incremented. If more offspring need to be generated to complete the population, the iteration counter `NbIterations` is incremented and a new pair of parents is selected and submitted to the operators. If the iteration counter reaches `Iteration number` before a new population has been created, the algorithm stops and the best current solutions are returned.

Once a number of offspring equal to the population size have been accepted, both the parent and offspring populations are combined and the Pareto-rankings determined. The population replacement strategy is applied, the new population is created, and the iteration counter is reset. This process goes on until the number of generations reaches `threshold` and the algorithm stops.

3 Numerical experiments and results

The algorithm is coded in C++ ; all experiments are carried out using an Intel i5 6200 2.3 GHz processor with 8 GB of RAM.

3.1 Instances and GA parameters definition

The proposed NSGA-II algorithm is used on the instances provided by [10]. 80-20, 50-50 and 20-80 configurations are considered: the X-Y configuration consists of X% of jobs with one operation and Y% of jobs with two operations. Experiments are run on instances of each configuration with $n = 10$ jobs and 80-20 instances with $n = 30$ jobs. Instances of 10 jobs allow us to compare optimal results from the exact approach with those of the GA, while 30 jobs instances are closer to industrial size instances of around 100 jobs.

Table 2 details the chosen values of the GA parameters, obtained through a Taguchi experimental design.

3.2 Interpretation of the Pareto front

The Pareto front [2] provides the decision-maker with alternative solutions that represent the variety of possible results. Its size is limited by the maximum number of possible color changes. Although every Pareto point is an optimal solution, all of them might not be suited to a practical use ; thus, four key points are extracted for each instance.

Two extreme points $(z^{\min}_{\text{inventory}}, z^{0}_{\text{setup}})$ and $(z^{0}_{\text{inventory}}, z^{\min}_{\text{setup}})$, represent the cases where the decision-maker wishes to minimize one objective in priority. The ideal point $(z^{\min}_{\text{inventory}}, z^{\min}_{\text{setup}})$ is defined using the two optimum values of these points, i.e. the minimum quantity of inventory and minimum number of setups achievable. The coordinates of each point z^{it} are normalized using the formula $z^{\text{normal}} = \frac{z^{\text{it}} - z^{\min}}{z^{0} - z^{\min}}$ for both $z_{\text{inventory}}$ and z_{setup}. This norm provides new values between 0 and 1 to compare values of different nature and order of magnitude (inventory, number of setups).

Using these normalized values, the euclidean distance of each point to the ideal point is calculated. The solution located at the minimal distance from the ideal point $(z^{\min}_{\text{inventory}}, z^{\min}_{\text{setup}})$ is chosen as the trade-off point $z^{\text{trade-off}}$, which represents the best compromise in terms of number of setups reduction versus increase in inventory.

z_{percent} is the point with the highest difference between setup percentage reduction and inventory percentage increase. This point aims at providing an attractive option for decision-makers that wish to improve their environmental impact without affecting their inventory costs negatively. An example of Pareto front with its important points is shown in Figure 3.

3.3 Results

Table 3 shows that the multi-objective GA reaches the optimal solution a majority of the time for both the $z^{\min}_{\text{inventory}}$ and z_{percent} points, with average gaps not exceeding 3.3%. While the average gap appears to be slightly higher for the NSGA-II algorithm than for the weighted-sum one, optimal solutions are reached more often.

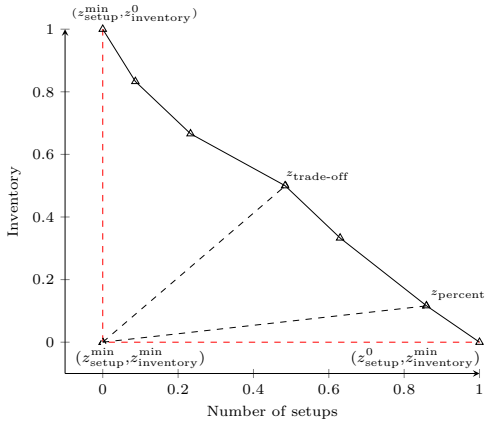

Fig. 3: Example of a Pareto front

Table 2: GA parameter values

	30-job instances	100-job instances
Population size	30	100
Swap rate	0.8	0.8
Insertion rate	0.5	0.8
Crossover rate	0.3	0.3
LOX rate	0.1	0.05
Threshold	1000	2500
Iteration number	1000	2500

The complete results for $z_{\text{trade-off}}$ and z_{percent} are shown in Tables 4 and 5 respectively. The multi-objective GA provides accurate results and manages to cover the majority of the Pareto front. As an example, for the 20-80 configuration an average Pareto front size of 4.6 is observed versus 5.6 for the MILP results. As a comparison, the weighted-sum GA by Le Hesran et al. [10] obtained an average size of 3.85 for the Pareto front of the same configuration.

In addition to a larger number of Pareto points obtained, results from the multi-objective GA are better than those from the weighted sum GA, obtaining a lower number of setups for the trade-off points and a lower inventory for the percent points on all configurations except for the 80-20 one.

Table 6 shows a direct comparison of the $z^{\text{trade-off}}$ and z^{percent} points obtained using the MILP (solved with CPLEX) and weighted sum GA from Le Hesran et al. [10], and our new multi-objective GA on the 30-job instances, most of which cannot be proved optimal by the MILP model within its time limit of 1800 seconds. The MILP results dominate both the weighted-sum and multi-objective GAs. However, their computation requires upwards to half an hour per point, which is not suited for practical applications on large instances. On the other hand, the multi-objective GA largely outperforms the MILP and is better than the weighted-sum GA regarding computation time.

Table 3: Comparison on key points (10-job instances)

		Weighted sum GA [10]		NSGA-II : $z^{\text{inventory}}_{\text{min}}$		NSGA-II : z_{percent}	
n	Config.	Average gap (%)	Nb opt/ nb total	Average gap (%)	Nb opt/ nb total	Average gap (%)	Nb opt/ nb total
10	80-20	1,0%	27/30	3,0%	27/30	3,3%	28/30
10	50-50	1,8%	24/30	1,0%	25/30	1,0%	25/30
10	20-80	1,0%	20/30	2,0%	21/30	1,2%	21/30

Table 4: Characteristics of the $z^{\text{trade-off}}$ point (standard deviation in parenthesis)

		Weighted sum GA [10]				NSGA-II			
n	Config.	$z^{\text{trade-off}}_{\text{setup}}$	$z^{\text{trade-off}}_{\text{inventory}}$	CPU (s)	Pareto size	$z^{\text{trade-off}}_{\text{setup}}$	$z^{\text{trade-off}}_{\text{inventory}}$	CPU (s)	Pareto size
10	80-20	3.1	3056	0.45 (1.2)	3.34	3.1	3084	14.1 (14.1)	2.97
10	50-50	3.8	5560	215 (539)	4.55	4.1	4584	12.0 (8.7)	3.75
10	20-80	4.4	7150	638 (737)	5.60	5.0	6374	13.1 (9.2)	4.57
30	80-20	8.9	16764	1714 (384)	9.01	9.1	21461	65.7 (79.1)	6.55

Table 5: Characteristics of the z^{percent} point (standard deviation in parenthesis)

		Weighted sum GA [10]				NSGA-II			
n	Config.	$z^{\text{percent}}_{\text{setup}}$	$z^{\text{percent}}_{\text{inventory}}$	CPU (s)	Pareto size	$z^{\text{percent}}_{\text{setup}}$	$z^{\text{percent}}_{\text{inventory}}$	CPU (s)	Pareto size
10	80-20	3.9	2215	0.10 (0.67)	3.34	3.8	2350	14.1 (141.1)	2.97
10	50-50	4.3	4385	118 (371)	4.55	4.2	3997	12.0 (8.7)	3.75
10	20-80	5.4	5852	595 (762)	5.60	5.4	6000	13.1 (9.2)	4.57
30	80-20	11.0	13179	1680 (392)	9.05	13	14497	65.7 (79.1)	6.55

Results from the multi-objective GA are closer to the MILP ones than those of the weighted-sum GA, providing a lower average number of setups for the $z^{\text{trade-off}}$ point, and lower average inventory for the z^{percent} one. Since $z^{\text{trade-off}}$ tends to be located on the left of the Pareto front (meaning a lower number of setups) and z^{percent} on the right (meaning a lesser inventory), results from the multi-objective GA are closer to those of the MILP than those from the weighted-sum GA.

Table 6: Points of interest for 30-job instances and 80-20 configuration

	Solving method	z_{setup}	$z_{\text{inventory}}$	Setup % reduc.	Inventory % inc.	CPU time (s)	Pareto size
$z^{\text{trade-off}}$	MILP	8.9	16764	38.5 (16.1)	54.8 (73.2)	1714.0 (384)	9.05
	Weighted-sum	11.5	18518	22.4 (16.0)	25.5 (35.6)	95.0 (55.5)	5.35
	Multi-obj	9.1	21461	34.2 (15.0)	62.5 (39.4)	65.7 (79.1)	6.55
z^{percent}	MILP	11.0	13179	25.9 (13.7)	12.3 (8.9)	1680.0 (392.0)	9.05
	Weighted-sum	12.8	16537	15.2 (17.1)	5.9 (9.3)	88.4 (58.0)	5.35
	Multi-obj	13.0	14497	9.6 (12.2)	4.4 (7.3)	65.7 (79.1)	6.55

In order to simulate a real life situation and show its performance on bigger instances, the proposed multi-objective GA is solved on 10 instances of 100 jobs (the size of a daily schedule) with the 80-20 configuration. Results for the z^{tradeoff} and z^{percent} point are available in Table 7. The genetic algorithm is more efficient on large instances, which are those which are important for manufacturers. Alternative daily schedules can be obtained in one hour and a half on average, which is an acceptable time-frame for a practical use. While percentages seem lower in both waste reduction and inventory increase, they still remain significant

Table 7: Points of interest for 100-job instances and 80-20 configuration

	n	Config.	z_{setup}	$z_{\text{inventory}}$	Setup % reduc.	Inventory % inc.	CPU time (s)
$z^{\text{trade-off}}$	100	80-20	34.2	253920	41.1 (8.5)	149.2 (98.2)	5718
z^{percent}	100	80-20	51.6	135431	11.1 (9.6)	4.3 (3.87)	5718

with such large quantities. The possible improvements are largely dependent on the instance, and can vary greatly as shown by the high standard deviations. The provided schedules are very efficient for some particular instances.

4 Conclusion

This paper tackles the issue of a single-machine scheduling problem with coupled-tasks, aiming at reducing waste generation due to setups and costs induced by inventory under the constraint of due dates. It proposes a new multi-objective genetic algorithm based on the NSGA-II structure. The new algorithm is more efficient in mapping the Pareto front than a GA using weighted sums. Thanks to evolutionary mechanisms designed for obtaining multiple alternative solutions, this algorithm obtain solutions that are both more optimized and more evenly spread out in the solutions space, improving the solutions provided to decision-makers. Computation times are also improved, as the multi-objective GA is able to map a Pareto front in a single run, as opposed to the weighted-sum one requiring multiple runs, and a larger number of Pareto points is obtained.

Alternative solutions can thus be obtained rapidly, providing the decision-makers with different options depending on their priorities and current situation. The z_{percent} point in particular is shown to be useful for decision-makers. These improved results highlight the potential of such waste-reducing schedules for both economic and environmental objectives.

Several perspectives can be considered for this study. Calculating lower bounds would help to assess the performance of the GA for large instances. The model could be extended to other types of workshops (e.g. multi-machines environments) or product types (e.g. multiple colors and coatings) as studied in Gould and Colwill [9]. From an environmental perspective, assessing the environmental impact of paint sludge production, and the potential benefits of reducing waste production could also motivate practitioners to implement such schedules. Indeed, if the actual cost of waste management was assessed, alternative schedules reducing waste generation could prove to be overall economically beneficial to companies.

References

1. Amrouche, K., Boudhar, M., Bendraouche, M., Yalaoui, F.: Chain-reentrant shop with an exact time lag: new results. International Journal of Production Research **55**(1), 285–295 (2017)

2. Blasco, X., Herrero, J.M., Sanchis, J., Martínez, M.: A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization. Information Sciences **178**(20), 3908–3924 (2008)
3. Blazewicz, J., Pawlak, G., Tanas, M., Wojciechowicz, W.: New algorithms for coupled tasks scheduling : a survey. RAIRO - Operations Research **46**(4), 335–353 (2012)
4. Condotta, A., Shakhlevich, N.V.: Scheduling coupled-operation jobs with exact time-lags. Discrete Applied Mathematics **160**(16-17), 2370–2388 (2012)
5. Courtad, B., Baker, K., Magazine, M., Polak, G.: Minimizing flowtime for paired tasks. European Journal of Operational Research **259**(3), 818–828 (2017)
6. Deb, K., Agarwal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. International Conference on Parallel Problem Solving From Nature pp. 849–858 (2000)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm : NSGA-II. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)
8. Giret, A., Trentesaux, D., Prabhu, V.: Sustainability in manufacturing operations scheduling: A state of the art review. Journal of Manufacturing Systems **37**, 126–140 (2015)
9. Gould, O., Colwill, J.: A framework for material flow assessment in manufacturing systems. Journal of Industrial and Production Engineering **32**(1), 55–66 (2015)
10. Le Hesran, C., Agarwal, A., Ladier, A.L., Botta-Genoulaz, V., Laforest, V.: Reducing waste in manufacturing operations: bi-objective scheduling on a single-machine with coupled-tasks. International Journal of Production Research pp. 7130–7148 (2019)
11. Le Hesran, C., Ladier, A.L., Botta-Genoulaz, V., Laforest, V.: A methodology for the identification of waste-minimizing scheduling problems. Journal of Cleaner Production **246**, 119023 (2019)
12. Li, H., Zhao, H.: Scheduling coupled-tasks on a single machine. Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling pp. 137–142 (2007)
13. Meziani, N., Boudhar, M., Oulamara, A.: PSO and simulated annealing for the two-machine flowshop scheduling problem with coupled-operations. European Journal of Industrial Engineering **12**(1), 43–66 (2018)
14. Orman, A.J., Potts, C.: On the complexity of coupled-task scheduling. Discrete Applied Mathematics **72**(96), 141–154 (1997)
15. Portmann, M.C.: Genetic algorithms and scheduling: a state of the art and some propositions. Proceedings of the Workshop on Production Planning and Control **9**(11), 1–24 (1996)
16. Sevaux, M., Dauzère-Pérès, S.: Genetic algorithms to minimize the weighted number of late jobs on a single machine. European Journal of Operational Research **151**(2), 296–306 (2003)
17. Shapiro, R.D.: Scheduling coupled tasks. Naval Research Logistics Quarterly **27**(3), 489–498 (1980)