Dynamic Rescheduling in Energy-Aware Unrelated Parallel Machine Problems

Sergio Ferrer[1], Giancarlo Nicolò[1], Miguel A. Salido[1](✉), Adriana Giret[2], and Federico Barber[1]

[1] Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Valencia, Spain
{serfers,giani,msalido,fbarber}@dsic.upv.es

[2] Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Valencia, Spain
agiret@dsic.upv.es

Abstract. Advance in applied scheduling is a source of innovation in the manufacturing field, where new results help industrial practitioners in production management. The literature of rescheduling problems for single-objective optimization is well study, while there is a lack of extensive studies for the case of rescheduling for multi-objective optimization, especially for energy aware scheduling. This paper extends a previous work over an energy aware scheduling problem, modelled from a real industrial case study, extending its manufacturing environment to a dynamic one. To this end, two rescheduling techniques are developed to tackle machines disruptions (greedy-heuristic and meta-heuristic). They are compared to existing approach thought manufacturing environment simulations. The results give insight to improve the production management in terms of rescheduling quality and computational time.

Keywords: Energy-aware rescheduling · Metaheuristics
Sustainable manufacturing

1 Introduction

Applied scheduling is the field of study that aims to fill in the gap between the scheduling theory and its application to real scenario, giving to production managers new and innovative solutions to face current trend in their domain of work. Of particular interest for the applied scheduling is the manufacturing domain, where one of the main application of the scheduling problems is in the context of optimizing the manufacturing line production. In this scheduling environment, the usual assumption is to analyse the problem as static, meaning that all the information of the scheduling problem does not change over time. While this assumption can be useful for theoretical results, when dealing with real scenario this assumption limits the possibility to concretely apply the developed static scheduling technique, because the only way to face environment disruption is to solve a new scheduling problem with substantial computational cost and lack of system responsiveness. To improve this situation the environment has to be considered as dynamic and faced with predictive-reactive scheduling strategies. In this way, a (predictive) scheduling solution is produced and when a disruption occurs a reactive strategy (rescheduling) is carried out to generate a new feasible solution.

Most of the literature works over rescheduling problems address scheduling environment in which the main objectives are related to production objectives (completion time, tardiness, lateness, etc.), while current trends in manufacturing are taking in to consideration more complex objectives, where environmental issues have to be addressed through the consideration of energy consumption in the optimization process (i.e. energy-aware scheduling problem) leading to the creation of the sustainable scheduling field of study. Even though the importance of sustainable scheduling is broadly affirmed through the scientific community [1,2,7], most of the current literature in this field is addressing static problems (predictive scheduling), while literature works related to dynamic environments are very few in comparison to the variety of possible dynamic rescheduling environments determined by the combinatorial combination between scheduling environments, optimization functions, constraints and analysed disruptions. [11] tackles a bi-objective single machine batch scheduling problem where the first objective is to minimize the makespan and the second is to minimize the total energy costs, by considering both the machine utilization and the economic cost. An integer programming model is proposed and then, an exact ϵ-constraint method is adapted to obtain the exact Pareto front. In [8], a simple multi-agent system model is proposed to decompose an energy-aware scheduling problem into smaller subproblems. In this approach, each agent solves a subproblem by using a Mixed Integer Linear Programming (MILP) model and the results are combined to obtain a global solution. A similar idea is proposed in [5] where a set of solving techniques are compared and job features are studied in order to tackle energy-aware scheduling problems.

The proposed work is an extension of a well-studied scheduling scenario derived from analysing a manufacturing real case of an injection moulding plastic industry [6]. This problem can be considered as an energy-aware unrelated parallel machine scheduling problem with machine-dependent energy consumption and sequence-dependent setup time. The current literature taking in consideration the problem under analysis, models it as a static environment (predictive scheduling), and no previous works have been developed to manage this problem as a dynamic one.

2 Description of the Scheduling Problem

The scheduling problem under analysis is the energy-aware unrelated parallel machine scheduling problems with machine-dependent energy consumption and sequence-dependent setup time. It was firstly introduced in [6] and better studied in [3–5,8]. In this problem, a set of orders, represented by jobs, has to be scheduled on a set of unrelated parallel machines. Each job has associated to its specific temporal and cost features: release date, due date and penalty cost. A job can be processed by one or more machines, where processing time and energy consumption of the job depends by the machine assigned to process it. Between the execution of two consecutive jobs over the same machine a setup time is needed, where its value depends on the jobs sequence and machine assigned to them. The problem is formally described as Mixed Integer Linear Programming model and the mathematical formulation is described in [6].

The problem is considered as multi-objective since there are three measures to be minimized, which are expressed as objective functions: the total weighted tardiness of the jobs $TT(s)$, the total energy consumption $EN(s)$, and the total setup time $ST(s)$. The solution s^* can be obtained by minimizing a 3-dimensional objective function:

$$s^* = \arg\min_{s \in S}[TT(s), EN(s), ST(s)] \tag{1}$$

where S denotes the feasibility space for the problem solution space.

3 Definition of Incidence

An “incidence” is an unpredictable event that transforms the original schedule into a non-executable plan, such as: breakdowns on machines, power cuts, work accidents, etc. The representation for an incidence is independent of its origin and nature. This means that the representation of an incidence only has information about its technical characteristics ignoring the reason for the incidence. An incidence i is represented by mean of 4 parameters: an identification number for the incidence, the point when the incidence occurs, the machine that is affected by this incidence and the time necessary to recover the affected machine. So, an incidence can be formally defined as:

$$i = [id_i, st_i, m_i, tts_i]$$

where

- id_i: an identification number for the incidence
- st_i (starting time): the time point when the incidence i starts
- m_i (machine): an integer representing the ID of the affected machine by i
- tts_i(time to solve): the necessary time to solve the incidence i. It has to be estimated by a human expert. From st_i to $st_i + tts_i$, the machine m will be inoperative.

3.1 A Greedy Rescheduling (GR) Algorithm

In contrast to a baseline technique, based on the propagation of the incidence along the schedule on the affected machine using a Right-Shift technique [10], a new reallocation approach is proposed. It analyzes every affected job to determinate if it has to be moved to another machine and, if applicable, determine what is the best machine and in which position to introduce this job. An "affected job by an incidence" is defined as "Every job scheduled in the same machine in which the incidence occurs and its execution time is scheduled after the incidence starts". Formally, the set of affected jobs (AJ) by an incidence i is defined as:

$$AJ = \{j : y_{jm_i} = 1 \wedge st_j > st_i\} \tag{2}$$

For each affected job, it has to be analyzed if it is convenient to move it to another machine or is more productive to wait until the machine is repaired. Thus, the Best Available Machine (BAM) for each job is determined. If the BAM of a job is the machine in which it is previously scheduled, the job will not be moved, otherwise the job will be moved to its BAM.

Given a job j already scheduled in a machine k', we need to do the next calculations to determinate its BAM. First, we need to study how the rest of machines improve or worsen their performance when receiving job j, s.t. $\forall j \in AJ$:

$$\Delta_k = F(S_k + j) - F(S_k), \forall k \neq k' \in M_j \tag{3}$$

where:

- M_j: is the set of machines that can execute job j (see Sect. 2).
- S_k: is the schedule of machine k. S_k is a sorted list including all the jobs assigned to machine k. Formally:

$$S_k = \{j : y_{jk} = 1\} \tag{4}$$

- $F(S_k)$: is the multi-objective value resulting from the application of the evaluation function on the schedule S_k.
- $S_k + j$: is the resulting schedule from inserting the job j into schedule S_k in the best position inside S_k (determined by brute force). In this case, the symbol '+' represents the operator of adding a new job into an existing schedule into the best position inside it.
- Δ_k: represents how much the evaluation of the schedule S_k is worsened when job j is added into it.

Each Δ_k stores how much each machine k is deteriorated when receiving the job j, so it can be selected the best machine k* with lower value:

$$k^* = argmin_{k \neq k'} \Delta_k \tag{5}$$

In this way, k^* is selected as the best machine to receive job j, but j is already in a machine k' that could event be better than k^*. To evaluate how job j fits on k', j is taken out from S'_k and an evaluation of how the machine k' performances without j is carried out by:

$$\Delta_{k'} = F(S_{k'}) - F(S_{k'} - j) \tag{6}$$

where:

– $S_{k'} - j$: is the resulting schedule from extracting the job j from schedule S_k.

So, $\Delta_{k'}$ stores how much machine k' deteriorates its performance due to job j. So, finally, the BAM for a given job j can be defined as:

$$BAM = k \in \{k', k^*\} : k = argmin \Delta_k \tag{7}$$

If BAM is *k'* means that j is already in its best machine and it does not need to be reallocated in other machine to produce better results. By contrast, if BAM is k^* means that moving job j into k^* will produce a better performance ($\Delta_{k^*} < \Delta_{k'}$).

Given this BAM definition, Algorithm 1 presents the pseudo-code used to manage the incoming incidences into an existing schedule:

Algorithm 1. Rescheduling algorithm

```
INPUT: solution S and incidence I
OUTPUT: solution S with I inside and affected jobs by I reallocated
1: id ← I[0]; st ← I[1]; m ← I[2]; tts ← I[3]; # see section 3
2: affectedJobs ← $\{j : y_{jm} = 1 \wedge st_j > st_{id}\}$
3: Right-Shift(I,S) # Inserts incidence.
4: for job ∈ affectedJobs do {
5: $\Delta_k = F(S_k + j) - F(S_k), \forall k \neq m \in M_j$
6: $k^* = argmin_{k \neq m} \Delta_k$
7: $\Delta_m = F(S_m) - F(S_m - j)$
8: $BAM = k \in \{m, k^*\} : k = argmin \Delta_k$
9: if (BAM ≠ m) {moveJob(job, BAM) } }
```

Line 3 implements the Right-Shift repairing technique [10]. That means to apply the baseline before rescheduling starts in order to take into account the delay introduced by the incidence when evaluating the reallocation possibilities for each job. In line 9, the *moveJob* function moves the job to its BAM, both elements received as parameters. The position for the job inside its BAM is selected by brute-force: trying one by one all the possible places. Formally, the index i to introduce the job j into its BAM is selected as:

$$i = argmin_{0 \leq i < |S_{BAM}|} F(S_{BAM} + j, i) \tag{8}$$

where:

– S_{BAM} is the current schedule for machine BAM.
– $F(S_{BAM} + j, i)$ is the application of the evaluation function to the schedule S_{BAM} with the job j introduced in the position i by Right-Sift technique.

3.2 A Genetic Algorithm with GR Initialization (GA+GR)

In this section, a Genetic Algorithm (GA) for rescheduling a solution after an incidence is proposed. The main idea is to run a GA only with the affected jobs

(AJ) (Eq. 2) in order to preserve the solution before the incidence and trying to reschedule all the jobs after the incidence with the objective of minimizing the impact of the disruption. Notice that incidences are supposed to appear in the production environment while the scheduling is being already executed by the machines, therefore the decision to keep the previous jobs without rescheduling them is not really a decision but a constraint of the environment. A competitive GA to tackle this problem has been used in [5], where the GA is embedded in a multi-agent system. Taking this GA as a starting point and our definition of AJ, some modifications to them have been carried out:

- In order to reschedule the whole set of jobs that can be moved after the incidence, a new definition of AJ is presented: $AJ = \{j : st_j > st_i\}$ With this new definition, the whole set of jobs executed after the incidence is selected for the rescheduling, independently of the machine in which the job is scheduled, which allows the GA to explore a larger search space than the GR algorithm. The main idea behind this choice is to be able to compare two options of general rescheduling approaches: preserve as much as possible the previous solution (stability) by rescheduling only the jobs of the affected machine (done by GR) or to reschedule the whole solution after the incidence (done by this modification of GA).
- The *fitness* function of the GA is modified to contemplate the previous schedule to the solutions given by the GA in order to calculate the multi-objective function of the complete schedule. Notice that GA is executed only with the AJ as input, so the schedule given as solution only contains the AJ. Nevertheless, the evaluation of the solution needs the previous schedule (the one containing the *non affected jobs*) in order to decide the quality of a solution. In this way it is achieved that the GA solves a sub-problem of scheduling (only the problem containing the AJ) but it evaluates its solutions with respect the complete schedule
- The initialization of the GA is modified to create an initial population which includes the solution given by the GR algorithm. The incidences will be solved accumulatively (see Sect. 4), which implies that as incidences appear, the GR and the GA may have different solutions for each incidence. The GA executes the GR when initialized, which provides a greedy solution. This greedy solution is used as an individual of the initial population of the GA. This decision is given to guide the GA search with the previous solutions with allows to study if these solutions are replaced during the process by another better solutions or they are preserved during the whole process.

4 Evaluation

To evaluate the proposed system, a set of incidences was generated. The incidences were randomly generated over a machine and before the 75% of the total time in the schedule. This decision avoids the generation of incidences at the end of the schedule because when an incidence is introduced at the end of the schedule the set AJ may be empty or composed of few jobs. This situation does not show an objective comparison of the different techniques. Furthermore, the time needed to solve the incidence was randomly generated between 10% of the longest job assigned to the affected machine and its total processing time.

The incidences were solved accumulatively, which means that the solution of the incidence *i* was the schedule in which the incidence *i+1* was introduced. By accumulating the incidences in the schedule, it can be compared how the different techniques absorb the incoming incidences as new disruptions arrive into the system.

The instances of the problem were taken from the benchmark proposed in [9] which presents 4 classes of instances depending on the number of jobs and machines per incidence: j30 with 4 machines and 30 jobs, j50 with 6 machines and 50 jobs, j100 with 10 machines and 100 jobs, and j250 with 20 machines and 250 jobs. The time needed to execute each technique is fixed to: baseline = 0.2 s, GR = 2.3 s, GA = 30 s and GA+GR = 30 s.

To test and compare the four techniques, a subset of the 10 largest instances per class were selected (40 instances). They were solved with the system proposed in [3] and the solutions given by this system were used as the input for our system.

Figures 1, 2, 3 and 4 show the results of applying the four techniques to each class of the problem. As shown on Fig. 1a, for small instances the GR algorithm obtained the best results, but as problem gets larger, the GA+GR algorithm got a better behavior than the rest of the techniques. It is also interesting to compare GA vs GA+GR since GA+GR maintained a better performance that GA in all cases, although both were very similar. This is due to the fact that the only difference between them is that GA+GR includes the greedy algorithm GR in each step. This difference could come from the fact that GA was designed in [5] to have a good performance in solving (not rescheduling) the instance and with a larger timeout (10 min) than the timeout fixed for the rescheduling tasks.

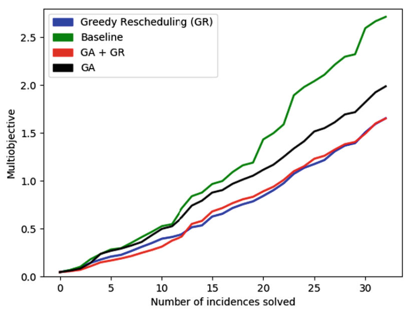

Fig. 1. Results on j30

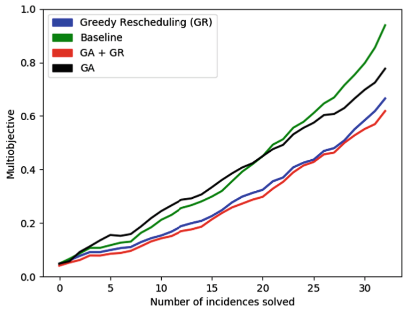

Fig. 2. Results on j50

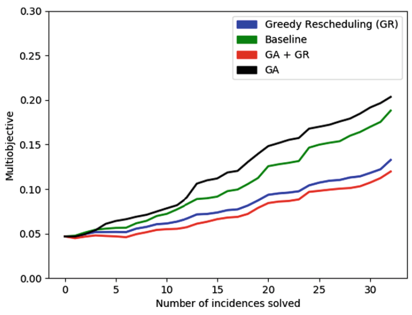

Fig. 3. Results on j100

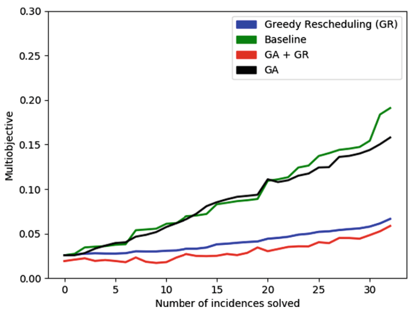

Fig. 4. Results on j250

5 Conclusions and Future Works

In this paper, two different techniques for solving the rescheduling on the unrelated parallel machine problem have been proposed. The proposed GR algorithm obtained the best results for small instances of the problem, while the GA+GR algorithm obtained a better behavior when instances are larger leading to useful insight over which technique to use regarding the problem size. As future work, it is proposed to manage the incidences by a match-up technique to recover the original solution as soon as possible in order to improve stability of solutions.

Acknowledgements. The paper has been partially supported by the Spanish research project TIN2016-80856-R and TIN2015-65515-C4-1-R, and the ACIF/2017 Program of the Conselleria d'Educació (Generalitat Valenciana).

References

1. Bruzzone, A., Anghinolfi, D., Paolucci, M., Tonelli, F.: Energy-aware scheduling for improving manufacturing process sustainability: a mathematical model for flexible flow shops. CIRP Ann.-Manuf. Technol. **61**(1), 459–462 (2012)
2. Dai, M., Tang, D., Giret, A., Salido, M.A., Li, W.D.: Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robot. Comput.-Integr. Manuf. **29**(5), 418–429 (2013)
3. Nicolo, G., Ferrer, S., Salido, M.A., Giret, A., Barber, F.: A multi-agent framework to solve energy-aware unrelated parallel machine scheduling problems with machine-dependent energy consumption and sequence-dependent setup time. In: ICAPS 2018, Delft, The Netherlands (2018)
4. Nicolò, G., Salido, M., Giret, A., Barber, F.: Assessment of a multi agent system for energy aware off-line scheduling from a real case manufacturing data set. In: COPLAS 2016, p. 35 (2016)
5. Nicolo, G., Salido, M.A., Ferrer, S., Giret, A., Barber, F.: A multi-agent approach using dynamic constraints to solve energy-aware unrelated parallel machine scheduling problem with energy-dependent and sequence-dependent setup time. In: COPLAS 2017, pp. 31–37 (2017)

6. Paolucci, M., Anghinolfi, D., Tonelli, F.: Facing energy-aware scheduling: a multi-objective extension of a scheduling support system for improving energy efficiency in a moulding industry. Soft Comput. **21**, 1–12 (2015)
7. Plitsos, S., Repoussis, P.P., Mourtos, I., Tarantilis, C.D.: Energy-aware decision support for production scheduling. Decis. Support Syst. **93**(Suppl. C), 88–97 (2017). http://www.sciencedirect.com/science/article/pii/S016792361630166X
8. Tonelli, F., et al.: Assessment of mathematical programming and agent-based modelling for off-line scheduling: application to energy aware manufacturing. CIRP Ann. - Manuf. Technol. **65**, 405–408 (2016)
9. Tonelli, F., Evans, S., Taticchi, P.: Industrial sustainability: challenges, perspectives, actions. Int. J. Bus. Innov. Res. **7**(2), 143–163 (2013)
10. Vieira, G.E., Herrmann, J.W., Lin, E.: Rescheduling manufacturing systems: a framework of strategies, policies, and methods. J. Sched. **6**(1), 39–62 (2003)
11. Wang, S., Liu, M., Chu, F., Chu, C.: Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. J. Cleaner Prod. **137**, 1205–1215 (2016)