Management Science Letters 8 (2018) 1117–1132

Contents lists available at GrowingScience

Management Science Letters

homepage: www.GrowingScience.com/msl

An adaptive genetic algorithm for a dynamic single-machine scheduling problem

Jose-Fernando Jimenez[a*], Eliana Gonzalez-Neira[a] and Gabriel Zambrano-Rey[a]

[a]*Industrial Engineering Department, Pontificia Universidad Javeriana, Bogota, Colombia*

C H R O N I C L E

Article history:
Received: July 9, 2018
Received in revised format: July 9, 2018
Accepted: August 20, 2018
Available online:
August 21, 2018

Keywords:
Adaptive Genetic algorithm
Dynamic Scheduling
Manufacturing control
Predictive-Reactive
Optimality
Reactivity

A B S T R A C T

Nowadays, industries cope with a wide range of situations and/or perturbations that endanger the manufacturing productivity. Traditionally, manufacturing control systems are responsible for managing the manufacturing scheduling and execution, as these have the capability of maintaining the production operations regardless of a given perturbation. Still, the challenge of these systems is to achieve an optimal performance after the perturbations occur. For this reason, manufacturing control systems must incorporate a mechanism with intelligent capabilities to look for optimal performance and operation reactivity regardless of any scenario. This paper proposes a generic control strategy for a manufacturing control system for piloting the execution of a dynamic scheduling problem, considering a new job arrival as the manufacturing perturbation. The study explores a predictive-reactive approach that couples a genetic algorithm for the predictive scheduling and an adaptive genetic algorithm for reactivity control aiming to minimize the weighted tardiness in a dynamic manufacturing scenario. The results obtained from this proposal verify that the effectiveness was improved by using adaptive metaheuristic in a dynamic scheduling problem, considering absorbing the degradation caused by the perturbation.

1. Introduction

A manufacturing system (MS) is the location where converge physical, human and economic resources to produce certain goods to fulfil the market demands. The operation of an MS is a complex task that involves the planning, scheduling and controlling of materials and information flow within a manufacturing environment. Specifically, the complexity of these systems increases because several operations are waiting for being scheduled, jobs are competing for using the available resources, the materials encounter bottlenecks that slow down the production flow, among many others (Halevi & Cunha, 2007; Banos et al., 2011). In fact, the complexity increases further during the execution as the production is often obstructed due to the occurrence of perturbations, such as resource breakdown, tool breakage and unexpected urgent orders. For this reason, an MS is managed with a control system that monitors and conducts the manufacturing tasks within the operation execution, called manufacturing control system (MCS). An MCS is an automatic system which deals with the progress, resource, maintenance, quality, monitoring, tracking and directing requirements of a manufacturing system (Naedele et al., 2015). MCS is also the central nervous system of the

* Corresponding author.
E-mail address: j-jimenez@javeriana.edu.co (J.-F. Jimenez)

doi: 10.5267/j.msl.2018.8.011

manufacturing location and it is responsible for coordinating the completion of the requested activities using the available resources (Baker, 1998). Consequently, industries expect that MCS can achieve optimal performances under complex demands while it can react to unexpected perturbations (Jimenez et al., 2015).

For the scheduling, there are mainly two strategies to control the production of the dynamic environment of an MS (Vlk & Barták, 2015): a) Centralized-based architectures and b) distributed-based architectures. Generally speaking, MCSs are composed by one or several entities hosting a decisional-making technique (Trentesaux, 2009). On one hand, centralized architectures are generally implemented to feature predictive decision-making techniques, such as mathematical programming models or metaheuristics. These aim to achieve optimal performance in the manufacturing scheduling through the optimization of the global performance indicators. On the other hand, the distributed architectures are normally implemented to feature reactive decision-making techniques, such as artificial intelligence or heuristic algorithms. These aim to feature reactivity over perturbed scenarios through the responsiveness of entities actions. However, these approaches do not address entirely the requirements of optimal performance and operation reactivity. While the centralized/predictive approach fulfils the optimal performance, the distributed/reactive approach fulfils the reactiveness requirement. For this reason, MCS must feature a mechanism to look for both optimal performance and operation reactivity regardless of the manufacturing scenario.

Recent developments in MCS have heightened the need for pursuing a balance between the optimal performance and reactivity requirements. These approaches, called predictive-reactive approaches, are designed to benefit from the advantages of predictive and reactive approaches without the associated drawbacks (Thomas et al., 2009). In general, these have two phases: Offline, which refers to the decision-making techniques carried out before the beginning of the execution, and the Online, which refers to the decision-making techniques carried out during the production execution. While the offline phase generates a manufacturing schedule with a predictive technique, the online phase repairs the schedule with a reactive technique aiming to react to perturbations. Some examples of this approach can be found in (Baños et al., 2011; Chu et al., 2014; Rey et al., 2014)

Recently, researchers have studied the use of adaptive metaheuristics for controlling the execution of manufacturing operations. Some examples of these types of approaches are in swarm optimization approaches (Barbosa et al., 2015; Holvoet et al., 2009; Novas et al., 2013) and evolutionary algorithms (Borangiu et al., 2014; Jimenez et al., 2017). This paper focuses on the use of evolutionary algorithms for the dynamic scheduling of manufacturing operations, specifically an adaptive genetic algorithm (AGA). The adaptive metaheuristic, one of the most studied mechanisms, is a population-based technique that simulates the evolution process in order to reach an optimal or near-optimal solution regarding a fitness indicator. The AGA, which inherited the characteristics of genetic algorithms (GA), has strong optimization ability, fast calculation, simple principles and operation, robust generality implicit parallelism and global search space ability (Pan et al., 2011). For this reason, we believe that the featured characteristics of AGA can be used in order to control the manufacturing execution framed under a predictive-reactive approach.

This paper proposes a generic control strategy for a manufacturing control system for piloting the execution of a dynamic scheduling problem. We explored a predictive-reactive approach that couples a GA and an AGA for the predictive scheduling and reactivity control respectively. Initially, in order to have solution reference with traditional techniques, we develop a parameter-tuned traditional GA to solve a static single machine weighted tardiness problem. Then, we construct three versions of the AGA considering inner adaptability parameter control as deterministic parameter control, adaptive parameter control and self-adaptive parameter control. Finally, we compared the performance of the proposed AGAs with the traditional GA in order to gain some insights about adaptable optimization tools in a single machine-manufacturing environment. For the validation of this proposal, this research

takes the form of a case study of a dynamic single-machine scheduling problem. It considers two main issues. First, it considers new job arrivals during execution as the manufacturing perturbation. Second, it aims to minimize the weighted tardiness within the dynamic manufacturing scenario. In specific, the originality of this validation is that it explores the response ability of the general control strategy proposed through the occurrence of successive perturbations or new job arrivals.

This paper is organized as follows. Initially, section 2 contextualizes manufacturing control systems, scheduling and rescheduling and adaptive genetic algorithms. Then, section 3 presents the design of the traditional genetic algorithm and the proposed adaptive genetic algorithm. Afterwards, section 4 executes the simulation for both metaheuristics (GA and AGA) and validates the results for verifying the efficiency of this proposal. Finally, section 5 presents the conclusions, considered further research and important remarks within this study.

2. Theoretical Context

2.1. Manufacturing control systems (MCS)

Traditionally, Industries follow a sequential decision-making process in order to fulfil market requirements. First, a material requirement and capacity planning in the strategic decision level is designed. Second, a master and detailed scheduling in the tactical decision level is planned. Third, a shop-floor control and rescheduling in the operational level is implemented. This practice is made mainly because it presents good and near-optimal performance within manufacturing productivity (Jimenez et al., 2013). Unfortunately, due to the difficulty of visualizing the entire manufacturing system, the production planning is based on some uncertain information, such as estimated capabilities, aggregated forecast and virtual inventories. At the same time, decisions-making in the planning and scheduling stage, consider unlikely static environments instead of real dynamic environments. In response to mentioned difficulties, MCS emerges to resolve planning, scheduling and execution activities. The term has evolved because of the necessities in a manufacturing system. At the beginning of 1990's, as a complete master-slave structure, the MCS was only a black box entity, which was received from external system and generated device-specific instructions necessary to enact the individual manufacturing task (Wysk & Smith, 1995). Later on, it has been recognized the necessity of dealing with unexpected events in the manufacturing execution. However, the events are handled as unplanned activities and are introduced to this black box as expedited rework (Norton, 1996). Subsequently, the manufacturing strategy of handling disruptions was to reduce the production complexity (Elmaraghy et al., 2012). In this case, the control decision-making is adjusted to immediate conditions, making resources assignments rather than jobs allocations (Halevi & Cunha, 2007). The MCS functionality is constructed with a centralized architecture. It consists of an SCADA system of which the main functions are supervision, data acquisition and estimation of its key performance indicators (Trentesaux & Prabhu, 2011). However, manufacturing systems should be capable of responding to environmental disruptions with flexible, expansible, agile and re-configurative architectures (Leitão, 2009). Additionally, it must also consider satisfactory decision-making in terms of productivity optimization (Thomas et al., 2012) and articulate simultaneously the optimal performance and reactivity operations.

2.2. Scheduling and Rescheduling

Scheduling is a manufacturing decision-making process that deals with the allocation of resources to jobs over the given time periods (Pinedo, 2016). Ideally, in the manufacturing system, it makes an optimal allocation subject to certain objectives and capabilities. However, the layout of the manufacturing shop-floor, the dynamism in the manufacturing processes and the circumstantial attempted objectives, define the complexity of scheduling problem. Among them, handling the dynamic environment and maintaining the scheduling plan to the sequence execution is a complex

task. Even it reaches the optimal, most robust, predictable and preventive sequencing order, uncertain events disturb the manufacturing process. Indeed, manufacturing activities is frequently subject to several random occurrences and perturbations (Madureira et al., 2000). In that case, it is clear that the dynamic characteristics of any scheduling problem are difficult to handle.

The disruption of the sequencing order is the most critical activity in the manufacturing process. This activity, called Rescheduling, is the process of updating an existing production schedule in response of disruptions or other changes (Vieira et al., 2003). It intends to correct the sequence order taking into account the actual deviation due to environmental perturbations (Nie et al., 2013) So, this technique works as a reactive catalyst on the MCS. Above all, the rescheduling is closely related to achieving system robustness. The rescheduling is classified in three categories (Ouelhadj & Petrovic, 2009). First, the *completely reactive approach,* which is characterized with a lack of a previous sequential arrangement and decisions are made locally in real-time (usually adopt heuristic rules). Second, the *predictive-reactive approach,* which creates a robust initial sequence foreseeing possible disruptions and react with single schedule adjustments (usually relies into the initial scheduling optimal decision). And third, the robust *proactive approach,* which focuses on building predictive schedules in order to minimize the schedule deviation (giving forecasted slacks to the sequencing problem).

In practice, different techniques for rescheduling process have been proposed. Initially, the first mechanism that handles rescheduling is the heuristics rules as dispatching rules or greedy heuristics (Li et al., 2000; Tan & Aufenanger, 2011). Particularly, this method has the advantage of being operatively simple and it delivers solutions in short time periods. However, it does not guarantee to maintain the robustness in terms of optimality and efficiency. Even though, it considers a research perspective due to the straightforward application in industrial factories. Then, the second rescheduling is based on the metaheuristics techniques such as genetic algorithm, particle swarm optimization or simulated annealing (Chryssolouris & Subramaniam, 2001; Jimenez et al., 2016). The metaheuristics is a search process that explores the space of possibilities in order to reach an optimal solution in a decision-making process. In the rescheduling context, it should minimize the variation of the performance indicator given by environmental disruptions. However, due to its iterative characteristics, the optimal feature vanishes with the increasing of problem size and, consequently, it lacks of finding a near-optimal solution in reasonable time (Bierwirth & Mattfeld, 1999). Nevertheless, its performance improves the heuristic approach and contributes a better decision-making support to the manufacturing control. Finally, the third rescheduling approach is handled with distributed artificial intelligence systems as multi- agents, petri-nets, or neural networks (Barbosa et al., 2011; Holvoet et al., 2009). Briefly, the MCS allocates a virtual entity to each physical object to represent the object in a virtual world. These entities can generate local decisions for adaptable and local performances and, at the same time, interact with other entities to fulfill global objectives. At this point, the challenge of these techniques relies on the ability of performing satisfactorily optimized subject to the additional restriction imposed by the manufacturing execution status and the disruption event. The rescheduling process satisfies the functioning rules for sequencing major or minor modifications and, ideally, it has the ability to sustain a reasonable performance under perturbations.

2.3. Adaptive Genetic Algorithm

A Metaheuristic is a high-level procedure to search a near optimal solution to an optimization problem (Bianchi et al., 2008). Depending on the problem, it can be used a population-based (Evolutionary Algorithm, Ant Colony, Particle Swarm Optimization, among others) or a trajectory-based (Tabu Search, Simulated Annealing, among others) procedure. These algorithms have proved to be very effective for solving various combinatorial problems, but they are usually limited to particular domain problems (Cowling & Chakhlevitch, 2003). Even thought, a metaheuristic stands over a

range of problems due to the simplicity of codifying and customizing the structure to any problem.

The genetic algorithm is described as a gradient-free robust optimization technique that mimic the evolutionary process of nature (Palit & Popovic, 2006). After tuning the metaheuristic (population size, crossover operator, crossover probability, mutation operator, probability of mutation, selection operator and replacement operator, among others), the genetic algorithm aims to optimize a specific problem representing its solutions through a restricted size population. Each feasible solution will be considered as an individual of the population that collectively and iteratively will evolve to new feasible solutions. Generally, at each iteration, the population passes through a selection of individuals which are crossed afterward by recombination and mutation processes (Genetic Operations). Finally, considering the initial population and the new set of new solutions, a replacement strategy will determine which individuals will stay in the population. After many iterations of this process, the population of individuals will evolve to better solutions in terms of fitness value. In general terms, the genetic algorithm provides a robust search and are less expensive in computational time compared to other optimization solutions.

The genetic algorithm has limitations. The main limitation is that it is a parameter-dependent method. Depending on the parameters, it might be that the solutions will converge before as expected, called premature convergence. Therefore, the algorithm leads the population to be trapped in local optima specifically by the lack of individual diversity. Consequently, the individuals will misplace individual competition and it will lead to an inefficient random search (Palit & Popovic, 2006). In response to genetic algorithm limitations, it is comprehensible to consider the adaptation of parameters within the genetic algorithm that improves the optimization search capabilities. In fact, adaptable genetic algorithms (AGA) can adjust crossover and mutation probability to alleviate Genetic algorithm problems (Pan et al., 2011). AGA has a monitoring and actuator module that, according to current key performance measures, will modify the parameters to improve optimization search. Therefore, adaptive versions of genetic algorithms are particularly needed because, in the process of the evolutionary search, the algorithm should converge to the global optimum with a high speed of convergence.

Depending on the specific need, the AGA might be classified based on the location and the adaptability. The AGA might be adaptable in the parameter settings, genetic operators, genetic selection, representation and fitness function (Herrera & Lozano, 1996). AGA can also be activated according to some adaptive rules. Thus, the adaptive strategy is settled according to the following approaches (Smit & Eiben, 2009). First, the deterministic parameter control, where the change of AGA functioning parameter takes place in a fixed or pre-determinate way where no processes monitors information) and works with the number of iterations. For instance, determining a fixed mutation probability for the initial stage of the search and then change it for finals stages, is an example of this approach. Afterwards, the adaptive parameter control is a feedback mechanism that depends on the monitoring of system performance. However, this approach is activated externally and it might not be continuous. In fact, the update mechanism serves as a directional strategy to modify the search with certain key information but not in a regular basis. Finally, the self-adaptive parameter control, which is activated with the gathered processed information are general rules of adaptation that works as boundaries of parameters, functions and operators dynamism. In this approach the evolving mechanism, reaches a certain level of freedom since the AGA functioning parameters will be adjusted depending on the evolving development. Still, the approach should have an implicit control, which is given to adjustments boundaries, and eventual deterministic parameter controls.

The AGA must monitor the performance of the operators and the diversity of population, among others in order to use the gathered information to adjust the functioning (Smith, 1998). The information gathered contributes the creation of a Key Performance Indicator (KPI). This KPI will be

the processed sensorial information that activates the actuator AGA controlling system. For example, a well-known KPI is related with the essential balance in the search process between the exploitation and exploration techniques (Palit & Popovic, 2006). Hence, a relation indicator between these techniques will serve as a diversity measure. The challenge is to construct the indicator and set the lower and upper bounds under dynamic environments. Likewise, other KPI is the percentage involvement that indicates the proportion of generation contributing to the next one (Baker, 1985). Again, as a second effort to avoid fall of diversity, this indicator is involved with the adaptation of the population size to sustain steady adaptability during iterations.

3. A reschedule mechanism for the dynamic single machine problem

The single machine scheduling problem (SMSP) is a manufacturing problem that deals with the sequencing of jobs on a single processor or machine (Madureira et al., 2000). This problem is the simplest of all manufacturing environments, but the study of this environment is relevant as it is considered an special case of all others more complex manufacturing environments (Pinedo, 2016). In fact, it gives some insights in the understanding, resolution, managing and modeling more complex multi-processor problems (Madureira et al., 2000). Then, it can be used in each sub-problem resulted from the decomposition of more complicated scheduling environments. In addition, this environment could be embedded within a large control system in order to assist the problem resolution.

Accordingly, the problem to solve in this study is limited to the single machine environment known as the $1 \mid r_i \mid \sum w_i T_i$, where there is one processor, a set of jobs j arrives to the system at the release date r_i and the objective $\sum w_i T_i$ is to minimize the total weighted tardiness. For researchers, this problem is considered in the static SMSP because the sequencing is executed only once and all jobs releasing dates are known before processing starts. However, due to the dynamic nature of real scheduling problems, the consideration of the arrivals to a sequencing order where they arrive with uncertainly is part of the dynamic SMSP (Madureira et al., 2000).

At this moment, it has been some advances to solve the dynamic SMSP. In static scheduling problems, there have been many approaches to solve the sequencing of this problem. These approaches range from very sophisticated techniques for optimal accuracy to fairly unsophisticated heuristic designing primarily for practice implementation (Pinedo, 2016). Still, neither search-based sophisticated techniques nor enumerative-based heuristic methods is completely appropriate because once the scheduling is settled, the sequence needs to be modified due to unexpected disruptions (Nie et al., 2010). For this reason, this paper proposes an approach that tackles the dynamic scheduling for the single machine problem. The dynamic scheduling problem is a scheduling problem where usually inevitable unpredictable real-time events may cause a change in the scheduled plans (Ouelhadj & Petrovic, 2009). These types of problems have been tackled with different strategies (Aytug et al., 2005): Completely reactive strategy, robust strategy and Predictive-Reactive. The completely reactive strategy constructs a new schedule based on local information, pre-settled assumptions and general normative during perturbations. The robust strategy creates a schedule that minimizes the effect of disruption on the main schedule objective. Finally, the predictive-reactive strategy, in a two steps procedure, sets a desire behavior at the beginning and then modifies it during the execution in response to unexpected disruptions. In general, there is not any rule that performs consistently better than all other rules under several manufacturing layout configurations (Nie et al., 2011). However, many researchers made some efforts to exploit several methods on artificial intelligence to learn to select rules dynamically according to the current state of the system.

3.1. Dynamic single machine problem

This paper focuses on dynamic single machine scheduling problem where the dynamic SMSP with job release dates is described as follows. The case study is composed with one machine and a set of n jobs to process i. The jobs are released over time and are processed once on the machine without preemption.

Each job has a processing time p_i, release time r_i, due date d_i, and priority weight w_i, which denotes the relative importance of job i, $i= 1,...,n$. For the dynamic scheduling problem, there are two sets of job: a subset where the release date r_i is known and another subset where the release date is unknown in advance. It is also assumed that the machine cannot process more than one job, simultaneously. The objective is to determine a sequence of jobs on the machine in order to minimize the weighted tardiness of the jobs' scheduled.

3.2. Proposed generic control strategy

This study proposes a framework of a manufacturing control system (MCS) for the dynamic SMSP. The MCS approach proposes piloting the execution of a dynamic scheduling problem, for handling the schedule and reschedule a new job arrival as the manufacturing perturbation. We explore a predictive-reactive approach that couples a genetic algorithm for the predictive scheduling and an adaptive genetic algorithm for reactivity control. The objective of this MCS is to minimize the weighted tardiness in a dynamic manufacturing scenario.

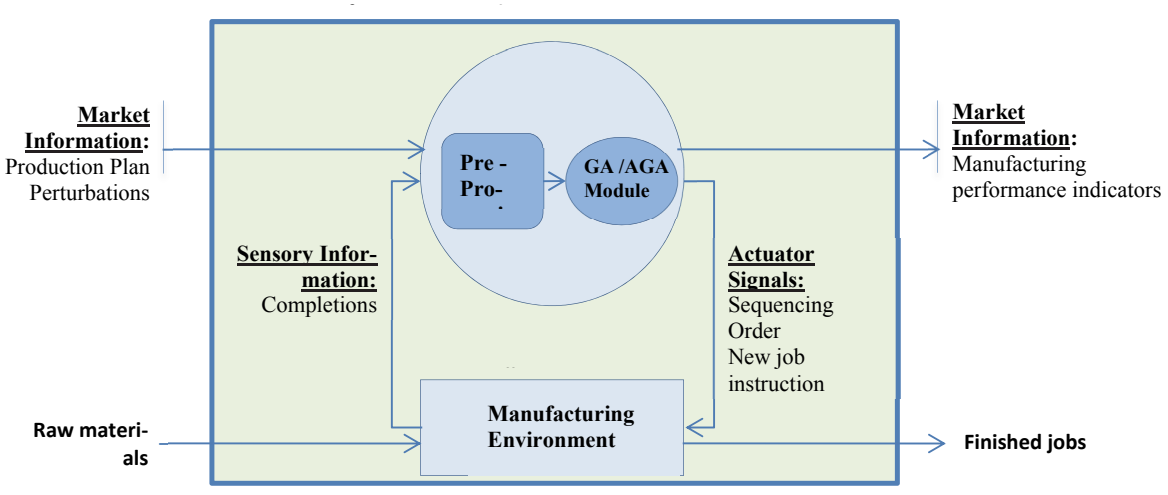


Fig. 1. Architecture of the proposed manufacturing control system

The architecture of the proposed MCS is based on the block-diagram of manufacturing control system (Baker, 1998). The proposed architecture is illustrated in Fig. 1. This architecture is composed by two entities: a) the control decision-making entity and b) the controlled execution entity. The relationship of these two entities is a master-slave relationship as the first commands the execution of the sequence and the latter follows these instructions. The functioning of these entities is as follows. The *control decision-making entity* receives the production order for the manufacturing system. In this case study, it also receives the instruction of processing new jobs (new arrivals) and it will be considered as the execution perturbation. The results of this entity is to provide the sequencing of the jobs for both the beginning (i.e. predictive scheduling) and during (i.e. Reactive control) the manufacturing execution. While a genetic algorithm is used for the predictive scheduling and an adaptive genetic algorithm is used for the reactive control. These two methods are further explained in section 3.3. On the other side, the controlled execution entity is in charge of executing the instructions commanded by the control decision-making entity. This entity simulates the operations in the manufacturing system and provides the information during the execution.

3.3. Genetic Algorithm and Adaptive Genetic algorithm

The MCS proposed a predictive-reactive approach for the scheduling and control of the production

order. In specific, this proposal uses the benefits of the evolutionary algorithms for featuring optimal performance and reactivity features in the manufacturing system. This proposal relies on the optimality of a genetic algorithm to set an initial schedule coupled with an adaptive genetic algorithm for supporting the corrective actions after a perturbation. The perturbation considered in this paper is based on new job arrival. In this sense, a traditional genetic algorithm (i.e. GA) is proposed by considering a parameter tuning procedure and three versions of an adaptive genetic algorithm (i.e. AGA1, AGA2 and AGA3) are proposed for changing the characteristics of the reactivity features. Therefore, these ranges from none to certain grade of adaptability.

The GA and AGAs proposed responds to the classification made by Smith and Eiben (2009). The first mechanism is the traditional ***GA***. It was created as a reference solution of the single machine environment, where it was statistically tuned with a genetic algorithm in the population size and mutation rates. The idea of tuning parameters was to consider the best available solution from the static version of the optimal mechanism. The second mechanism is the ***AGA1*** with deterministic parameter control. It represents a mechanism that alters the parameters with some deterministic rule. In this case, the heuristic rule is to maintain the same parameters value during the whole run but they are initialized regarding the current execution situation. The third mechanism is the ***AGA2*** with adaptive parameter control. It represents the mechanism that sets from the beginning of the run a change regarding execution indicators. In this paper, the mechanism changes a parameter per iteration regarding the time elapsed in the run. Finally, the fourth mechanism ***AGA3*** with self-adaptive parameter control. This last mechanism considers the parameters change depending on the evolutionary search completing an evolution of parameter together with the population. Briefly, the designs of previous adaptive mechanisms are efforts to improve search capabilities in the evolutionary algorithms.

Encoding. In this study, for the four tested mechanisms the solutions are codified in order that each gene represents a job index. Thus, each gene position in the chromosome represents the job sequence in the scheduling solution, and all together represents the solution to be scheduled in each individual.

Initial Population. An initial population is created randomly from the jobs to schedule. At this stage, we did not overlook for generating a good initial solution for assuring the possibility that each mechanism starts form a similar reference and diverse the initial population.

Operators. Four mechanism we have used traditional genetic algorithm operators in the permutation encoding. First, for the selection operator uses the roulette wheel. Then, the recombination and mutation operator are used to uniform permutation and single swap operators, respectively, Finally, for replacement we have used the steady state operator.

Fitness function. Considering the single machine environment, the weighted tardiness fitness function is used as a performance indicator. Basically, it is chosen because there is a penalization when job does not meet due dates and at the same time there is no benefit or penalty for early completions.

Population Size. The number of individuals is a parameter that changes in the constructed mechanisms. In the GA, there is a population of 100 individuals each time the MCS activates the optimal mechanism. In contrast, the population for the AGA mechanisms can be calculated with the following equation:

$$n = (200\,\Delta t + \beta)/\,10, \tag{1}$$

where n is the number of individuals to consider for the population, Δt is the time remaining to complete the execution of current job, and β is the number of jobs pending in the production order.

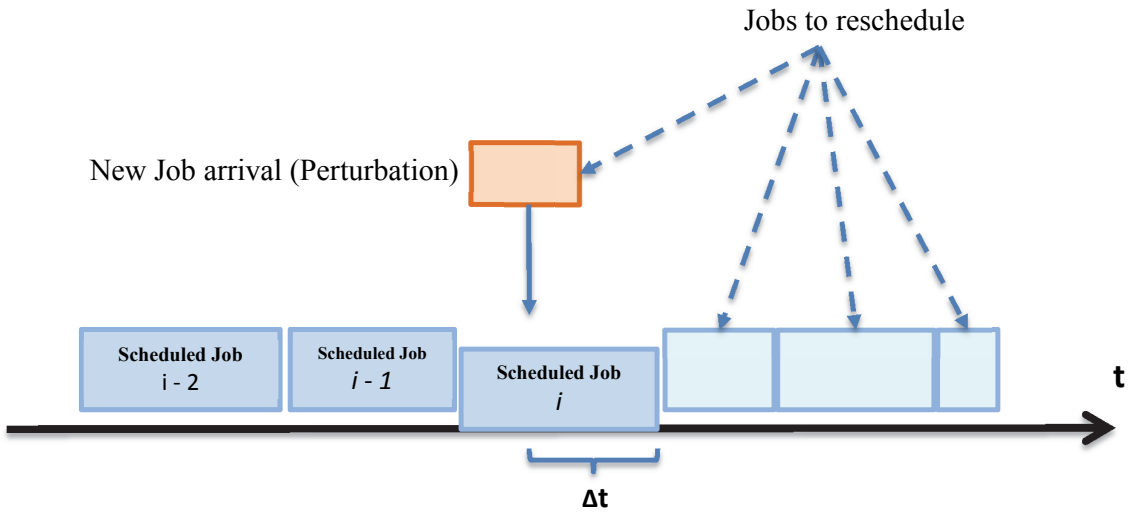


Fig. 2. New job arrival triggers an identification of jobs and parameters for rescheduling

Mutation Probability (P_m). The mutation probability is also a parameter that differs depending on the optimal mechanism. While for the ***GA*** and the ***AGA1***, the mutation rate is 0.01, for the other two the rate includes an inner dynamism. For the ***AGA2*** , the P_m changes linearly starting from 0.01 until it is finished when Δt time has passed with 0.1 (Pan et al., 2011). Certainly, this parameter is recalculated each time the mechanism is reinitiated due to a new perturbation. Finally, for the **AGA3**, the execution indicator chosen is the distance between the average and the best solution in its phenotypic fitness function. Its value can be calculated with the following expression (Srinivas & Patnaik, 1994):

$$p_m = \begin{cases} k_1 \times (f - f_{min}) / (f_{avg} - f_{min}) & f \geq f_{avg} \\ k_2 & f < f_{avg} \end{cases} \quad (2)$$

where f_{min} is the best minimum fitness value, f_{avg} is the average fitness value, f is the fitness value of individual to be mutated and *k1, k2* are the constants values for the mutation.

4. Experiments and results

The validation of the MCS approach was performed by generating an instance of 210 jobs in total (See appendix A): 200 jobs with release date is from the beginning of the execution and 10 jobs are considered for new arrivals for the execution. These new job arrivals are considered for the perturbation in the dynamic scheduling environment and it is necessary to include accordingly to the sequencing order. For the two types of jobs, the attributes were created as follows. The processing times of each job were distributed with a discrete probability following a uniform function U [1,20]. The priority weights of each job were also created with a uniform function distribution U [1,10].

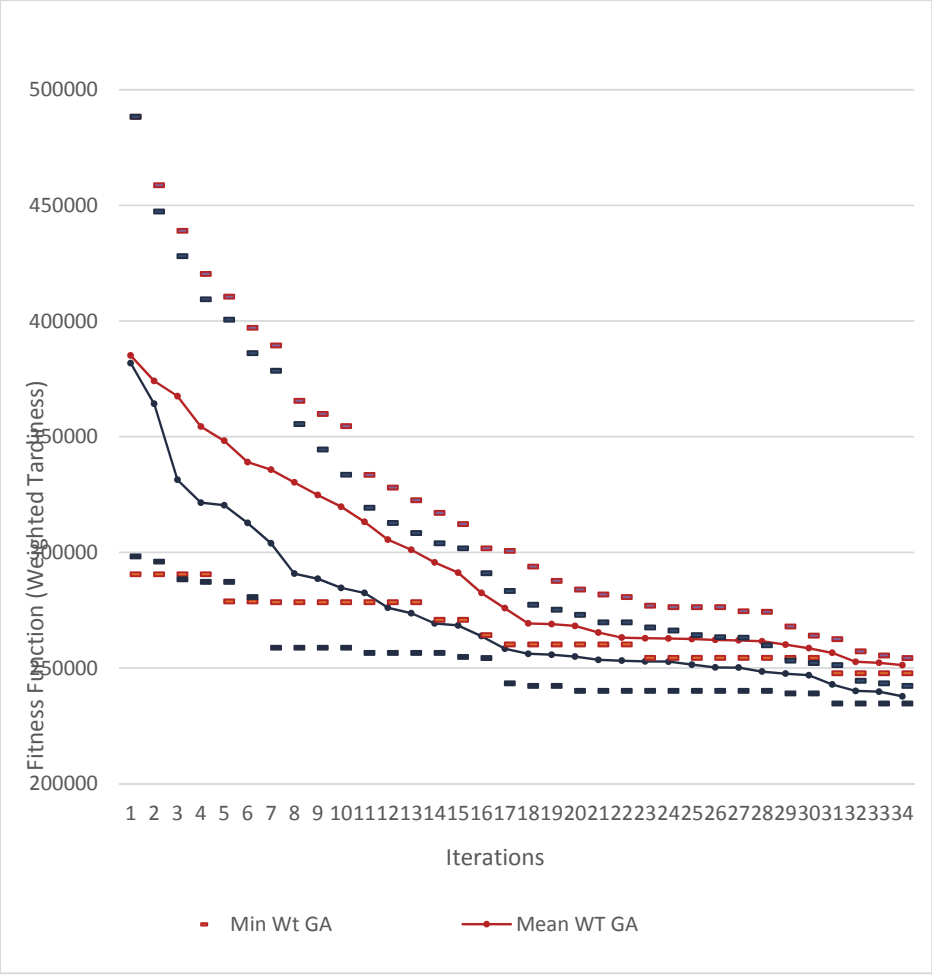
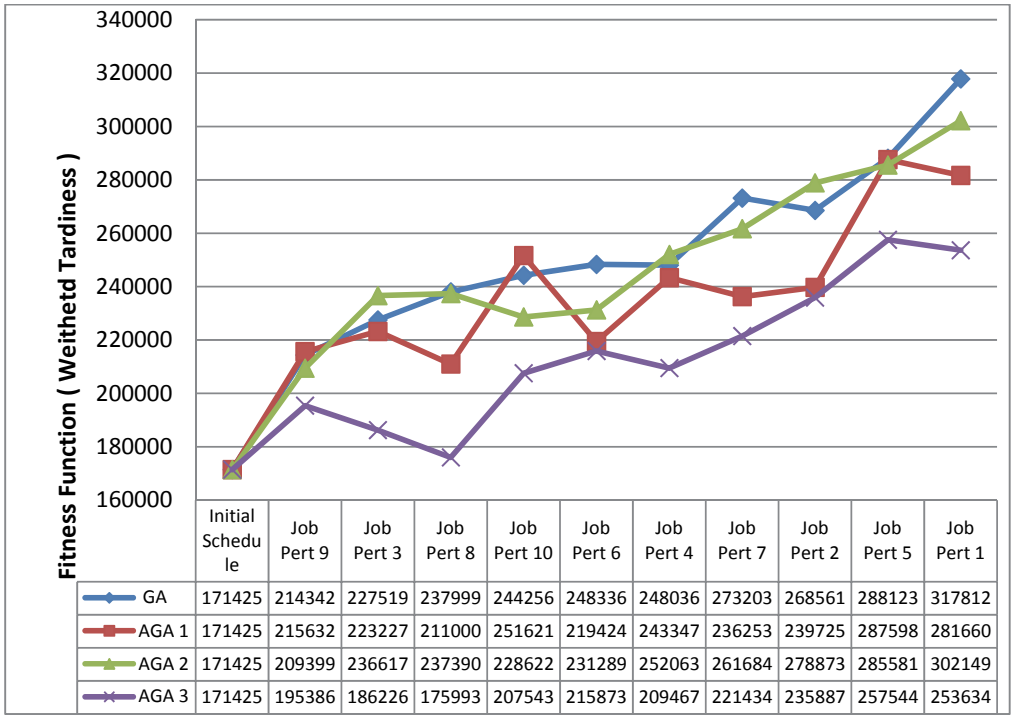


Fig. 1. Evolution of reschedule by traditional GA and AGA 2

The release time for the first 200 jobs were fixed in zero, which means that there were available for processing from the beginning; and the release time for new arrival jobs were generated with a uniform function distribution U (1, *1/2*$\sum p_i$), where $\sum p_i$ is the aggregation of the 200 jobs processing times. Finally, the due date for each job was generated with a uniform function distribution U (*1/3* $\sum p_i$, *2/3* $\sum p_i$). For replication purposes of this research, index No 1 presents the instance created for this paper. First of all, to illustrate the functioning, we plot in Fig. 3 the inclusion of the sixth job (Release date of 663) to a schedule made previously with the traditional GA (the algorithm comparable). At this point, it was being processed the job 176, it still remains 4 seconds to its completion (Δt) and 154 jobs to be rescheduled (Including the new job arrival). At this point, the traditional GA and the AGA 2 were tested by considering the best, average and worst fitness values per iteration. As it can be seen from the Fig. 3, the average fitness is better in AGA while the algorithm was running. Indeed, it is remarkable that the convergence for the initial stage is faster in AGA 2 than in traditional GA. In fact, this outcome becomes a desire behavior due to the possibility of having optimal performance in limited time (Or even less). Additionally, besides of the average, the difference between the best performances of both mechanisms exhibits a superiority of AGA. Equally, the individuals between that deliver other solutions for the control problem, and they will be eventually preferred by the control decision-making entity, i.e. Multi-Objective Criteria. Now, in order to check the efficiency of the optimal mechanism, the MCS is configured with the schedule and reschedule technique (i.e. GA and AGA) and is tested with the generated instance to compare long time run results. Fig. 4 illustrates the best fitness at each perturbation with the estimated weighted tardiness. As can be seen from the data comparison of the algorithm, the efficiency of AGA 3 algorithm is higher than the traditional GA and first two AGA versions. Foremost, when

setting an optimal mechanism into a manufacturing execution, the AGA mechanism supports constantly a superior condition.



	Initial Schedule	Job Pert 9	Job Pert 3	Job Pert 8	Job Pert 10	Job Pert 6	Job Pert 4	Job Pert 7	Job Pert 2	Job Pert 5	Job Pert 1
GA	171425	214342	227519	237999	244256	248336	248036	273203	268561	288123	317812
AGA 1	171425	215632	223227	211000	251621	219424	243347	236253	239725	287598	281660
AGA 2	171425	209399	236617	237390	228622	231289	252063	261684	278873	285581	302149
AGA 3	171425	195386	186226	175993	207543	215873	209467	221434	235887	257544	253634

Fig. 2. Comparison of optimal performance in the manufacturing execution

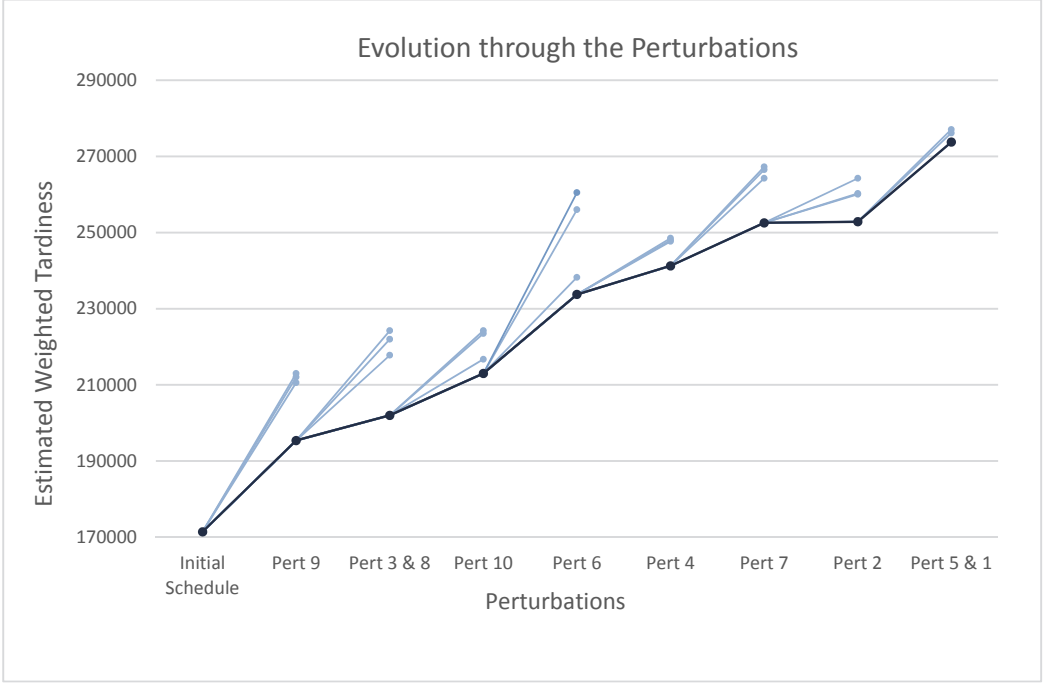


Fig. 3. Manufacturing execution monitoring degradation and choosing best AGA control strategy

However, to strengthen the conclusion to articulate an adaptive metaheuristic in shop-floor control systems, it is necessary to validate the mechanism at each perturbation. In Fig. 5, we have compared each perturbation with four the optimal mechanisms. However, in order to fairly compare the mechanisms, each time is chosen the best performance to simulate with the next mechanism. Note that the first sequence was scheduled by the traditional GA and we have observed that in two times more than one perturbation arrives at the same time of the executing job. That situation causes that the reschedule to be restricted with the last arrival remaining time.

Yet, the AGA stands over the traditional GA. Over the tested disruptions, there is a dominant superior performance in the adaptive algorithm. At certain point, depending on the complexity that the new job represents (in terms of time remaining, number of job to reschedule and the inner parameter of jobs to order), the optimal mechanism will control the increasing trend of a new job arrival. In that case, the performance and the convergence velocity will be crucial for achieving appropriate corrective actions. For this reason, it is recommendable the inclusion of an adaptive metaheuristic, which eventually will enhance optimality to the search procedure, to potentiate the performance of a control necessity.

Nevertheless, Fig. 5 demonstrates that the optimal mechanism might have certain conditions to accomplish their performance objectives. From the perturbations, it has been six where the best was the AGA 3. Undoubtedly, the good performance of this mechanism relies on checking how it is behaving during its run and sets its parameters to improve the results. But, on the other hand, it has been 2 versions of AGA 2 that are better in the perturbations. At a point, it is remarkable that the self-adaptive might not be superior because it has to balance the self-adaptation to avoid falling in premature convergences. In that case, it is not possible to choose a better AGA version in this case, but it is certain to include an adaptive genetic algorithm, instead of traditional genetic algorithms, to ameliorate the new schedule at perturbations.

Therefore, for validating these results, statistical test has been conducted, between the fitness value of the 8 perturbations with 10 replications between the traditional GA and the AGA 2. The objective of this statistical test is to check whether there is significant difference between its fitness values.

Table 1
Manufacturing execution monitoring degradation and choosing best AGA control strategy

ANOVA	Perturbations							
One way	Pert 9	Pert 3 & 8	Pert 10	Pert 6	Pert 4	Pert 7	Pert 2	Pert 5 & 1
F-Value	46,46	2,07	20,91	62,16	46,9	0,084	9,77	62,74
P - Value	2,21,E-06	1,67,E-01	2,36,E-04	3,01,E-07	2,08,E-06	7,75,E-01	5,84,E-03	2,82,E-07
Critical F(1,18)	4,41	4,41	4,41	4,41	4,41	4,41	4,41	4,41
Hypothesis	Rejected	Fail	Rejected	Rejected	Rejected	Fail	Rejected	Rejected

A one-way analysis of variance (ANOVA) showed that, with a confidence of 95%, the effect of noise was significant in the used algorithm with F (1,18) = 4.41, for 6 out of 8 tests. This means that in this case it is rejected that the sample the fitness values comes from the same population and, in fact, they have a significance difference. In the fail to reject cases, there is not any possibility of knowing with the sample values. The AGA 2 achieves significantly better results than the traditional GA.

5. Conclusions

In this paper, a manufacturing control system has been constructed, which could manage the sequence of jobs at the beginning and during the execution of the manufacturing operations. This approach explores the rapid convergence of adaptive genetic algorithm to achieve an optimal performance and reactivity in a predictive-reactive scheduling approach of a manufacturing control system. The experiments

conducted illustrate that the adaptive genetic algorithm has provided some satisfactory results by considering some limited execution time. At the same time, the findings suggest that the use of an adaptive mechanism achieves better performance results due to the convergence achieved and the customization that parametrize the adaptive metaheuristic. It was also appropriate to potentiate the ability of optimization techniques for the extreme conditions in order to enhance the optimal performance capability. Therefore, the results of this research support the idea that adaptive algorithms were appropriate to be extended to the control of manufacturing systems. These findings provide the following insights for future research: a) the coupling of static metaheuristics of the predictive scheduling and adaptive metaheuristics for the reactive control should be explored further to find the limitations derived from this approach. b) this approach could feature adaptation features in different locations of the architecture of the control system (i.e. adaptive genetic operators, adaptive genetic operator selection, adaptive representation and adaptive fitness function). c) it is possible to explore some mechanisms to find a trade-off between the optimal performance and reactivity given by the control system, such as multi-objective metaheuristics or artificial intelligence methods.

Acknowledgements

This research was conducted under the post-graduate programme for lecturers [Programa de Formacion al Profesor] by the University: “Pontificia Universidad Javeriana”

References

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, *161*(1), 86-110.

Baker, A. D. (1998). A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: dispatching, scheduling, and pull. *Journal of Manufacturing Systems, 17*(4), 297-320.

Baker, J. E. (1985, July). Adaptive selection methods for genetic algorithms. *In Proceedings of an International Conference on Genetic Algorithms and their applications* (pp. 101-111).

Banos, R., Manzano-Agugliaro, F., Montoya, F. G., Gil, C., Alcayde, A., & Gómez, J. (2011). Optimization methods applied to renewable and sustainable energy: A review. *Renewable and Sustainable Energy Reviews, 15*(4), 1753-1766.

Barbosa, J., Leitão, P., Adam, E., & Trentesaux, D. (2015). Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in Industry, 66*, 99-111.

Barbosa, J., Leitão, P., Trentesaux, D., & Adam, E. (2011, November). Enhancing ADACOR with biology insights towards reconfigurable manufacturing systems. *In IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society (pp. 2746-2751)*. IEEE.

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing, 8*(2), 239-287.

Bierwirth, C., & Mattfeld, D. C. (1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary computation, 7*(1), 1-17.

Borangiu, T., Răileanu, S., Berger, T., & Trentesaux, D. (2015). Switching mode control strategy in manufacturing execution systems. *International Journal of Production Research, 53*(7), 1950-1963.

Chryssolouris, G., & Subramaniam, V. (2001). Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing, 12*(3), 281-293.

Chu, Y., You, F., & Wassick, J. M. (2014). Hybrid method integrating agent-based modeling and

heuristic tree search for scheduling of complex batch processes. *Computers & Chemical Engineering, 60*, 277-296.

Cowling, P., & Chakhlevitch, K. (2003, December). Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In Evolutionary Computation, 2003. CEC'03. The 2003 Congress on (Vol. 2, pp. 1214-1221). IEEE.

ElMaraghy, W., ElMaraghy, H., Tomiyama, T., & Monostori, L. (2012). Complexity in engineering design and manufacturing. *CIRP Annals-Manufacturing Technology, 61*(2), 793-814.

Halevi, G., & Cunha, P. F. (2007). *Self organization shop floor control*. In Digital Enterprise Technology (pp. 107-114). Springer, Boston, MA.

Herrera, F., & Lozano, M. (1996). Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing, 8*, 95-125.

Holvoet, T., Weyns, D., & Valckenaers, P. (2009, September). Patterns of delegate mas. In Self-Adaptive and Self-Organizing Systems, 2009. SASO'09. *Third IEEE International Conference on (pp. 1-9). IEEE*.

Jimenez, J. F., Bekrar, A., Trentesaux, D., Montoya-Torres, J. R., & Leitão, P. (2013, October). State of the art and future trends of optimality and adaptability articulated mechanisms for manufacturing control systems. *In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on (pp. 1265-1270)*. IEEE.

Jimenez, J. F., Bekrar, A., Trentesaux, D., Zambrano-Rey, G., & Leitão, P. (2015). Governance mechanism in control architectures for flexible manufacturing systems.

Jimenez, J. F., Bekrar, A., Zambrano-Rey, G., Trentesaux, D., & Leitão, P. (2017). Pollux: a dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research, 55*(15), 4229-4247.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence, 22*(7), 979-991.

Li, H., Li, Z., Li, L. X., & Hu, B. (2000). A production rescheduling expert simulation system. *European Journal of Operational Research, 124*(2), 283-293.

Madureira, A., Ramos, C., & do Carmo Silva, S. (2000). A genetic algorithm for the dynamic single machine scheduling problem. *In Advances in Networked Enterprises (pp. 315-324). Springer*, Boston, MA.

Naedele, M., Chen, H. M., Kazman, R., Cai, Y., Xiao, L., & Silva, C. V. (2015). Manufacturing execution systems: A vision for managing software development. *Journal of Systems and Software, 101*, 59-68.

Nie, L., Gao, L., Li, P., & Shao, X. (2013). Reactive scheduling in a job shop where jobs arrive over time. *Computers & Industrial Engineering, 66*(2), 389-405.

Nie, L., Gao, L., Li, P., & Wang, X. (2011, June). Multi-Objective optimization for dynamic single-machine scheduling. *In International Conference in Swarm Intelligence* (pp. 1-9). Springer, Berlin, Heidelberg.

Nie, L., Shao, X., Gao, L., & Li, W. (2010). Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *The International Journal of Advanced Manufacturing Technology, 50*(5-8), 729-747.

Norton, N. (1996). Shop-floor control. *Manufacturing Engineer, Volume 78*, 175 – 178.

Novas, J. M., Van Belle, J., Saint Germain, B., & Valckenaers, P. (2013). A collaborative framework between a scheduling system and a holonic manufacturing execution system. *In Service orientation in holonic and multi agent manufacturing and robotics (pp. 3-17). Springer, Berlin, Heidelberg*.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling, 12*(4), 417.

Palit, A. K., & Popovic, D. (2006). *Computational intelligence in time series forecasting: theory and engineering applications*. Springer Science & Business Media.

Pan, Y., Zhang, W. X., Gao, T. Y., Ma, Q. Y., & Xue, D. J. (2011, June). An adaptive Genetic Algorithm for the Flexible Job-shop Scheduling Problem. *In Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on (Vol. 4, pp. 405-409). IEEE*.

Pinedo, M. L. (2016). *Scheduling: theory, algorithms, and systems*. Springer.

Pinedo, M. L. (2016). *What Lies Ahead?*. In Scheduling (pp. 545-554). Springer, Cham.

Smit, S. K., & Eiben, A. E. (2009, May). Comparing parameter tuning methods for evolutionary algorithms. *In Evolutionary Computation, 2009. CEC'09. IEEE Congress on (pp. 399-406). IEEE.*

Smith, J. (1998). Self adaptation in evolutionary algorithms (Doctoral dissertation, University of the West of England).

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer, 27*(6), 17-26.

Tan, Y., & Aufenanger, M. (2011, July). A real-time rescheduling heuristic using decentralized knowledge-based decisions for flexible flow shops with unrelated parallel machines. *In Industrial Informatics (INDIN), 2011 9th IEEE International Conference on (pp. 431-436). IEEE*.

Thomas, A., El Haouzi, H., Klein, T., Belmokhtar, S., & Herrera, C. (2009). Architecture de systèmes contrôlés par la produit pour un environnement de juste à temps. *Journal Européen des Systèmes Automatisés (JESA), 43*(4-5), 513-535.

Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence, 22*(7), 971-978.

Trentesaux, D., & Prabhu, V. V. (2011). Introduction to Shop-Floor Control. *Wiley Encyclopedia of Operations Research and Management Science*, 1-9.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling, 6*(1), 39-62.

Vlk, M., & Bartak, R. (2015). Replanning in Predictive-reactive Scheduling. Association for the Advancement of Artificial Intelligence.

Wysk, R. A., & Smith, J. S. (1995). A formal functional characterization of shop floor control. *Computers and Industrial Engineering, 28*(3), 631-644.

Rey, G. Z., Bonte, T., Prabhu, V., & Trentesaux, D. (2014). Reducing myopic behavior in FMS control: A semi-heterarchical simulation–optimization approach. *Simulation Modelling Practice and Theory, 46*, 53-75.

Appendix A

Initial Jobs																										Perturbation jobs				
Job	Pj	Wj	dj	rj	Job	Pj	Wj	dj	rj	Job	Pj	Wj	dj	rj	Job	Pj	Wj	dj	rj	Job	Pj	Wj	dj	rj	Job	Pj	Wj	dj	rj
Job 1	8	6	1068	0	Job 41	20	8	1091	0	Job 81	4	7	1206	0	Job 121	8	6	1132	0	Job 161	7	3	1305	0	Pert 1	19	4	905	872
Job 2	5	9	1276	0	Job 42	5	3	1490	0	Job 82	14	8	1094	0	Job 122	9	9	1052	0	Job 162	11	4	1287	0	Pert 2	16	4	1318	818
Job 3	13	2	1460	0	Job 43	20	3	947	0	Job 83	5	9	901	0	Job 123	8	9	985	0	Job 163	14	8	1466	0	Pert 3	7	1	1256	267
Job 4	13	9	973	0	Job 44	18	2	1137	0	Job 84	10	6	1321	0	Job 124	13	6	980	0	Job 164	14	3	1377	0	Pert 4	10	5	1232	663
Job 5	8	4	1424	0	Job 45	16	8	1089	0	Job 85	13	2	1441	0	Job 125	7	6	1463	0	Job 165	20	3	1089	0	Pert 5	5	6	1442	867
Job 6	12	7	1368	0	Job 46	18	10	1085	0	Job 86	8	8	1227	0	Job 126	6	1	1005	0	Job 166	5	4	1237	0	Pert 6	18	4	1204	589
Job 7	9	5	1125	0	Job 47	19	4	1180	0	Job 87	18	8	1215	0	Job 127	15	9	1241	0	Job 167	20	6	1245	0	Pert 7	13	5	1237	746
Job 8	12	1	1471	0	Job 48	6	6	980	0	Job 88	10	10	1014	0	Job 128	19	4	1494	0	Job 168	11	1	968	0	Pert 8	18	2	971	272
Job 9	18	3	910	0	Job 49	20	3	1254	0	Job 89	12	7	1097	0	Job 129	13	6	1053	0	Job 169	9	8	1333	0	Pert 9	4	7	1477	8
Job 10	6	1	1363	0	Job 50	4	4	1303	0	Job 90	9	1	1316	0	Job 130	11	2	1155	0	Job 170	12	8	1139	0	Pert 10	5	1	1158	319
Job 11	7	2	1347	0	Job 51	14	2	1260	0	Job 91	17	8	1054	0	Job 131	7	2	1204	0	Job 171	18	5	1149	0					
Job 12	8	4	1462	0	Job 52	10	5	1413	0	Job 92	5	1	1328	0	Job 132	7	2	1478	0	Job 172	9	3	1034	0					
Job 13	19	2	1473	0	Job 53	17	2	1298	0	Job 93	9	9	905	0	Job 133	16	7	1270	0	Job 173	5	3	905	0					
Job 14	9	2	923	0	Job 54	6	3	1230	0	Job 94	10	1	905	0	Job 134	6	9	1224	0	Job 174	17	5	1430	0					
Job 15	5	2	1196	0	Job 55	16	3	1483	0	Job 95	6	1	1083	0	Job 135	11	8	1307	0	Job 175	5	10	1371	0					
Job 16	15	1	1356	0	Job 56	20	4	1216	0	Job 96	7	10	1220	0	Job 136	4	3	1215	0	Job 176	13	10	1200	0					
Job 17	20	2	1102	0	Job 57	8	5	1047	0	Job 97	12	2	905	0	Job 137	11	5	1132	0	Job 177	20	2	939	0					
Job 18	12	5	1406	0	Job 58	18	5	1126	0	Job 98	19	1	936	0	Job 138	19	1	1077	0	Job 178	16	1	907	0					
Job 19	19	10	1397	0	Job 59	16	1	1041	0	Job 99	7	9	1134	0	Job 139	18	10	1094	0	Job 179	4	4	1167	0					
Job 20	7	5	1031	0	Job 60	14	4	1413	0	Job 100	6	8	1348	0	Job 140	17	2	1396	0	Job 180	10	2	1376	0					
Job 21	19	9	1366	0	Job 61	5	8	1057	0	Job 101	6	9	1337	0	Job 141	16	5	1062	0	Job 181	13	3	1475	0					
Job 22	16	2	911	0	Job 62	12	4	1327	0	Job 102	14	6	1090	0	Job 142	14	1	1266	0	Job 182	16	3	1083	0					
Job 23	20	9	1064	0	Job 63	4	7	1243	0	Job 103	15	5	1038	0	Job 143	8	3	1363	0	Job 183	8	8	1079	0					
Job 24	18	9	1171	0	Job 64	5	2	1339	0	Job 104	19	9	1337	0	Job 144	8	7	1046	0	Job 184	18	4	1287	0					
Job 25	8	3	1480	0	Job 65	17	3	1279	0	Job 105	9	9	1087	0	Job 145	15	1	1071	0	Job 185	8	6	1179	0					
Job 26	20	2	975	0	Job 66	13	5	1498	0	Job 106	10	5	1136	0	Job 146	18	8	1308	0	Job 186	11	7	1074	0					
Job 27	8	4	1094	0	Job 67	4	9	1095	0	Job 107	16	7	1130	0	Job 147	7	4	1277	0	Job 187	5	9	926	0					
Job 28	7	3	1397	0	Job 68	14	7	1002	0	Job 108	13	6	1368	0	Job 148	9	6	1029	0	Job 188	4	4	1125	0					
Job 29	15	1	1443	0	Job 69	12	8	1203	0	Job 109	12	6	1195	0	Job 149	4	10	911	0	Job 189	9	8	1488	0					
Job 30	15	3	1306	0	Job 70	13	9	1044	0	Job 110	16	8	919	0	Job 150	14	3	1034	0	Job 190	6	1	1266	0					
Job 31	18	5	1149	0	Job 71	11	8	1044	0	Job 111	17	7	1467	0	Job 151	7	3	1437	0	Job 191	14	7	1060	0					
Job 32	18	8	1041	0	Job 72	11	7	1087	0	Job 112	8	7	1051	0	Job 152	4	5	1347	0	Job 192	5	6	1108	0					
Job 33	5	8	1232	0	Job 73	20	1	951	0	Job 113	7	1	1445	0	Job 153	19	5	1076	0	Job 193	18	6	1403	0					
Job 34	4	2	1434	0	Job 74	12	4	1457	0	Job 114	14	2	1405	0	Job 154	6	6	1068	0	Job 194	13	6	938	0					
Job 35	16	4	1457	0	Job 75	14	8	1228	0	Job 115	12	8	1442	0	Job 155	5	3	1031	0	Job 195	10	3	1025	0					
Job 36	5	8	1022	0	Job 76	19	10	1167	0	Job 116	6	8	943	0	Job 156	17	7	1115	0	Job 196	18	7	1021	0					
Job 37	15	2	1465	0	Job 77	10	2	1057	0	Job 117	12	4	929	0	Job 157	19	9	1077	0	Job 197	6	1	1384	0					
Job 38	19	2	1309	0	Job 78	4	4	941	0	Job 118	4	6	1055	0	Job 158	7	4	1044	0	Job 198	4	5	1024	0					
Job 39	8	2	1141	0	Job 79	16	9	1340	0	Job 119	19	1	1067	0	Job 159	11	5	1297	0	Job 199	15	10	1226	0					
Job 40	8	2	964	0	Job 80	6	7	1488	0	Job 120	10	5	1205	0	Job 160	4	1	1412	0	Job 200	18	7	948	0					

p_j = *Procesing time of job j*
w_j = *Weight priority of job j*
d_j = *Due date of job j*
r_j = *Release time of job j*