A novel scheduling framework: integrating genetic algorithms and discrete event simulation

Luca Fumagalli, Elisa Negri, Edoardo Sottoriva, Adalberto Polenghi, Marco Macchi

luca1.fumagalli@polimi.it – corresponding author -,
elisa.negri@polimi.it, edoardo.sottoriva@polimi.it,
adalberto.polenghi@polimi.it, marco.macchi@polimi.it

Department of Management, Economics and Industrial Engineering, Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milan, Italy

Abstract: Most of the research works on methods and techniques for solving the Job-Shop Scheduling Problem (JSSP) propose theoretically powerful optimization algorithms, that indeed are practically difficult to apply in real industrial scenarios due to the complexity of these production systems. This paper aims at filling the gap between research and industrial worlds by creating a framework of general applicability for solving the JSSP. Through a literature analysis, the constituent elements of the framework have been identified: an optimization method that can solve NP-hard JSSP problem in a reasonable time, i.e. the genetic algorithm (GA), and a tool that allows to precisely model the production system and evaluate the goodness of the schedules, i.e. a simulation model. A case study of a Company that bases its business in the manufacturing of aerospace components proved the applicability of the proposed framework.

Keywords: scheduling framework; job shop; JSSP; genetic algorithm; discrete event simulation.

Reference to this paper should be made as follows: Author. (xxxx) 'Title', *Int. J. xxxxxxxxxx xxxxxxxxxx*,

Biographical notes:

This paper is a revised and expanded version of a paper entitled ***Simulation-supported framework for job shop scheduling with genetic algorithm*** *presented at XXII Summer School of Industrial Mechanical Plants "Francesco Turco", Palermo (Italy), 13th-15th September 2017.*

1 Introduction

In the current manufacturing environment, the importance of operational scheduling is becoming strategic, since a careful production programming and a detailed scheduling activity are necessary conditions for answering customers' requests for order variety, short lead time and fast deliveries (Liao et al. 2013) and for best exploiting the flexibility of production systems enabled by new technologies (Joseph & Sridharan 2011). The overall result is that a careful scheduling allows to increase the competitiveness of the company onto the market.
Engineering researchers applied efforts to solve the Job-Shop Scheduling Problem (JSSP), by studying a wide range of different techniques, in order to support industrial practitioners. Several scientific articles have been proposed since the eighties to face scheduling problems with methods, techniques and algorithms of different natures and research interest on this topic is still very high until nowadays (Cavalieri et al. 2000; Cavalieri et al. 2007). The challenging issue that fuels this research interest is that the complexity of the JSSP has been demonstrated to be NP-complete (Garey & Johnson 1975). Furthermore, digitalization is a new driving force that can help the deployment of new scheduling techniques. In fact, it is possible to create innovative solutions, by exploiting the Industry 4.0 vision of having all the machines of the shop floor connected together and linked to the corporate information systems thanks to the Internet of Things (IoT) paradigm (Negri et al. 2017; Abramovici et al. 2015; Lee et al. 2014; Ashton 2009). This is enhanced by the possibility to analyse Big Data, to perform real-time computing and so to create accurate and reliable virtual representation of the physical production system (Digital Twin), that are enabled by the Industry 4.0 technologies.

This work proposes, in the first instance, a methodological framework of general applicability to perform operative scheduling on any production system, as a further elaboration of the research work described in (Fumagalli et al. 2017),. It then presents also the results of the actual application of the developed technique in a real industrial scenario.

The optimization strategy chosen for the proposed methodology is the genetic algorithm (GA), already widely used in literature for complex problems (Kurdi 2015). In parallel to the presentation of the framework for scheduling, this work pays attention on the literature about GAs, providing a classification of the several GA variants. One of the characteristics of the GAs is to be independent on the type of production system, in fact (Liao et

al. 2013)proposes an example of use of GAs to solve scheduling problems for different kinds of production system configurations. With respect to traditional GAs, the introduced novelty lies in the fact that the behaviour of the production system under a specific schedule is no longer assessed with an analytical formula, but with a simulation, run on the Digital Twin model. In order to validate the applicability of the proposed framework for scheduling, it has been implemented considering the real industrial case of a company that desired to optimize the operative schedule on its job shop floor.

The paper is organized as follows: Section 2 proposes an analysis of the literature of the GAs and production scheduling approaches; Section 3 defines the research objectives of the work; Section 4 describes the developed JSSP; Section 5 shows the validation process of the proposed framework at a company; in Section 6 conclusions are proposed.

2 Scheduling and Genetic algorithms literature analysis

The scheduling is the short period phase of production planning and control, whose goal is to output operative orders receiving production orders as input. The operative orders describe the quantities of products to be produced within a given time period. Thus, scheduling is a decision-making process that aims at matching the required jobs against the resources of the company (Brandolese et al. 1991). Traditionally, companies applied a practical approach in order to solve the scheduling problem, i.e. to recognize the complexity of the problem and the imperfections of the reachable solutions and then to provide contingent solutions (bigger production capacity, outsourcing, higher prices for faster deliveries, etc.) to infeasible schedules. This approach seems no longer valid in the current industrial world, because companies look at improving profitability and competitiveness and this can be done through an optimized resource allocation (Salido et al. 2016).

The research community has defined different classes of scheduling problems depending on their nature (Framinan et al. 2014):

- open vs closed: depending on the presence of relationships with other decision-making processes;
- deterministic vs stochastic: depending on the nature of the input data of the problem;

- static vs dynamic: depending on the availability of relevant data at the beginning of the decision-making process.

Once the problem has been delineated it is possible to define objectives, variables and constraints through a modelling process and then to apply an optimization algorithm to find a solution (Nocedal & Wright 2006).

A classification of the different solving techniques for the optimization of the scheduling problem can be provided; many contributions have been explored (Xhafa & Abraham 2008; Mallikarjuna 2014; Jourdan et al. 2009; Ruiz & Maroto 2005; Brandolese et al. 1991; Framinan et al. 2014) and a common classification structure that matches with all of them has been developed (Figure 1).

The first distinction is done between exact and approximate methods: the former ones ensure the best (optimal) solution with respect to the fixed objectives and are accompanied with a formal demonstration (Brandolese et al. 1991), the latter ones accept a lower probability to find a global optimal solution, in order to get near-optimal solutions in reasonable and practical computational times (Xhafa & Abraham 2008), making them particularly suitable for the complex problems, such as the NP-complete JSSP problem, for which the necessary computational effort exponentially increases with the problem size the (Garey & Johnson 1975).

The second distinction is done within the approximate methods, between heuristic and meta-heuristic approaches.

- Heuristics are empirical procedures, i.e. sets of instructions derived from the practice, following which, it is possible to find a solution to a specific problem. They are typically suited for low complexity systems, while for high complexity problems the use of these approaches is very limited (Liao et al. 2013).
- The need to have more flexible and robust approaches (Liao et al. 2013), leads to the use of meta-heuristic methods. (Blum & Roli 2003) reports some formal definitions of meta-heuristics, but they can be briefly described as search space exploration methods aimed towards a satisfying (near-optimal) solution. Their characteristic of being non problem specific (Blum & Roli 2003) insures the flexibility, instead the robustness is provided by the “intelligent” direction of exploration (Xhafa 2008), driven by logical moves and aware of their effect to the optimization process, in this way facilitating the escape from local optima.

(Çaliş & Bulkan 2015) have provided a summarizing chart of the applications of the several meta-heuristic optimization methods to the

scheduling problem. In this chart, GAs are indicated as the most used to solve scheduling problems. Following the analysis of (Çaliş & Bulkan 2015), this paper focuses on GAs, as the most fit algorithms to solve the faced problem.

From the beginning of the theorization of GAs (Holland 1975), they resulted a good solution to the production systems management optimization, due to their feature of being able to solve large-scale combinatorial optimisation problem (Coley 1999), such as the JSSP problem. In the middle of the eighties, it was also discovered that GAs were promising approaches to problems such as operative scheduling. As an example (Palmer et al. 1988) tried one of the first implemented algorithms based on evolutionary computation, and demonstrated its efficiency improvement with respect to an algorithm-based approach. A more comprehensive overview of the state-of-the-art was presented by (Rodammer & White 1988), which proposed a detailed analysis of seven different approaches to scheduling activities. The conclusion of this contribution was that control theory, simulation and artificial intelligence (AI) are well suited to perform scheduling activities taking into account common features of this kind of problems: boundary stage, batch size, setup costs, process routings, random events, disturbances, performance criteria and multiple objectives.

GAs spreading was facilitated by the advent of computers, that were able to take on the high computational effort of GAs. Nevertheless, at that time the running of a GA was still a very high time-consuming activity due to the relative low computational ability of calculators in the first yeas of the nineties. This has initiated a research stream focused on the reduction of individuals in the population, that compose the overall search space, in order to improve the performances of the algorithms: an example is found in the article by (Buckles et al. 1990) that used the schema theorem in order to explore this path. A knowledge-based approach integrated with GA is demonstrated to give good results in reducing the search space of the problem, managing the idle times (Ying & Bin 1996).

In parallel with the adaptation of GAs to the technology, another branch of research on GAs was focused on creating valid individuals at every iteration of the algorithm itself, in order to avoid the generation of individuals that are highly fitted with respect to the problem objective, but physically infeasible, for example not respecting a constraint. This was reached by (Falkenauer et al. 1991) with the development of peculiar genetic operators of crossover and mutation. On the other hand, a good

example of this integration of other methods such as dispatching rules into the GA to solve the JSSP problem was provided by (Kumar & Srinivasan 1996), that were able to reduce the makespan of 30% with respect to the former production system. (Cavalieri et al. 1999) enhanced the applicability of GA for scheduling problems, implementing a specific GA for the scheduling optimization in a flexible job shop environment, which is the most complex problem among the JSSP problems (Chen et al. 1999), proposing a solution of GA with double encoding. Even (Zhiming & Chunwei 2000) underlined the importance to join two different approaches together in order to face flexible JSSP problems. Meanwhile (Ponnambalam et al. 2001) introduced a Multi Objective Genetic Algorithm (MOGA) that was built to avoid stalling into local optima, by changing weights of the fitness function at every iteration in a random fashion. In (Chryssolouris & Subramaniam 2001) the optimization strategy based on a GA is able to take into account three main aspects of a JSSP problem: random dynamic events, multiple scheduling criteria and multiple job routes, simulating the behaviour of a real production system. For the tendency to approach the real behaviour of manufacturing systems, (Kacem 2003) realized a GA for scheduling that considered technical-operational performance objectives, such as: minimum makespan, optimum workload of the critical machine or balanced total workload for all the machines, instead of costs or other economical indexes of more strategical nature. (Gonçalves et al. 2005) hybridized their GA with a local search procedure in order to enhance the algorithm effectiveness, but they touched another essential point for scheduling algorithms, i.e. they studied a codification based on random keys that allows to create only feasible solutions in a population and the objective function is devoted to evaluate the performance of a schedule. Also (Xing et al. 2006) found a way to avoid GA to get stuck in local optima by changing the GA steps according to the value of the fitness function of the individuals that are continuously updated; another solution to the premature convergence has been demonstrated by (Xing et al. 2007) whom act on crossover and mutation probability.

The generation of the initial population is also a critical aspect during the development of a new GA: a high performing starting population, in terms of fitness function, means a lower number of iterations to be done. Some heuristics should be used to optimize the generation of the population, as Tabu Search (Vilcot et al. 2006).

(Pezzella et al. 2008) demonstrated that by means of a highly customized GA framework it is possible to outperform many of the already existing scheduling algorithms.

The articles reviewed within this Section are thus representative of the main improvements in GA for the scheduling problem that have been carried out during the last decades. Also, it has to be underlined that Evolutionary Algorithms, to which GA belongs to, are continuously used for optimization even nowadays: it underlines once more the impact of these type of algorithms in research and industry. Just to mention a variant of GA, GP (Genetic Programming) is used for the dynamic scheduling of job shop system configuration (Park et al., 2018).

As it is possible to notice, GAs should be enhanced by modifying operators considering specific properties of the problem or by hybridizing them with other methods, in order to work effectively. The latter option has been the trend of the last years for the researcher community, according to a thorough literature review done by the authors, considering the time span from 2012 to 2016. The aim of this analysis was to provide a complete classification of the various GA applied to the JSSP problem and is reported in Appendix A. In particular, Figure 2 represents the percentages of application to the JSSP problem of the different GAs classified by type and shows that the most applied ones are hybrid genetic algorithms (HGA).

3 Research objectives

From all applications of GAs to scheduling found in literature, it arises that is often difficult to faithfully reproduce the functioning of production systems, taking into account physical, productive and management constraints typical of the manufacturing industries. In fact, the complexity of the systems usually does not match with the hypotheses that must be performed in order to run the optimization process, and when hypotheses are laid down, the found solution does not find immediate applications in the actual system. Literature proposes to solve these problems by enriching the GA with heuristic algorithms to solve specific functions (Zhou et al. 2001) or with the support of simulation models to carefully take in account the dynamics of the production environment (Chryssolouris & Subramaniam 2001).

The objective of the present research contribution is to elaborate a scheduling framework that leverages both on a GA and on a support tool: the former to actively search for an optimal solution and the latter to best adapt to the real industrial environment.

4 Proposed scheduling framework

This work proposes a framework for the scheduling optimization composed of two different elements joint together (Figure 3). The first and core component is the GA; it is the brain of the optimization process, since it creates populations of individuals (i.e. possible schedules of the production orders), reads the fitness value of each of them and operates iteratively in order to find the path to the optimum. The second component is the support tool: a simulation model that represents the specific production system to be scheduled. This is the operative arm of the optimization process: it simulates the schedules created by the GA, outputting their performances in the given production system, that will be used by the GA for evaluation of the alternatives and for the next iterations.

4.1 Main steps of the proposed framework

The proposed framework for the scheduling activity is composed of two main objects, the GA and the simulation model and its key feature is the interconnection between them, i.e. the continuous updating of schedules from the GA to the simulation model and the updating of performances from the simulation model to the GA. The optimization process is performed by the GA. In other words, the GA performs all the steps useful to identify the best-fit solution, meanwhile the aim of the simulation model is to simulate the timeline of the sequence of operations considering all the production constraints in the real production system. Moreover, once the time schedule has been defined, the model is able to output relevant performance indexes. A representation of the overall framework is reported in Figure 3. In the next pages, a description of each framework step is provided.

Input Data

Considering the purpose of the framework, that is to schedule operations within the production system, input data include the workpieces to be produced and the operations in their production cycle (i.e. jobs).

The set of jobs to be scheduled is not sufficient as input data, since it is necessary also to know the actual state of the machines (i.e. possible down states and initial configurations).

- If the machine of the real production system is in down-state, the actual production capacity is lower than the nominal one and the model has to take this into account.

- The state of the machines must describe the initial configuration of the machines, such as identifying the equipment mounted on them, or the parameters defining its initial functioning regime.

Finally, to sum up, input data to the framework are:

- Workpieces to be produced;
- List of jobs to produce the workpieces;
- Available resources in the production system;
- Actual state of the resources.

Population initialization

The GA evaluates the fitness of a defined number of points of the search space at every iteration, until the best-fit is found. The set of points of the search space that are evaluated at each iteration is a population of individuals. Each individual of the GA population corresponds to a so-called chromosome, i.e. a structure containing all its genes, where a gene represents a certain characteristic of the individual itself. The whole set of genes of a chromosome is called genotype and they can be externally observable through the phenotype. Therefore, an individual is externally represented as a possible schedule (phenotype), but it is also represented within the GA structure, with the codified representation of a possible schedule (genotype). A coding activity is necessary to elaborate the genotypes: in this case, the genotype has been represented thanks to a double array code. It means that, in order to have all the needed information to define a schedule, two arrays of the same length are coupled (Figure 4).

Thus, the digits of the first array find a direct link with the corresponding digits of the second one:

- The first array represents the sequence of jobs (every number univocally represents a job) in the order they enter the production system,
- The second array explicates the resources associated to the jobs in the corresponding position in the first array.

At the first iteration of the GA, permuting the digits of both the arrays a defined number of times, a population of individuals will be created. The permutations of the double array allow not only to change the loading sequence of the production system, but also to change the machines where the single jobs are processed, allowing a deeper search space exploration.

Individual export

Once a population has been generated by the GA, the simulation model takes an individual at a time as input.

This action seems to be automatic, but actually it is the de-coding process, which aims to translate the information contained in the two arrays into a language understandable by the simulation model. It is done through a piece of programming code representing the interface between the two components of the overall framework, the GA and the simulation model.

Application of simulation model for performance evaluation

The simulation model takes the sequences of jobs that have to be processed as input and, running the simulation, shows how the system behaves, recording its performances as output. The objective of the optimization process is indeed the optimization of the output of the simulation model, i.e. the performances associated to the individuals.

Individuals assessment

At the end of the simulation runs of all individuals of each iteration, the assessment of the performances of the individuals occurs. As previously said, to each individual are associated one or more performance indexes by the simulation model and in this step, those performance indexes are used to compute a fitness value, univocally associated to each individual. When the interesting production performances are more than one, one possible strategy to create the fitness function is to merge them into a single parameter, e.g. with a weighted mean. The fitness function expresses the optimization objectives of the problem at hand and cannot be therefore generalized: each application will have its own fitness function expression that describes the parameters according to which the individuals of the population are ranked.

Optimal solution search

Once a fitness value-based ranking of individuals has been created for the i-th iteration of the GA, the algorithm recognizes if a satisfactory optimal or sub-optimal solution has been reached. In this case, the algorithm stops showing which is the best schedule and all the parameters (scheduled

times and performances) that make it the best-fit. If it is not the case, it executes another iteration.

In order to teach the algorithm to identify the best-fit solution, it is necessary to implement termination criteria, that may be:

- stall generation criterion: the algorithm stops after a predetermined number of generations where the fitness value has not improved; meaning that the algorithm has found an optimum, even if this is only local. The optimum will be the best solution among the generated ones;
- maximum number of iterations: sometimes the algorithm does not converge to an optimum in a reasonable amount of time; for this reason, it is necessary to force it to stop after a certain number of iterations (i.e. a predefined upper limit) and the best solution found until then is the output of the algorithm.

The settings of these criteria must trade off two main conflicting aspects, i.e. the search space exploration and the computational effort. On the one side, the better the search space is explored, the better solutions are found and the risk to fall in local optima is reduced; on the other side, the more explorations are performed, the higher time and effort to compute iterations are required (that would imply a difficult applicability of the framework in real industrial environments, since the scheduling would not be performed in a timely manner).

Operators application

When the algorithm recognizes that the current iteration does not include the best-fit solution, it applies operations onto the individuals of the current population in order to create new ones. These operations are defined through genetic operators, that basically select some individuals from the current population according to certain criteria and permute them, following predefined rules.

Since the aim of the GA is to find the path to the optimum value of the fitness function, genetic operators tend to perform their operations on the fittest individuals of the current population, with the hope that, hybridizing already good individuals, the generated ones will happen to be even better (Koza 1995). This implies that genetic operators are fitness-based operators.

The drawback of using fitness-based operators is that it is possible to fall in a local optimum, so it is always necessary to weaken the data transfer only from the fittest individuals, picking up also other individuals randomly chosen from the whole search space.

The genetic operators are applied to the population of a certain iteration, in the following sequence:

1. Selection: A defined number of individuals from the population are picked following a fitness-based criterion; the probability of an individual to be selected is proportional to its fitness value;
2. Crossover: The selected individuals undergo a crossover operation two by two and new individuals are created by mixing the features of the original ones;
3. Mutation: Eventually, based on a certain probability set a priori, some of the selected individuals are mutated, meaning that some of their features digits are modified randomly;
4. Replacement: The newly generated individuals replace the ones that were previously selected.

The formation of a new population thanks to the application of genetic operators is iterative, so they will be applied at every iteration until the best-fit solution is reached.

5 Validation

In order to prove the applicability of the conceptual ideal of the scheduling framework, the authors applied the scheduling framework to a real Company.

The Company is a manufacturing company that operates in the market of forged and laminated rolled rings; their production is based on several materials (such as carbon and alloyed steels, cobalt alloys, titanium, aluminium, nickel and copper). The production system is organized in a job shop configuration where there is the furnaces department and the plastic deformation department (the latter composed only of one mill). The production cycle continuously produces inter-departments flows of materials, because the workpieces must be iteratively first heated in the furnaces department and then worked on the mill. These two operations must be repeated several times. The furnaces are all of the same type and a workpiece can be processed by any of them in its production cycle.

The proposed scheduling framework has been applied to the described case with the maximization of the utilization of the mill set as the main optimization criteria. The following activities were performed: mapping of

the characteristics of the system (see Section 5.1); customization of the framework to comply with the found system's characteristics, comprising the adaptation of the GA and of the simulation model (see Section 5.2); and analysis of the found results (see Section 5.3).

In Section 5.4 the pseudocode developed in the MATLAB/SIMULINK environment is presented: together with Figure 3 this will enhance the understanding of how the GA and the Discrete Event Simulation (DES) has been integrated to build up a unique scheduling framework.

5.1 Mapping

The implementation of the framework for scheduling follows the preliminary activity of choosing the most suitable software; in particular, for this application the implementation has been performed using MATLAB & SIMULINK software thanks to the MATLAB optimization toolbox and to SIMULINK capability to model and simulate a production system. The choice of the most proper software tool has followed a structured approach based on the Analytic Hierarchy Process, where both researchers and company experts provided their insights and priorities in order to take into account the perspectives of all actors involved in the research study. The software evaluation considered the criteria shown in Table 1.

Software Selection	**Implementation**	**Input/Output**
		Efficiency
		Programming aspects
		Visual aspects
		General characteristics
		Technical characteristics
	Validation	**Experimentation**
		Testing
	User	**Output**
		Support
		Pedigree

Table 1. merged hierarchy for simulation software selection.

When we are dealing with real industrial environments, the mere description of the production system, considering only its production equipment and the products' production cycle is not enough to understand its actual functioning. This happens because, usually, there are a lot of practical rules applied by the production personnel, that do not arise without a deep data analysis. In particular, this includes a detailed mapping of technical and managerial constraints to be considered in the scheduling activity, that are applied in the real system.

The first step of the mapping activity was to collect data for a whole working week, concerning:

- which workpieces enter in each machine;
- time of the entries of each workpiece in each machine;
- time of the exits of each workpiece in each machine.

The obtained result has been the production schedule GANTT chart that was actually performed in the real production system along a working week. With this, it was possible to discuss with the company the reason of specific scheduling decisions, in order to understand the physical and managerial constraints that led to the actual production schedule.

5.2 Customization to the case study

The proper implementation of the framework on the considered industrial scenario must include a customization of the theoretical structure, in order to consider all the constraints arisen during the mapping. As reported in Table 2, some of the constraints have been implemented in the GA, while others are modelled in the simulation model

Table 2. Constraints and their implementation

Object	Constraint	Explanation	Implementation
Pieces	Batch homogeneity	The production batches must be composed of workpieces with homogeneous features.	Simulation Model
Furnace	Specification	Each part number have specific furnaces where it can be processed. Moreover, when a workpiece has been	Genetic Algorithm

Object	Constraint	Explanation	Implementation
		assigned to a furnace, it must complete the production cycle in that furnace.	
	Temperature	Every part number has to be processed at a specified temperature. The temperature change between two different part numbers implies a setup time.	Simulation Model
	Capacity	Furnaces have limited capacity and it depends on which part number has to be processed.	Simulation Model
	Service time and tolerance	Workpieces are associated to a service time and a tolerance time. After staying in the furnace for the service time, workpieces are allowed to stay in the furnace only up to an additional time equal to the tolerance time.	Simulation Model
Mill	Capacity	The mill can process only one piece at a time.	Simulation Model
	Cycle time	For each workpiece, there must be no waiting time between two subsequent operations of heating and rolling and vice-versa.	Genetic Algorithm

5.2.1 GA implementation

The implementation of the GA needs setting some parameters and decisions. All of them are summarized in Table 3.Table 3

Parameter	Value/Description
Population size	100
Encoding	Double encoding consisting of: - one array devoted to operations and - one array devoted to machines. The coupling of these arrays highlights where each operation must be performed.
Selection	Roulette wheel
Crossover	2-points
Mutation	2-points
Fitness function	It consists of different indexes corresponding to the performance of the system (mill utilization) and constraints satisfaction.
Stopping criteria	- Maximum number of iterations = 100 - maximum number of stall generations = 30

Table 3. GA parameters summary

The definition of the fitness function is maybe the most important issue to have a GA work properly, in fact both the path to the optimum and the eventual convergence of the algorithm, depend on the shape of the fitness function.

The fitness function of the industrial case is presented by the expression below.

Fitness = - score_feas - score_tol - score_timediff - score_millutil;

Where the single terms have the following meanings:

- *score_feas* takes into account operations precedences, i.e. all the operations performed on a workpiece must be in a progressive order. It is a Boolean value that indicates the feasibility of a sequence. In particular, if the operations order is respected, the sequence is feasible and the score will be 1, otherwise if the operations order is not respected, the sequence is infeasible and the associated score will be -1;

- *score_tol* indicates if the jobs do respect the tolerance of its service time when they are inside the furnace (ref. to Table 2, Service time and tolerance of furnace). In particular, it is the ratio between the number of jobs that respect the tolerances over the total number of jobs of the sequence. It can take values in the interval (0,1). Values close to 0 indicate that most of the jobs do not respect the tolerances, instead values close to 1 indicate that most of the jobs respect tolerances;
- *score_timediff* indicates the cycle time constraint (ref. to Table 2, Cycle time of mill), i.e. it evaluates whether the sequence of operations for a certain workpiece occurs one immediately after the other, without interruptions, as indicated by the production cycle. It can take values in the interval (0,1). Values close to 1 indicate that most of the jobs respect the constraint, otherwise the scores are closer to 0;
- *score_millutil* is the business performance to be optimized (i.e. the mill utilization). It is defined as the ratio between the occupation time of the mill and the total makespan. It can take values in the interval (0,1). Values close to 0 indicate that the mill is under saturated; values closer to 1 indicate that the mill is well saturated.

After a trials campaign to validate the functioning of the implemented framework, the settings of the termination criteria are: 100 iterations for the maximum number of iterations criterion, and 30 generations for the stall generation criterion, as these values represent a good trade-off among computational effort and search space exploration.

The overall framework has been implemented into the MATLAB software. It has been written in a higher-level script that, after a section of data elaboration, contains the GA function that calls the production system simulation model every time the fitness function has to be evaluated. In fact, the values of the fitness function performance indicators are computed downstream of the simulation run, related to the current individual to be evaluated.

The pseudocode used to develop the framework is presented in Section 5.4.

5.2.2 Simulation model implementation

The simulation model has been built into the SIMULINK application of MATLAB software, exploiting the SimEvents toolbox, that is the one suitable for the DES. Moreover, SIMULINK can be easily integrated with the MATLAB workspace.

Also in the simulation models some constraints have been implemented. Referring to Table 2, these were: service time and capacity of the mill, furnace capacity, batch homogeneity, furnace temperature (setup time), furnace service time and tolerance.

The functioning of the simulation model is schematized in Figure 5, in particular it illustrates the three phases of the simulation model running:

- The first, *sequence upload*, is a piece of programming code that picks one individual at a time from the whole population and loads all the useful data into the digital twin of the production system.
- The second, *production simulation,* is the actual simulation of the production system producing parts following the input sequence.
- The third, *performance evaluation*, is aimed at evaluating the goodness of the reached schedule, assessing the performance indexes that will constitute the fitness function.

The representation of the model in SIMULINK environment, is reported in Figure 6.

The layout of the simulation model resumes in its blocks the three-phases functioning scheme. The blocks in the left part of the model upload the sequence of jobs to be processed. They select an individual from the population created by the GA at the i-th iteration and take all the needed information from the database to perform the simulation. The actual production system is represented by the blocks named “FURNACES” and “MILL”. The workpiece meets iteratively furnaces and mill with production parameters defined according to its production cycle. All the other blocks are necessary for the performance evaluation phase; they record production times and use them to compute the performance indexes, that are used in the fitness function.

5.3 Results

The framework provides two different outputs, that are: (i) the optimal schedule of the input sequence of jobs, that indicates the starting and the ending times of the processing of every job, and (ii) a plot of the trend of

the fitness function with the iterations of the algorithm, an example of which is represented in (Figure 7).

Simulations have been performed with a PC with Intel® Core™ i7-5500 CPU @2.40GHz with 16 GB RAM and the average computational time has been about 8 hours. A further increase of the GA parameters, such as max number of iterations and population size, would have required longer times for simulation runs.

In Figure 7 it is possible to analyse the path followed by the algorithm in terms of fitness function until the optimum value. It presents the best value that seems constant along iterations, but actually, the termination criteria of stall generation did not stop the algorithm; this means that a certain improvement of the fitness function happened at every iteration, even if very small.

The second output, that is the schedule of the input sequence of jobs, is provided by MATLAB in a table format, in which every row corresponds to a certain job reporting also its starting and ending times of the processing of each step of the production cycle. This is sufficient to create a Gantt chart that represents the optimal schedule in a more user-friendly way.

5.4 MATLAB pseudocode

The goal is to provide a comprehension of the integration of GA and DES presenting the pseudocode developed within the MATLAB environment. Together with Figure 3, the pseudocode completes the understanding of how GA and DES could work together to improve the JSSP performance, either from a high-level perspective, that is the scheduling framework, either from a more practical perspective, that is the script developed to run the framework.

The next sections will describe the code in terms of: input, output and logic. All the next sections will reflect the various step of the framework developed in Section 4.1.

Population initialization

This part of the code is used to generate the initial population needed by the GA to start. The creation of the initial population is not mandatory in general GA coding except for the generation of particular individuals, for example composed by two arrays, as the developed framework needs.

Input: list of jobs that are needed to produce the set of workpieces
Output: set of permutations of the input sequence of jobs with the associated furnaces
Logic:

```
Population=cell(population_size,1);
for i=1:population_size
   Population{i,1}(:,1)=randperm(jobs);
      for j=1:number_of_jobs

Population{i,1}(j,2)=available_furnaces(randi(length(available_furnaces)));
      end
end
```

Selection function

This part of the code is used to create the selection function that will be roulette wheel-based. MATLAB has already implemented a set of default fitness function among those it is possible to choose the most fitted one for the problem under analysis.

Input: population of a certain iteration of the GA
Output: set of indexes of the population chosen to be submitted to the crossover and mutation function and to advance in the iterations
Logic: MATLAB default roulette wheel selection function

```
optimoptions('ga','SelectionFcn',@selectionroulette)
indexes=selectionroulette(population);
```

Crossover function

This part of the code is used to define the crossover function that will be a 2-points crossover. Since the population was customized for the industrial case under analysis, default options built in MATLAB are not available. Thus, the main effort and attention is put on the correct definition of the two points along the arrays.

Input: indexes of the selected elements of the population
Output: set of individuals with permutated elements with respect to the 2-points crossover function
Logic:

```
for i=1:(number_of_selected_individuals-1)
```

```
    parent_1=Population(indexes(i));
    parent_2=Population(indexes(i+1));
    points=sort(randi(number_of_jobs,2,1), 'ascending');
     for j=1:(points(2)-points(1))
        ord_2_1(j)=find(parent_1(min(points)+j)==parent_2);
        ord_1_2(j)=find(parent_2(min(points)+j)==parent_1);
     end
    child_1=[parent_1(1:points(1)); parent_2(ord_2_1);
              parent_1(points(points(2):end))];
    child_2=[parent_2(1:points(1)); parent_1(ord_1_2);
             parent_2(points(points(2):end))];
end
```

Mutation function

This part of the code is used to define the mutation function that will be a 2-points mutation. Since the population was customized for the industrial case under analysis, default options built in MATLAB are not available. Thus, the main effort and attention is put on the correct definition of the two points along the arrays.

Input: Indexes of the selected elements of the population
Output: set of individuals with permutated elements with respect to the 2-points mutation function
Logic:

```
for i=1:(number_of_selected_individuals)
    parent=Population(indexes(i));
    points=sort(randi(number_of_jobs,2,1), 'ascending');
    child=parent;
    child(points(1))=parent(points(2));
    child(points(2))=parent(points(1));
end
```

Fitness function

This part of the code is used to evaluate the fitness function. The simulation model, developed in SIMULINK (see Section 5.2.2), is used as test-bench through which the fitness is evaluated after running the simulation with the sequence represented by the individual.

Input: one individual of the population every time the function is called
Output: parameters concerning the production system performances and constraint satisfaction rates
Logic:

```
for i=1:population_size
   [mill_util,score_feas,score_tol,score_timediff]=simulation(individual);
end
```

GA options setting and launch

This part of the code is devoted to the correct setup of the GA in the MATLAB environment, defining all the correct options and linking the previously defined functions to the GA, which will recall them every time needed.

After the setting up of the GA, the algorithm is launch with the final command presented and the results are collected and analysed.

Input: functions developed before that will complete the GA setting
Output: the jobs scheduling (sequence, machine allocation and time allocation) and the fitness function value
Logic:

```
options=optimoptions(@ga,'PopulationType','custom',...
                     'PlotFcn',@gaplotbestf,...
                     'CreationFcn',@create_pop, ...
                     'CrossoverFcn',@crossover_permutation, ...
                     'MutationFcn',@mutate_permutation, ...
                     'MaxGenerations',10,'PopulationSize',10, ...
                     'MaxStallGenerations',8,'UseVectorized',true, ...
                     'SelectionFcn',@selectionroulette);

[x, fval, ~, ~, popf]=ga(FitnessFcn,NVARS,[],[],[],[],[],[],[],[],handles.options);
```

6 Conclusion

The framework must still be improved in terms of computational effort, but represents a good contribution on the topic of scheduling techniques. In fact, it proposes a novel approach that exploits GAs and simulation together. This has been possible thanks to a careful modelling of the production system and development of the algorithm steps.

The schedule obtained as output of the optimization process has been compared to the production data recorded by the Company and it has been approved since it satisfied all the production constraints and the simulated functioning is aligned with the real functioning of the production system.

The implementation of the scheduling framework would be exploited by the Company to program the daily production, at reducing difficulties in programming activities from which can derive errors and a decrease of the production system performances.

Having applied this non-conventional method for scheduling operations in a real industrial case has proved its efficacy, mostly on those systems that have to face complex managerial and logistics problems.

The introduction of a simulation model within the optimization algorithm, overcomes the typical problems related to traditional scheduling algorithms that usually lead to a large number of simplification hypotheses and assumptions that result most of the times too restrictive for real industrial cases.

Moreover, the utilization of a simulation model within the framework paves the way for the exploitation of the Industry 4.0 paradigm. In fact, the simulation model could be connected to the real production system thanks to intelligent sensors installed in the machines that provide Big Data streams to the model itself. In this case, a digital twin will be created: the virtual system becomes able to know the actual status of the machines of the physical one and automatically be updated. Considering again the aim of the framework, the direct consequence of this approach is that the produced schedules will be always consistent with the condition of the real production system, avoiding the necessity of manually setup the model and decreasing the possibility of errors.

The use of Industry 4.0 technologies applied to a scheduling framework, such as the one proposed in this contribution, would also open the way to interesting research investigations in the field of production management, such as the so-called Synchro-push paradigm (Garetti et al., 2016). In fact, the proposed scheduling framework could be used to boost the manufacturing responsiveness and efficiency, thanks to the possibility to support the connection between planning and scheduling activities and the real status of the manufacturing system, being able to run the GA-based and simulation-based scheduling optimization at any time.

The presented work could be further expanded in various directions. Some ideas include the following:

- Introduction within the scheduling framework of an additional block, named Statistics checking (Figure 8), where a statistical analysis of the schedule performances is done, when input data contain some stochastic variables. This is useful when machines have a variable behaviour, i.e. production parameters such as processing times and temperatures are no longer deterministic. This implies that, even if always the same sequence is simulated, the resulting schedule can be different at every run, so it is necessary to launch the simulation of the same sequence more times to have a probabilistic understanding of the behaviour of the overall systems. As the input data vary, also the output performances vary with respect to the run and so it is no longer possible to associate a single value of the performance index to an individual, but this single value becomes a confidence interval. Once a set of simulation runs for every sequence has been performed, the statistics checking block advances an ANOVA analysis on the population of the current iteration in order to understand whether the sequences are statistically different. After this analysis it is possible to provide a ranking of the different sequences of the population in order to choose which are the best ones to continue the optimization process. The ranking can be based on heuristic rules that take into account either the peak value of the performance or its variability, or on the intervention of a human decision maker that chooses according to his own experience;
- The simulation model could be designed to have a higher modularity in order to be have the capability to be easily reconfigurable to follow changes in the production system. This could be obtained thanks to the realization of standard blocks representing generic machines. This issue can be very relevant in those production environments where operation conditions dynamically change, e.g. where machines are often under maintenance and the actual configuration of the production system continuously vary, or where the configuration changes due to the purchase of a new machine, done to increase the production capacity, etc. Thus, the modularity of the simulation model insures a rapid setup of the scheduling framework, without much effort in programming activities in this way facilitating applicability.

7 References

Abramovici, M., Göbel, J.C. & Neges, M., 2015. Smart Engineering as Enabler for the 4th Industrial Revolution. In M. Fathi, ed. *Integrated systems: Innovations and applications*. pp. 163–170.

Asadzadeh, L., 2015. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, 85, pp.376–383.

Ashton, K., 2009. That ' Internet of Things ' Thing. *RFiD Journal*, 22(7), pp.97–114.

Blum, C. & Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3), pp.268–308.

Brandolese, A., Pozzetti, A. & Sianesi, A., 1991. Gestione della produzione industriale. *Hoepli, Milano*.

Buckles, B.P., Frederick, E.P. & Kuester, R.L., 1990. Schema survival rates and heuristic search. In *Tools for Artificial Intelligence*. pp. 322–327.

Çaliş, B. & Bulkan, S., 2015. A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, 26(5), pp.961–973.

Cavalieri, S. et al., 2000. Experimental benchmarking of two multi-agent architectures for production scheduling and control. *Computers in Industry*, 43(2), pp.139–152.

Cavalieri, S., Crisafulli, F. & Mirabella, O., 1999. A Genetic Algorithm for Job-shop Scheduling in a Semiconductor Manufacturing System. *25th Annual Conference of the IEEE Industrial Electronics Society*, 2, pp.957–961.

Cavalieri, S., Terzi, S. & Macchi, M., 2007. A Benchmarking Service for the evaluation and comparison of scheduling techniques. *Computers in Industry*, 58(7), pp.656–666.

Chen, H., Ihlow, J. & Lehmann, C., 1999. A Genetic Algorithm for Flexible Job-Shop Scheduling. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 2(May), pp.1120–1125.

Cheng, R., Gen, M. & Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. *Computers & Industrial Engineering*, 30(4), pp.983–997.

Chryssolouris, G. & Subramaniam, V., 2001. Dynamic scheduling of

manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, 12(3), pp.281–293.

Coley, D.A., 1999. *An introduction to genetic algorithms for scientists and engineers*, World scientific.

Falkenauer, E., Bouffouix, S. & Roosevelt, F.D., 1991. A Genetic Algorithm for Job Shop. In *Robotics and Automation*. pp. 824–829.

Framinan, J.M., Leisten, R. & García, R.R., 2014. Manufacturing scheduling systems. *An integrated view on Models, Methods and Tools*, pp.51–63.

Fumagalli, L. et al., 2017. Simulation-supported framework for job shop scheduling with genetic algorithm. In *Proceedings of the XXII Summerschool of Industrial Mechanical Plants "Francesco Turco", Palermo (Italy), 13th-15th September 2017*. pp. 1–8.

Garey, M.R. & Johnson, D.S., 1975. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4), pp.397–411.

Gonçalves, J.F., De Magalhães Mendes, J.J. & Resende, M.G.C., 2005. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1), pp.77–95.

Holland, J.H., 1975. *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence*,

Huang, J. & Süer, G.A., 2015. A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels. *Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems*, 86, pp.29–42.

Ishikawa, S., Kubota, R. & Horio, K., 2015. Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 42(24), pp.9434–9440.

Jalilvand-Nejad, A. & Fattahi, P., 2015. A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. *Journal of Intelligent Manufacturing*, 26(6), pp.1085–1098.

Joseph, O.A. & Sridharan, R., 2011. Effects of routing flexibility, sequencing flexibility and scheduling decision rules on the

performance of a flexible manufacturing system. *The International Journal of Advanced Manufacturing Technology*, 56(1), pp.291–306.

Jourdan, L., Basseur, M. & Talbi, E.-G., 2009. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3), pp.620–629.

Kacem, I., 2003. Genetic algorithm for the flexible job-shop scheduling problem. *International Conference on Systems, Man and Cybernetics SMC'03*, 4, pp.3464–3469.

Koza, J.R., 1995. Introduction to genetic algorithms.

Kumar, N.S.H. & Srinivasan, G., 1996. A genetic algorithm for job shop scheduling — a case study. *Computers in Industry*, 31(2), pp.155–160.

Kurdi, M., 2015. A new hybrid island model genetic algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, 88, pp.273–283.

Kurdi, M., 2016. An effective new island model genetic algorithm for job shop scheduling problem. *Computers and Operations Research*, 67, pp.132–142.

Lee, J., Kao, H. & Yang, S., 2014. Service innovation and smart analytics for Industry 4.0 and big data environment. In *Procedia CIRP*. Elsevier B.V., pp. 3–8. Available at: http://dx.doi.org/10.1016/j.procir.2014.02.001.

Li, X. & Gao, L., 2016. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, 174, pp.93–110.

Liao, C.J. et al., 2013. Meta-heuristics for manufacturing scheduling and logistics problems. *International Journal of Production Economics*, 141(1), pp.1–3.

Lu, P.H. et al., 2015. A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *Journal of Intelligent Manufacturing*.

Mallikarjuna, K., 2014. A Review On Job Shop Scheduling Using Non-Conventional Optimization Algorithm. *International journal of engineering Research and applications*, 4(3), pp.11–19.

May, G. et al., 2015. Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research*, 53(23), pp.7071–7089.

Negri, E., Fumagalli, L. & Macchi, M., 2017. A review of the roles of Digital Twin in CPS-based production systems. *Procedia Manufacturing*, 11(June), pp.939–948. Available at: http://dx.doi.org/10.1016/j.promfg.2017.07.198.

Nocedal, J. & Wright, S.J., 2006. *Numerical Optimization*,

Palmer, M., Liepins, G.E. & Hilliard, M.R., 1988. Machine Learning Applications To Job Shop Scheduling. *Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems-Volume 2*, pp.728–737.

Park, J., Mei, Y., Nguyen, S., Chen, G. and Zhang, M. (2018) 'An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling', *Applied Soft Computing*. Elsevier, 63, pp. 72–86. doi: 10.1016/J.ASOC.2017.11.020.

Pezzella, F., Morganti, G. & Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10), pp.3202–3212.

Ponnambalam, S.G., Ramkumar, V. & Jawahar, N., 2001. A multiobjective genetic algorithm for job shop scheduling. *Production planning & …*, 12(8), pp.764–774.

Rodammer, F.A. & White, K.P., 1988. A Recent Survey of Production Scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 18(6), pp.841–851.

Ruiz, R. & Maroto, C., 2005. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2), pp.479–494.

Salido, M.A. et al., 2016. A genetic algorithm for energy-efficiency in job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, 85(5-8), pp.1303–1314.

Vilcot, G., Billaut, J.-C. & Esswein, C., 2006. A genetic algorithm for a bicriteria flexible job shop scheduling problem. In *Service Systems and Service Management, 2006 International Conference on*. IEEE,

pp. 1240–1244.

Xhafa, F., 2008. *Metaheuristics for Scheduling in Industrial and Manufacturing Applications* A. Abraham, ed., Springer Berlin Heidelberg.

Xhafa, F. & Abraham, A., 2008. *Metaheuristics dor Scheduling in Industrial and Manufacturing Applications*,

Xing, Y. et al., 2007. An improved adaptive genetic algorithm for job-shop scheduling problem. *Third International Conference on Natural Computation (ICNC 2007)*, 4, pp.287–291.

Xing, Y. et al., 2006. An Improved Genetic Algorithm with Recurrent Search for The Job-Shop Scheduling Problem. *2006 6th World Congress on Intelligent Control and Automation*, 1, pp.3386–3390.

Yan, H.-S., Wan, X.-Q. & Xiong, F.-L., 2015. Integrated production planning and scheduling for a mixed batch job- shop based on alternant iterative genetic algorithm. *Journal of the Operational Research Society*, 66(8), pp.1250–1258.

Ying, W. & Bin, L., 1996. Job-shop scheduling using genetic algorithm. *Systems, Man, and Cybernetics IEEE International Conference*, 3, pp.1994–1999.

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

Zhiming, W. & Chunwei, Z., 2000. Genetic algorithm approach to job shop scheduling and its use in real-time cases. *International Journal of Computer Integrated Manufacturing*, 13(5), pp.422–429.

Zhou, H., Feng, Y. & Han, L., 2001. The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engineering*, 40(3), pp.191–200.

Author

FIGURES

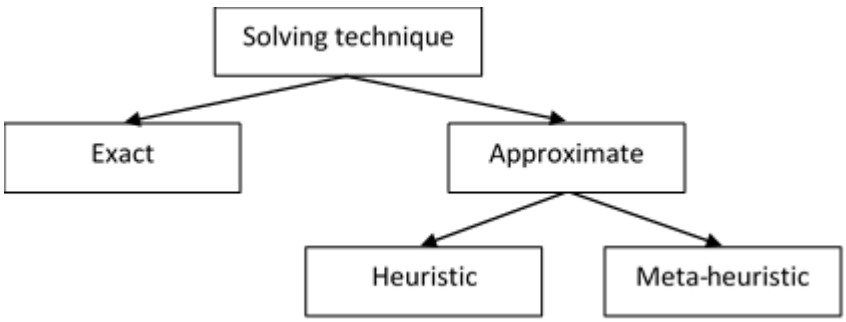

Figure 1 - Resolution method scheduling problem classification

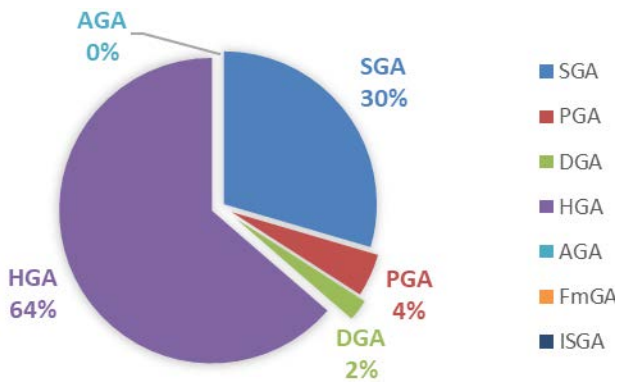

Figure 2 - GA types in articles between 2012 and 2016

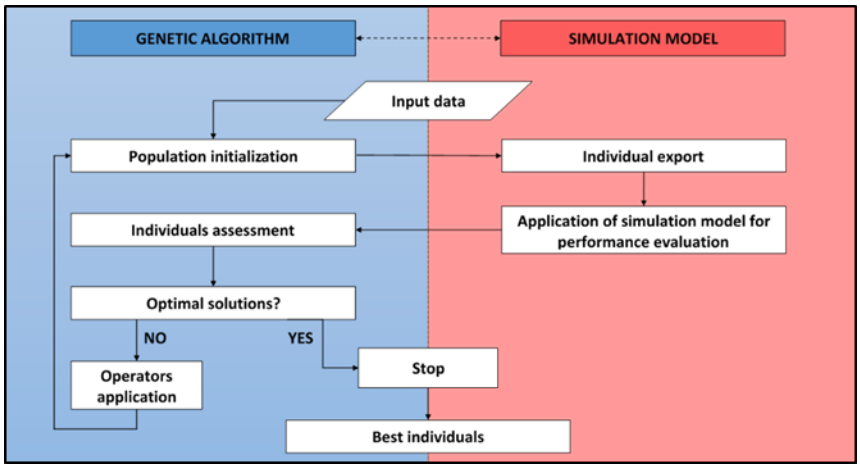

Figure 3 - Proposed simulation-supported framework for JSSP with GA

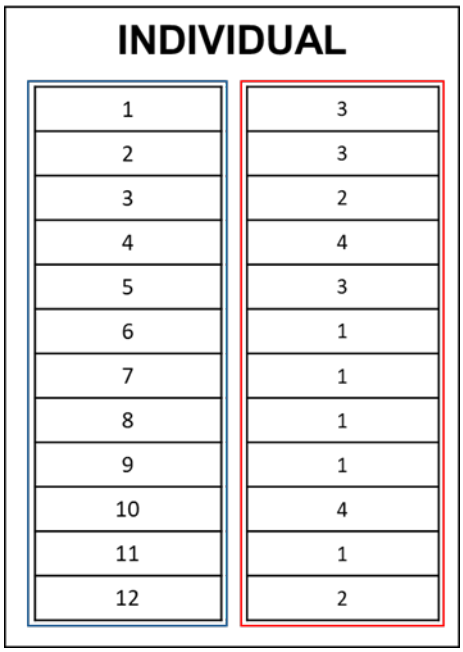

Figure 4 - Individual representation.

Author

Figure 5 - Simulation model phases

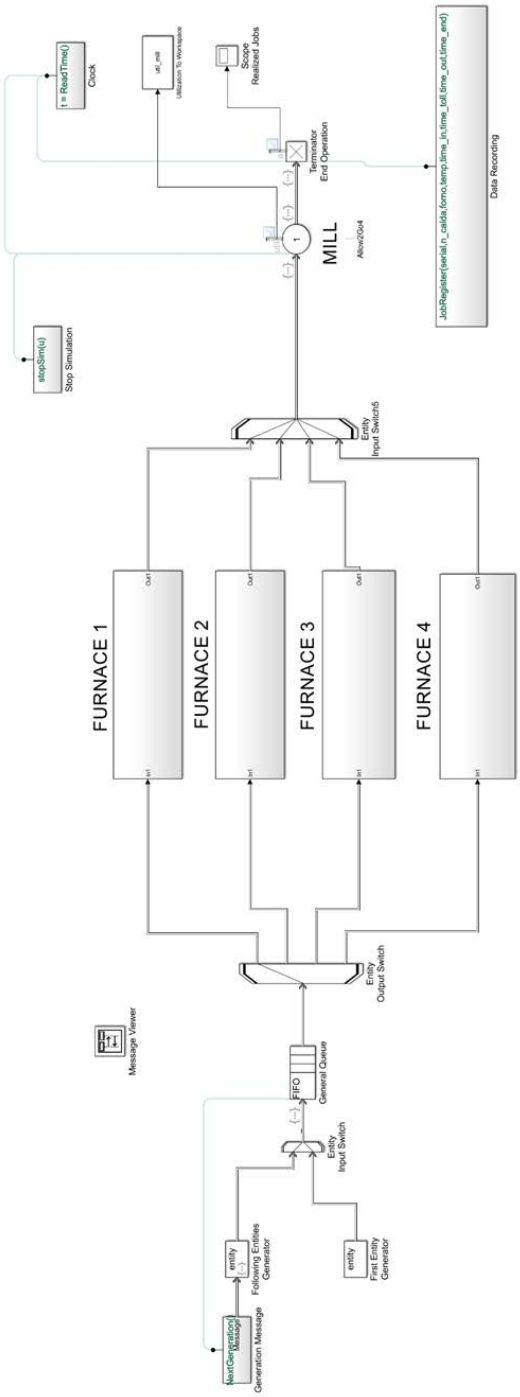

Figure 6 - Simulation model implementation in the Simulink environment

Author

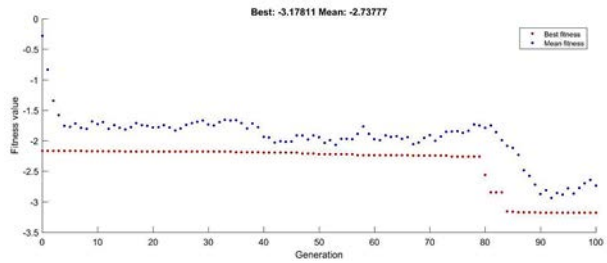

Figure 7 - Graphical representation of framework functioning

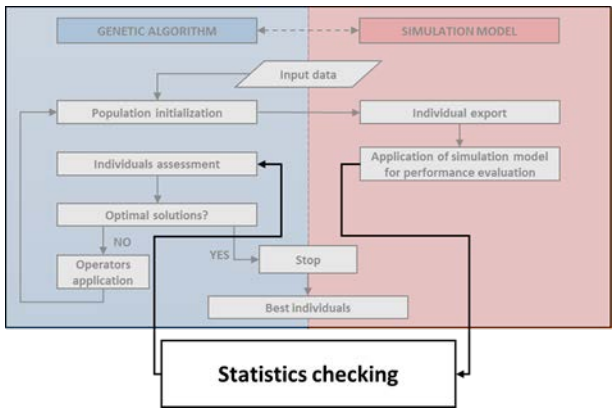

Figure 8 – Stochasticity-improved framework