Simulation-supported framework for job shop scheduling with genetic algorithm

Fumagalli L., Polenghi A., Sottoriva E., Negri E., Macchi M.

Department of Management, Economics and Industrial Engineering of Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan – Italy (luca1.fumagalli@polimi.it– corresponding author -, marco.macchi@polimi.it, elisa.negri@polimi.it)

Abstract: The Job Shop Scheduling Problem (JSSP) is recognized to be one of the most difficult scheduling problems, being NP-complete. During years, many different solving techniques were developed: some techniques are focused on the development of optimization algorithms, whilst others are based on simulation models. Since the 80s, it was recognized that a combination of the two could be of big advantage, matching advantages from both sides. However, this research stream has not been followed to a great extent. The goal of this study is to propose a novel scheduling tool able to match these two really different techniques in one common framework in order to fill this gap in literature. The base of the framework is composed by a genetic algorithm (GA) and a simulation model is introduced into the evaluation of the fitness function, due to the inability of GAs in taking into account the real performances of a production system. An additional purpose of this research is to improve the collaboration between academic and industrial worlds on the topic, through an application of the novel scheduling framework to an industrial case. The implementation to the industrial case suggested an improvement of the tool: the introduction of the stochasticity into the proposed scheduling framework in order to consider the variable nature of the production systems. This novel framework overcomes all the afore-mentioned criticalities and the practical confirmation may allow thinking that, with further developments, it will outperform all the other scheduling techniques. Moreover, this proposed scheduling framework contributed to the ongoing investigation related to the recently formulated Synchro-push concept, a production control paradigm that relies on an integrated view of the shop floor logistics and the planning of production.

Keywords: scheduling framework; job shop; JSSP; genetic algorithm; simulation; software selection.

1.Introduction and objectives of the work

The scheduling activity has started to play an important role in managing correctly and efficiently a production system, resulting in great competitive advantages. Since this trend became evident also the academic world started researching this topic; from the eighties, the number of scientific articles describing new algorithms or methods to schedule activities has grown in number and also nowadays the scheduling-related works produced every year are numerous. Related literature has considered two aspects: the development of new techniques or methods specific for scheduling and the adaptation of already existing algorithms to it.

This paper contributes to this relatively new research branch. The goal of this work is not to provide a ready-to-use solution for scheduling problem, but a base framework that should be customized every time to adapt to the new problem. Since job shop scheduling has been demonstrated to be NP-complete (Garey & Johnson 1975), it has been addressed by many researchers due to its hardness. This framework focuses onto the JSSP, proposing a structure composed by two distinct methodologies already present in the literature: the genetic algorithm (GA) and the simulation. The development of this tool is made by means of MATLAB, especially the capabilities of the Simulink environment to reproduce the working way of the production system. The application of such structure to a real industrial scenario shows its effectiveness and it suggests also further developments, including the introduction of stochasticity in the framework, to represent the production system variability. The paper is so organized: Section 2 proposes an analysis of the development of the GAs with respect to the production systems; Section 3 describes the proposed scheduling framework for the JSSP; Section 4 deals with the validation of the proposed framework, describing respectively the industrial case and the application to real scenario; in Section 5 conclusions are drawn, work limitations are explained and future improvements are suggested.

2. Genetic algorithms literature

Since the development of the genetic algorithm theory (Holland 1975), its adaptation to production systems seems to be obvious, due to the tendency of such algorithms to find the best option among several candidates. It was further forecasted that GAs are going to be successful in problems like scheduling, implementing one of the first algorithms based on evolution process, which is able to improve the efficiency of the solution with respect to an algorithm-based approach (Palmer et al. 1988). In the same year, (Rodammer & White 1988) analysed seven paradigms used in the scheduling activities: industrial practice, machine sequencing and scheduling theory, resource-constrained project scheduling, control theory, discrete event simulation, stochastic optimization and artificial intelligence (AI).

The GAs are notoriously highly time-consuming algorithms; therefore, many studies are focused on the optimization of their performances. (Buckles et al. 1990) tried the reduction of number of individuals in the population, based on the schema theorem. Three novel algorithms are proposed by (Ying & Bin 1996) with the aim of reducing the searching space. Instead, (Falkenauer et al. 1991) focused on special crossover and mutation operators to meet job shop scheduling requirements. (Davis et al. 1993) started doubting about the effectiveness of the individual use of GAs and assumed the integration with other methods would improve the scheduling performance. A real achievement for JSSP with GA is reached by (Kumar & Srinivasan 1996), whose application with integration of dispatching rules enhanced the makespan reduction of about 30% of the actual production system. (Cavalieri et al. 1999) implemented a GA algorithm for a flexible job shop (FJS), which is recognized to be the most complex among JSSP (Chen et al. 1999). In the same research field, (Zhiming & Chunwei 2000) proposed another GA for FJSSP and underlined the necessity to integrate GA with other methods. Contemporarily, a MOGA (multi objective genetic algorithm) is developed by (Ponnambalam et al. 2001) that is focused on avoiding the possibility of GA to get stuck in local optima. The same year, (Chryssolouris & Subramaniam 2001) studied a GA-based algorithm able to take into account random dynamic events, multiple scheduling criteria and multiple job routes, enhancing the approaching to real production systems. (Kacem 2003) developed a GA-based scheduling algorithm that considers performance objectives more consistent with intrinsic behaviours of the manufacturing system, that are: makespan, workload of the critical machine and total workload of all the machines. Many authors continued the study on performance improvement of GAs by acting on their parameters: (Gonçalves et al. 2005) introduced random keys codification; (Xing et al. 2006) developed a continuously updated GA able to change its routing according to the fitness function of each individual; (Xing et al. 2007) worked on operator probability.
As can be deduced from the above discussion, GAs could be improved through highly problem-specific operators or through hybridization with other methods. Especially this last way has been highly followed in the last years: a survey collecting more than fifty scientific articles was realized by the authors and the results are shown in Figure 1, where AGA is adaptive genetic algorithm, SGA is simple genetic algorithm, PGA is parallel genetic algorithm, DGA is distributed genetic algorithm and HGA is hybrid genetic algorithm (Sivanandam 2008).

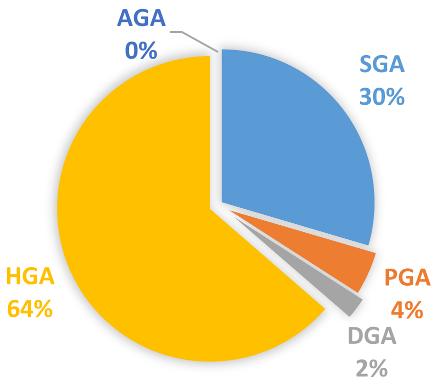


Figure 1. GAs survey for articles between 2012 and 2016.

As it appears evident, the trend is to focus on hybridization. The shortcomings derived from this approach are related to the ability of the hybrid algorithms to efficiently represent the functioning of a production systems, taking into considerations all the constraints and management issues typical of the manufacturing companies. This objective could be reached by means of a simulation model and the goal of the actual research is to enlarge the possibilities given by the HGA: the hybridization would not consider only other algorithms in supporting GA, but also simulation could be a useful tool to improve its capabilities. The proposed framework claims the integration of these two methods to form a complete structure able to optimize the scheduling without the necessity to make simplified assumptions for describing the production system.

3.Proposed scheduling framework

The proposed scheduling framework lies its key characteristics in the connections and in the continuous exchange of data and results between the GA and the simulation model. The optimization process follows the iterations of the GA: at every iteration, the GA interacts with the simulation model receiving the performance parameters to be fed into the fitness function. The GA performs all the needed operations to reach the identification of the best-fit solution, meanwhile the simulation model processes the input individuals created by the GA, each of them representing a possible production schedule. The overall framework is reported in Figure 2, in which the main steps and the flow of data are depicted.

3.1 Input data

The

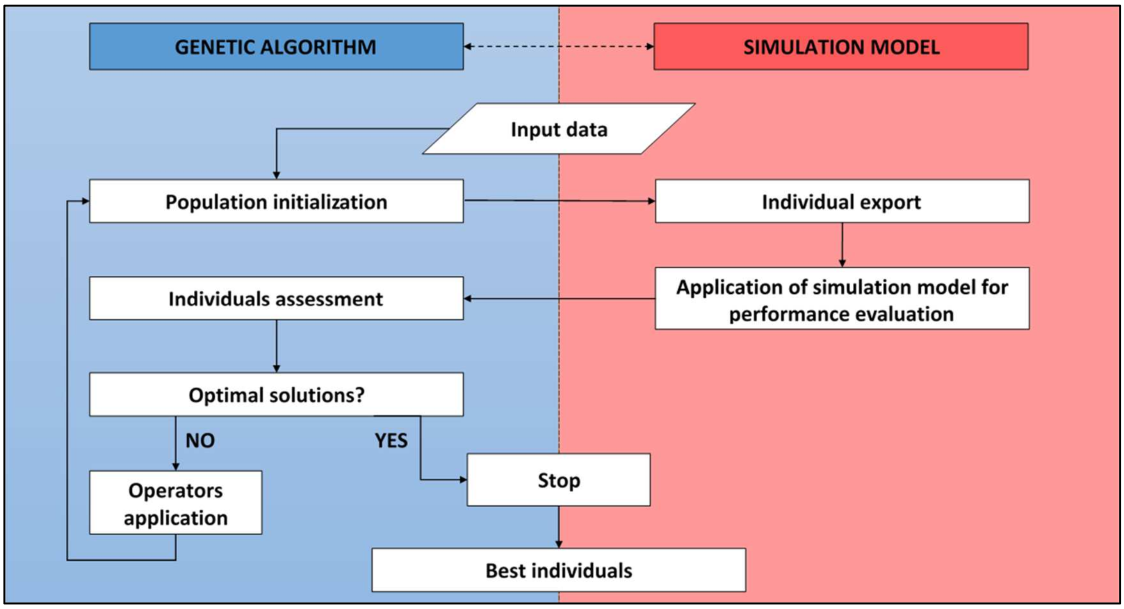


Figure 2. Proposed simulation-supported framework for JSSP with GA.

proposed scheduling framework receives the workpieces to be produced. By knowing their production cycle, automatically the set of operations to carry out within the production system, and so to schedule, is defined. For the sake of simplicity, an operation has been defined as *Job*. Additional input data are the machines capable to perform certain operations and their state.

3.2 Population initialization

A GA evaluates the fitness of a defined number of points of the search space at every iteration until the best-fit one is found. A point of the search space is a so-called individual and it has a double representation: for the JSSP point of view, an individual is a schedule (phenotype), instead for the GA point of view is the codified representation of the schedule (genotype).
To pass from one representation to the other, an encoding or decoding process must be carried out. The code chosen for the purpose of this work is a double array codification.
The first array represents the sequence of jobs (every number univocally represents a job) in the order they are received from the scheduling tool. The second array, contains the same number of digits of the job sequence and every one of them represents the machine associated to the corresponding job.
The population of each iteration of the GA is obtained through a defined number of permutations of the rows of an individual, both of the first and of the second array.
The proposed codification allows to associate at every operation to perform, the machine where to perform it and in this way to explore much more points of the search space because the permutations are doubled.

3.3 Individual export

Once the GA has generated a population, the simulation model imports one individual at a time. This passage seems to be automatic, but actually it performs the de-coding activity, that is typically performed by a piece of programming code (i.e. the interface between the GA and the simulation model), in order to translate the information contained in the two arrays into a format that is understood by the simulation model.

3.4 Application of simulation model for performance evaluation

The next step is the actual simulation of the sequence of jobs related to the specific imported individual and in that sense the model gives as output one or more performances of the simulated sequence, that will be later used by the GA. In fact, the GA will optimize a fitness function that receives simulation-generated performances associated to the individuals as input.

3.5 Individuals assessment

At any iteration, when all the individuals of the population have been processed by the simulation model, their assessment occurs. After scaling and translating the raw scores obtained by the simulation model, the performance indexes are input into a fitness value univocally associated to each individual. Necessarily the fitness function is highly problem dependent and so it must be tailored to the problem under consideration. Then a ranking of the individuals of a population is done considering their fitness value.

3.6 Optimal solution research

When the GA has generated a ranking of individuals, it understands if in this ranking there is the optimal solution or not. If there is an optimal solution, the algorithm stops and gives the optimal or a sub-optimal schedule with respect to the considered performance indexes; if not, it goes on with a further iteration. In order to instruct the algorithm to make this decision, it is necessary to implement termination criteria, that may be:

- stall generation criterion: after a certain number of generations without improvements in the fitness value, the algorithm is stopped and the best solution is taken from the ones already generated;

- maximum number of iterations: if the algorithm reaches a defined upper limit on the number of iterations, the best solution found until then is chosen.

These criteria must be a trade-off between computational burden and search space exploration, because if they stop the algorithm too soon, the possibility is to fall in a local optimum that can be very far from the global one; if the criteria stop the algorithm only in a very long time, the possibility is to increase enormously the computational time, compromising the usefulness of the GA for practical applications in industry.

3.7 Operators application

If the best fit solution has not been identified and a new iteration is needed, the GA applies genetic operators to the current population of individuals in order to create a new population for the next iteration. Operators have the aim to select some individuals from the current population and in some way permutate them. Typically, genetic operators are fitness-based, which means that mostly the fittest individuals of the current population will be selected to transfer their data to the next population with the hope to improve the population. But this criterion could reveal as a double-edged sword, because if operators transfer data only considering the fittest individuals, it is possible to fall in a local optimum. For this reason, it is necessary to permutate also other individuals, randomly chosen within the search space.

The typical sequence of operators applied on the population is:

- *selection* of a defined number of individuals with a certain criterion;

- *crossover* on some of the selected ones, i.e. creating new individuals by mixing their characteristics;

- *mutation* on some of the selected ones, i.e. changing randomly some of their digits;

- *replacement,* i.e. replace some individuals of the current population with some of the newly created ones.

This procedure by steps is iterative and so the new formed population must be evaluated and the process starts again until an optimum is reached.

4.Validation

The proposed scheduling framework is applied to a case study based on a Company that wants to remain unnamed for privacy issues.
The Company produces forged and laminated rolled rings of several kinds of materials, from carbon and alloyed steels, to nickel, titanium, cobalt alloys, aluminium and copper. The production system is composed by many machines grouped in departments according to their functions. The production cycle of the parts under analysis is composed by several sequences of two operations: heating and milling; the latter is performed every time a piece exists the heating furnace. The systems could be classified as a job shop, in particular as a flexible job shop, because the allocation to furnaces for heating is not fixed, but it is demanded as an output of the scheduling activity. In the considered system, there is only one mill.
The application of the proposed scheduling framework to the real case scenario has involved several activities: data gathering, mapping, customization, results analysis. The goal is the maximization of the milling machine.

4.1 Data gathering

This operation is devoted to collect all necessary data to have a better overview onto the production cycle and especially onto the constraints, both physical and managerial, that should be taken into account. For an entire week, the collection of data in real time from the production cycle has been performed and the obtained schedules are organized as easily-readable Gantt charts.

4.2 Mapping

The mapping activity involves a deeper look into the system, analysing every input and output from every machine and the flow of parts on the shop floor. The final result is a description of all the constraints the actual schedule is forced to respect. The summary is presented in Table 1.

4.3 Application

The application to the real scenario passed through an extensive customization to introduce every constraint according to Table 1: some of them are implemented in the GA, whilst some others in the simulation model.

4.3.1 GA implementation

The population size of the genetic algorithm is a trade-off between two counteracting effects for which an optimum has still to be found (Goldberg et al. 1991). If the population size is small, then the computational time required for the calculation is less, but the convergence to the optimum is not guaranteed due to low samples onto which the simulation could be run. If the size is too big, the time consumed by GA is very high, but the convergence is most probable.
The encoding is double: two arrays are representative for the individuals. The first array contains the information

about the jobs and the sequence in which the jobs are ordered is the schedule to be tested by the simulation model. The second array represents the allocation to the furnace of each job; the allocation of the jobs to the mill is not considered since there is only one mill and so all the jobs will visit it. Some authors implemented also a three-dimensional encoding, but the mole of data to be managed would increase exponentially (Yin et al. 2007).

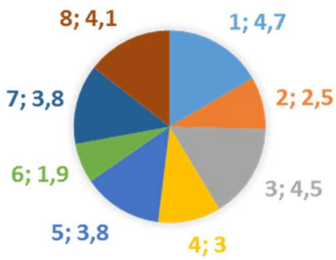
Table 1. Constraints and their implementation.

Object	Constraint	Explanation	Implementation
Pieces	Batch homogeneity	Pieces of the same batch must have homogeneous characteristics.	Simulation Model
Furnace	Specification	Not all the part numbers can be processed in all the furnaces. When a part number has been assigned to a furnace, it must complete its production cycle in that furnace.	Genetic Algorithm
	Temperature	Every part number has to be processed at a specified temperature. The temperature change between the processing of two different part numbers implies a setup time.	Simulation Model
	Capacity	Furnaces have limited capacity. It varies with respect to the part number.	Simulation Model
	Service time and tolerance	The stay of a piece in the furnace depends on the part number and it is subjected to tolerance. A piece cannot stay inside a furnace more than the service time plus the tolerance time.	Simulation Model
Mill	Capacity	The mill can process only one piece at a time.	Simulation Model
	Cycle time	Once a piece has finished its process on the mill, it must immediately return into the furnace for the subsequent operation (if there is any).	Genetic Algorithm

The chosen *selection* option is the roulette wheel method: each individual is associated to a slice of a roulette wheel, whose dimension is proportional to the fitness value of the individual itself. In this way well-scored individuals have higher probability to be chosen for reproduction (Mitchell 1998). An example is shown in Table 2.

Table 2. Roulette wheel example.

Sequence ID	Score	Roulette wheel
1	4.7	
2	4.3	
3	4.5	
4	3	
5	3.8	
6	3.9	
7	3.8	
8	4.1	



The operators devoted to evolve individuals generation by generation are both *crossover* and *mutation*: they are 2-points operators, so two points of each sequence are selected and the crossover and mutation act as in Figure 3 and Figure 4, respectively.

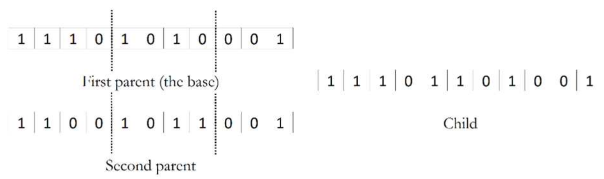


Figure 3. Crossover operator.

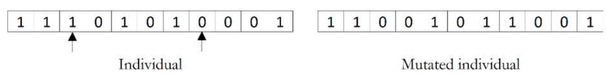


Figure 4. Mutation operator.

The fitness function is one of the most important factors in the GA iterations because it influences how and if GA will converge to the optimal solution. The expression of the fitness function is presented in Equation 1: the term $score_{feas}$ represents the feasibility of a sequence in terms of precedence rules (i.e. operation 1 must be performed before operation 2); the term $score_{tol}$ describes the respect of the furnace tolerance (ref. to Table 1, Service time and tolerance of furnace); the term $score_{timediff}$ verifies the cycle time constraints (ref. to Table 1, Cycle time of mill); the term $score_{millutil}$ is the performance the Company is interested in optimizing (i.e. the mill utilization).

Equation 1. Fitness function.

$$fitness = -\,score_{feas} - \,score_{tol} - \,score_{timediff} - score_{millutil}$$

The negative value of the fitness function derives by the need of optimization algorithms, which are settled to find the minimum of a function (as default of such Matlab function). The maximization of the mill utilization is mathematically formulated as the minimization of its opposite.
The stopping criteria are the maximum number of iterations, set to 100, and maximum number of stall generations, set to 30. Table 3 summarizes the described characteristics of the applied GA.

Table 3. GA parameters summary.

Parameter	Value/Description
Population size	100
Encoding	Double encoding with one array devoted to operations and corresponding loci in the second array highlighting the machine
Selection	Roulette wheel
Crossover	2-points
Mutation	2-points
Fitness function	Combines functions of real performance index (mill utilization) and constraints satisfaction
Stopping criteria	Maximum number of iterations (100) and maximum number of stall generations (30)

The GA is implemented into the MATLAB environment; the overall script will launch the production system model while GA is evaluating the fitness function.

4.3.2 Simulation model implementation

The simulation model is coded into the MATLAB/Simulink environment in order to favour the integration of the two parts of the proposed scheduling framework.
The model should take into account all the constraints not yet satisfied by the GA, i.e. batch homogeneity, furnace temperature (setup time), furnace capacity, service time and tolerance of the furnace, service time and capacity of the mill (ref. to Table 1).
The furnace tolerance is implemented both by GA and simulation model, because it must be dealt with in two moments: in the simulation, the constraint is modelled and the GA evaluates the performance related to it for the current sequence.
A scheme of the Simulink model structure is presented in Figure 5.

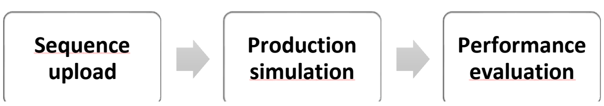


Figure 5. Simulation model phases.

The higher level simulation model is presented in Figure 6: the left part is the one devoted to the sequence upload phase and it is composed by entity generators and queues; the central part (both furnaces and mill) is the core of the model and it is the part responsible for the simulation in a strict sense; the remaining blocks do not represent a physical process or resource, they provide performance values and data to the GA, which elaborates them to calculated the fitness for each individual.

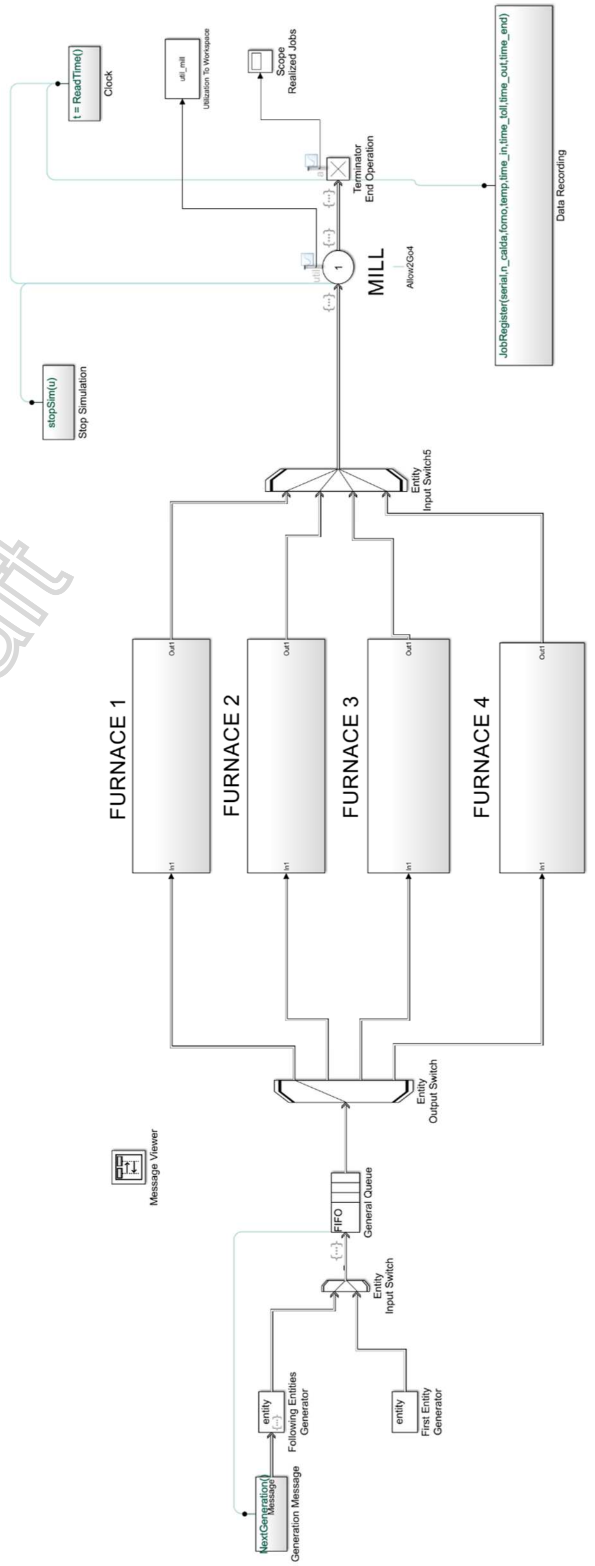


Figure 6. Simulink model.

4.4 Results

Once the overall framework has been implemented thanks to MATLAB capabilities, the outputs are: an optimal sequence of jobs and a plot showing the way pointing toward the optimum that has been followed by the simulation-supported GA (Figure 7).

For the Figure 7 it seems that there is a long central plateau, however, since the stall generation stopping criterion did not act, it can be concluded that every iteration had effectively produced an improvement in the fitness function, despite this might be very little.

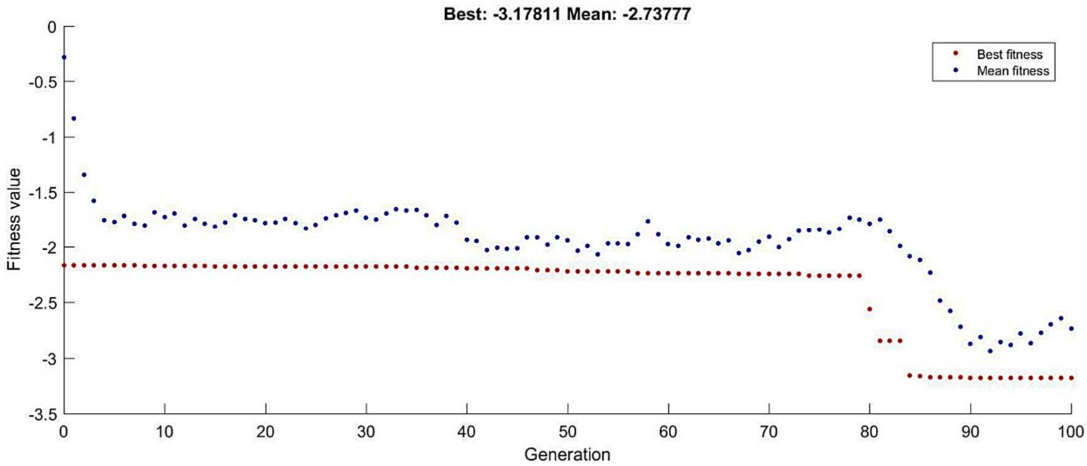


Figure 7. Graphical representation of framework functioning.

The output sequence is provided as a MATLAB-shaped table in which every row, representing a job, is characterized by production times in order to support the Company in creating a scheduling chart, like a Gantt chart.
The table itself is no reported due to privacy issues. The Company does not want to show any of their production parameters.

5. Conclusions

The results were analysed together with the Company. The output sequence was recognized to be aligned with the production constraints and the logics implemented into the production model appropriately simulate the functioning of the manufacturing system.
The Company is interested in exploiting the scheduling framework in the MATLAB environment every day to schedule the day-by-day production, to improve the efficiency in the short time horizon, but with a look to an increase of the production performance also in the long-term.
The application of the proposed simulation-supported framework with GA to a real industrial case demonstrates the efficacy of the model in finding a well-performing solution to the job shop scheduling problem.
The introduced innovation is especially suitable for those cases characterized by a complex managerial and logistic structure of the production systems. The simulation has the intrinsic capability to overcome all the algorithms in the manufacturing system modelling and its integration with GA leads to an interesting innovative contribution to the knowledge on the topic.

5.1 Work limitations

The proposed scheduling framework has been tested only when facing the job shop environment, thus the efficiency with respect to the other configurations has not been tested. However, the job shop is the most general plant type, so the authors are comfortable to affirm its efficiency also in addressing other environments.
Moreover, the GA presents high computation times that hinder a full industrial exploitation at this time. However, the work on the GA can be improved to speed up the algorithm running and to find an optimal solution in a faster way.

5.2 Future developments

The proposed research work has not concluded its potential, in fact further steps to carry on and improve the proposed framework have already been identified and started.

- Improvement of the scheduling framework by the integration of an additional block, called *Statistics checking*, able to manage the stochasticity of the production system: more runs for every sequence will be carried out and a confidence interval is created for each individual instead of a single performance evaluation, for a more robust GA optimization (Figure 8);

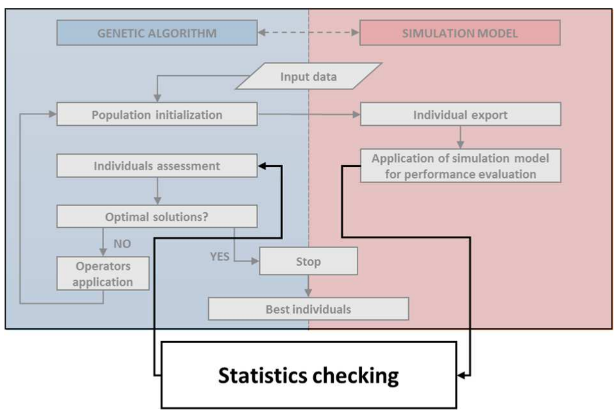


Figure 8. Stochasticity improvement framework.

- Modularity objective: creation of standard blocks able to represent a generic machine in order to create a user-friendly simulation model that could be scalable and reconfigurable in a fast way.

This research work has already been the launching base for future interesting investigations related to the so-called Synchro-push paradigm (Garetti et al. 2016), as the developed framework proposes an optimization of scheduling for an increased manufacturing responsiveness and efficiency and by leveraging on the possibility to run the simulation-based GA optimization at any time supports the close connection between planning and scheduling systems and the real manufacturing systems operations.

References

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

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.

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.

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

Davis, W.J. et al., 1993. Recent advances in the modeling, scheduling and control of flexible automation. In *Proceedings of the 25th conference on Winter simulation*. ACM, pp. 143–155.

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

Garetti, M. et al., 2016. Synchro-push: a new production control paradigm. In *21st Summer School Francesco Turco 2016*. pp. 150–155.

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

Goldberg, D.E., Deb, K. & Clark, J.H., 1991. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, 6, pp.333–362.

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*,

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.

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.

Mitchell, M., 1998. *An introduction to genetic algorithms*, MIT press.

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.

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.

Sivanandam, S.N., 2008. *Introduction to Genetic Algorithms* S. N. Deepa, ed., Springer Berlin Heidelberg.

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.

Yin, H. et al., 2007. A Three-Dimensional Encoding Genetic Algorithm for Job Shop Scheduling. *Computational Intelligence and Security Workshops, 2007.CISW 2007.International Conference on*, pp.175–178.

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

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.