Simulation-based Optimization Using Genetic Algorithms for Multi-objective Flexible JSSP

Elena Simona NICOARĂ[1], Florin Gheorghe FILIP[2,3], Nicolae PARASCHIV[1]

[1] Petroleum-Gas University,
39, Bucureşti Blvd., Ploieşti, 100520, Romania,
snicoara@upg-ploiesti.ro

[2] Romanian Academy - INCE and BAR,
125, Calea Victoriei, Bucharest 010071, Romania,
filipf@acad.ro

[3] National Institute for Research & Development in Informatics –ICI Bucharest,
8-10, Maresal Al. Averescu Blvd., Bucharest 011455, Romania

Abstract: The fast technological progress, along with growing requirements in the manufacturing systems have led in the last decades to a true revolution regarding the optimization methods for *job shop scheduling problem* (JSSP), which regularly has the greatest impact on the global optimality from the temporal perspective. An extension to the mathematical framework associated to the JSSP for multi-objective flexible JSSP (MOFJSSP) is proposed; here, the *flexibility of type II*, where the routings of the jobs on the resources are not fixed is considered. Also, a short review of the most used simulation-based optimization methods for (MOF)JSSP is made and a genetic algorithm-based control system is proposed. This is then tested on a complex real-world MOFJSS instance and the *ft*10 test-instance.

Keywords: Multi-objective Flexible Job Shop Scheduling Problem, Simulation-based Optimization, Genetic Algorithm, GA-based Control, NSGA-II

1. Introduction

Generally speaking, a job shop scheduling problem (JSSP) is a decision-making process for time optimal assignment of some (limited) resources to some heterogeneous jobs consisting in many operations. The resources have to be available and the associated optimization problem is either mono-objective or multi-objective. This kind of scheduling places the problem in the *discrete-event systems* (DES) domain, whose optimal control often involves computer simulation, at least in the large-scale real-world manufacturing systems.

As shown in [11] the simulation-based optimization can be utilised in the decision-making process for DES. For the specific JSSP case, there are two main aspects which make the decision difficult, namely: a) the constraints can not be explicitly expressed related to the decision variables, and b) the number of the decision alternatives in the search space is huge.

Besides the trivial case when the number of decision alternatives is small to average, where simulation-based optimization consists in evaluating all alternatives to detect the one that provides the best value for the optimization criterion/criteria, the proper meaning of the simulation-based optimization refers to an ordered simulation sequence, determined by an algorithm, applied to different decision parameters until a (near) optimal solution is found [11].

This paper is concerned with simulation-based optimization appropriate to the *Multi-objective Flexible JSSP* (*MOFJSSP*). It is organised as follows. An extension of the classical formulation of JSSP to MOFJSSP is presented first. Next, the most used simulation-based optimization methods in the scheduling area are reviewed and a control method, based on a genetic algorithm, is proposed and the test results are presented.

2. Multi-Objective Flexible JSSP

MOFJSSP definition

The *flexible job shop scheduling* is meant to properly handle the manufacturing process flexibility related to the structure either of the jobs or of the resources (processing machines and transportation vehicles) during the scheduling.

In the *flexible JSSP* (*FJSSP*) *of type I*, there are a) jobs with alternative sequences of operations and b) operations which can be performed on sets of (identical or different) machines. In the *FJSSP of type II*, the jobs have fixed sequences of operations, but the routings of the jobs on the machines are flexible; in other words, there are operations which can be processed by any machine in a specified set [4].

There are various possible formulations for the (F)JSSP. The problem can be viewed either as *combinatorial optimization problem* or as *constraint programming problem*.

A combinatorial optimization problem has discrete solutions and is defined by objective and constraints. In this context, the (F)JSSP can be defined as *integer linear programming*. This formulation however hardly allows a practical method to attain the solutions [15]. Moreover such a formulation is computationally infeasible [14]. Though the *mixed integer linear programming* is another possible formulation, and a conceptually elegant one, according to [21], the number of integer variables grows exponentially with the instance dimension and it requires too many constraints to be satisfied. One of the most used formulations in this category is the *disjunctive programming* formulation where the solutions (schedules) are represented as disjunctive graphs. This is especially adequate to solving techniques based on graphs / trees, such as *branch and bound* methods.

In the *constraint programming* formulation only the constraints to be satisfied are set. The solutions do not have to extremize any objective function and, consequently, to well define the problem, combining it with an optimization formulation is necessary.

In this section an extension of the mathematical model of the deterministic predictive JSSP presented in [3] is made in order to include the type II flexibility. The *definition of the FJSSP* is approached as a combinatorial optimization formulation. The traditional assumptions are used, such as: a) the job release dates is time T=0, b) all the machines are available at time T=0, c) the number of machines and jobs are finite and constant in time (with respect to their characteristics), d) the processing times of the operations are finite and constant, and e) the probability for machine breakdowns and the setup times are statistically included in the processing times.

The *input data of FJSSP* are:

- a finite set M of m $(m \in Z)$ machines, where Z is the set of integers;
- a finite set J of jobs, each job $i \in J$ consisting in an ordered sequence of n_i operations, $o_{i,j}$, $j = \overline{1, n_i}$;
- for each operation $o_{i,j}$ $(i \in J,\ j = \overline{1, n_i})$, the set of machines which can perform it, $MA_{i,j} \subseteq M$, with $card(MA_{i,j}) \geq 1$, where $card(MA_{i,j})$ is the cardinality of the set $MA_{i,j}$, and the processing times $\tau_{i,j}^{k} \in Z$, $k \in MA_{i,j}$ are given.

Therefore, to each operation, $o_{i,j}$, one can associate a set $D_{i,j}$ of processing times:

$$D_{i,j} = \{\tau_{i,j}^{k} \in Z \mid i \in J, \ j = \overline{1, n_i}, \ k \in MA_{i,j}\} \tag{1}$$

A *candidate-solution* is a valid schedule for J, *which* is defined as a collection of machine schedules:

$$s_k : \{o_{i,j} \mid k \in MA_{i,j}, i \in J,\ j = \overline{1, n_i}\} \rightarrow Z,\ k = \overline{1, m} \tag{2}$$

which satisfies the constraints associated to the process to be stated in the sequel. A partial solution is a partial valid schedule which satisfies as well the constraints.

An overall schedule is $S = \{s_k \mid k = \overline{1, m}\}$, where all the operations performed on all the machines $k = \overline{1, m}$ are scheduled. The list of start times for the operations performed on every machine k, $k = \overline{1, m}$, specified by functions (2), can be determined through various methods to obtain either a semi-active schedule, or an active one or a non-delay schedule.

There are three *constraints*: a) the precedence constraint, b) the non-preemption constraint and c) the resource capacity constraint [19].

The *(F)JSSP solution* is the overall schedule, S, which consists in all the operations of all the jobs on the machines, ordered by the positive integer values of the functions s_k, $k = \overline{1, m}$. To this schedule a *performance measure*, $C_{\max}(S)$, is assigned to be minimised:

$$C_{\max}(S) = \max_{i \in J,\ j = \overline{1, n_i},\ k = \overline{1, m}} (s_k(o_{i,ni}) + \tau_{i,ni}) \tag{7}$$

The relation (7) computes the *makespan* as the maximum stopping time, considering all the operations in the schedule.

Besides the makespan, which is the main subject of interest of the scheduling system, secondary aspects can be taken into consideration, such as: total workload on the machines, maximum workload, jobs flow time, the amount of work-in-process, machines

unused capacity, the average idle ratio (regarding the jobs or the machines), the number of late operations etc. and are related to the scheduling optimization criteria.

The *objective* is to extremize a particular performance measure which indicates, from the point of view of temporal constraints, how well the scheduling is handled. More precisely, the primary objective consists in minimization of the makespan:

$$C^*_{\max}(S) = \min_{S \in \{ feasible\_schedules \}} (C_{\max}(S)). \quad (8)$$

A schedule with minimum makespan is named *optimal solution of the (F)JSSP*.

For the multi-objective FJSSP, the objective consists in the simultaneous optimization of many objectives, for example the output values mentioned before. Though the most natural (altough not simple) way to handle the multiple objectives is their aggregation into one single objective to consider further the mono-objective problem, the Pareto optimality approach, which separately considers the objectives, proves to be more beneficial [7].

MOFJSSP complexity

The JSSP is a NP-hard combinatorial problem, for which the classical approaches, based on exhaustive search, have a limited success. The only polynomially solvable instances are those mono-objective unconstrained without flexibility involving maximum two jobs and two machines. Instances with higher dimensions are even strongly NP-hard, for example some instances with two or three jobs with recirculation, as it is stated in [25]. If, additionally, one considers multiple objectives and supplementary parameters, the problem becomes even more difficult. The practice showed that the JSSP are very difficult to solve even heuristically. Once supplemented with the flexibility feature, JSSP become more complex. A broder analysis of the complexity results for the (F)JSSP offer [19] and [25].

3. Simulation-based Optimization for MOFJSSP

The research in the JSSP area in the last sixty years has shown several distinct phases. The first efforts were focused on the design of priority dispatch rules heuristics. A few examples of the many rules are: first-in-first-out, shortest processing time, shortest remaining processing time, time-insystem [15, 25, 8]. Every such rule sequences the jobs to be scheduled based the specific criterion and therefore a priority list for the jobs is created.

The experience showed that for the big complex instances the simple priority dispatch rules do not lead to an adequate success. Therefore, some research was made to combine simple rules in weighted dispatching criteria. Such a result is the algorithm designed by Filip [10] for real time production control, based on parametric decomposition [11, 12]. The algorithm is designed to use either simple priority dispatching rules or a composite weighted priority rule to help to decide which is the most appropriate job to be assigned to a vacant machine. The set of dispatching rules used includes: a) the minimum duration of the next operation, b) the Carrol's rule, c) the maximum time spent in queues, d) the maximum remaining processing time/current processing time, e) first in first out, f) the estimated size of the next queue and g) an external priority. The efficiency of each schedule obtained through simulation is evaluated by an aggregated utility function calculated as a weighted sum of several performance measures such as: a) estimated delay of completing the job, b) estimated time for an early delivery, c) the waiting time spent in queues, d) total idle time of machines. The value of the aggregated performance measure serves to adjust the values of importance parameters of the composite dispatching rule through a pseudo-Newton algorithm. The method proved to be beneficial for middle-size multi-objective JSSP. The dynamic adjustment of weight parameters of the dispatching rules based on fast simulation allowed also for real time decisions to be made in the case of „crisis" situations (urgent unexpected orders, machine failures and so on).

The dispatching rules have recently re-gained popularity in the context of dynamic scheduling for real-time assignment in supply chains [8].

For the small to middle-size JSSP, the exact optimization methods are also suitable. They comprise, on the one hand, the enumerative methods (such as backtracking, branch and bound and dynamic programming), and on the other the calculus-based techniques, the Lagrangean relaxation and the decomposing strategies [19]. Though the exact optimization

methods can identify the global optimal solution(s), the JSSP instances over 15 jobs on 15 machines exceed the solving power of these methods [20].

As opposed to the exact methods, the approximation methods are suitable to the larger and more complex instances of JSSP or MOFJSSP, though they do not guarantee the identification of the global optimal solution, if such a solution exists [6, 18, 19, 20, 25].

Shifting bottleneck heuristic, iterative local search heuristics and *metaheuristics* (such as genetic algorithms, tabu search, simulated annealing and GRASP) also received high theoretical and practical interest in scheduling area [11, 18, 19, 20, 25].

Among all such metods, the *genetic algorithms* (GA) have a low design cost, are easy to extend and to connect to the existing models and simulations, can be used in parallel processing and, the most important, they simultaneously operate on many candidate-solutions in a way that overcomes the issues in abrupt search spaces with many local optima. Thus, for certain complexity features of the big (MOF)JSSP, GA might provide the best performance.

In the last 20 years the research was focused on *agent-based techniques* (mainly consisting in simulation of natural social optimizer behaviour, such as Ant Colony Optimization, Wasp Behavior Model and Particle Swarm Optimization), *artificial neural networks*, *expert systems*, *knowledge based systems* and *fuzzy techniques* [19, 20, 25].

Also *hybrid techniques* prove for specific manufacturing contexts the ability to identify (near) optimal solutions in a reasonable amount of time [5]. A more ample discussion over the scheduling simulation-based optimization methods and the comparative analysis is made in [24].

The choice of appropriate representation for the schedules, the design for efficient search operators and the parameters tuning remain challenging issues for the most of the metaheuristics. The core interest should be given to the domain knowledge incorporation, to the constraints handling techniques, to the specialized operators and to the local search heuristics.

Genetic algorithms

A GA is a powerful weak adaptive optimization technique [16]. It is a weak method because a little information is necessary about the problem to use it, but in contrast with other weak methods, a GA exploits in a sophisticated manner this information with a relatively limited search effort, fact that gives to it the power feature.

The first study on GA applied to scheduling was made 25 years ago, by Davis [6]. Since then, the most of the theoretical and practical research works have been mainly focused on the mono-objective scheduling problems; the multi-objective flexible JSSP did not benefit of a similar attention.

GAs have recently proved to be effective in scheduling applications in supply chain management and control [8] and disassembling line balancing [9, 22].

4. NSGA-II-DAR Optimization Control Model for MOFJSSP

A direct consequence of applying No Free Lunch Theorem [26] in the GA case is that to search the best GA is senseless; instead of this, we can identify the best GA for a given instance and a given search space, from the NP-completeness perspective.

In this context, the GA under evaluation was designed for the difficult middle to high dimension MOFJSSPs and the search spaces formed by schedules encoded as permutations without repetitions of the operations to be scheduled set. The algorithm, NSGA-II-DAR (*NSGA-II with dynamical application of genetic operators and population partial re-initialization*) comprises an adaptive heuristic procedure embedded in the NSGA-II algorithm proposed by Deb et al. [7].

Genetic encoding and performance evaluation

For the candidate-solutions (individuals that are schedules in the GA), a genetic encoding as permutation without repetitions of the operations set is chosen:

$$Op = \bigcup_{i \in J,\, j=\overline{1,n_i}} \{(i, j)\} \tag{9}$$

Given this chromosome encoding, decoding a chromosome means to establish the scheduling sequence based on the start times of the operations in the chromosome. Start times are set according to the principle “to the first non-

processed operation in the schedule, the necessary resource is assigned once it becomes available". In other words, to the chromosome a semi-active decoding procedure is applied, where no operation can be started earlier without modifying the processing order or violating the technological constraints. The start time for an operation $o_{i,j}$ in chromosome is [2]:

$$s_k(o_{i,j}) = \begin{cases} \max(at_k, s_{k'}(o_{i,j-1}) + \tau_{i,j-1}), \\ \qquad \text{if } 1 \le j \le n_i \\ at_k, \text{if } j = 1 \end{cases} . \quad (10)$$

In the above relation, which is the expression of the precedence constraint and the capacity constraint for FJSSP, at_k is the availability time for the resource k which process the operation $o_{i,j}$ and $s_{k'}(o_{i,j-1}) + \tau_{i,j-1}$ is the end time of the preceding operation (in job i) on the corresponding resource k'.

Having set the start times for all the operations in the schedule, the performance assessment for the schedule (the fitness) is computed, in correlation with the objective function(s). According to the relation (7) in the FJSSP formulation and to the relation (10), the makespan of a schedule S is:

$$C_{\max}(S) = \max\{at_k \mid 1 \le k \le m\}. \quad (11)$$

Through the particular working methodology, a GA does not identify all the feasible schedules in the search space, but it evolves lower makespan schedules.

DAR control strategy

The performance of a GA, especially when applied to the multi-objective optimization, is preponderantly determined by the level of balance between population diversification and searching intensification. The better this balance maintained, the more efficient is the ability to avoid the premature convergence to local optima regions of the search space.

The aim of the proposed strategy, named DAR, is to avoid the premature convergence of the genetic algorithm. This is achieved through two mechanisms:

- dynamical application of crossover and mutation operators (DA) and
- population partial re-initialization (R).

Both of them are based on the average progress of the genetic operators during the evolution.

The first mechanism of the adaptive DAR control strategy dynamically selects in each generation of GA a crossover operator and a mutation operator to be applied located in two sets of operators.

We used two crossover operators: UX (Uniform Order-Based Crossover) and PPX (Precedence Preservative Crossover) and three mutation operators: frame-shift, translocation and inversion. To note that any number of operators one uses, the procedure remains valid and applies with a relative similar complexity.

The two operators that score the best average progress from the beginning of evolution are selected in the current generation, so that the overall evolution is reached by taking advantage of the best operators adequate to the considered instance. The progress of each operator determines a certain selection probability for the operator. With every application of an operator, its progress updates and consequently its selection probability is updated as well.

In [1] a dynamical application of two mutation operators is proposed and tested on a bi-objective flow shop scheduling problem and return good results.

In the proposed approach a modified progress assignment for the mutation operators is used, which better takes into account the Pareto dominance relation and the crossover as well is dynamically applied. The formula for the crossover progress assignment accentuates the distinction between the offspring which dominates both parents and the offspring which dominate only one parent or none.

Let S be the offspring resulted in generation t from the parents (S_1, S_2) by applying the crossover operator x. The progress of x is [24]:

progress(x)= (12)

- 1, if S dominates $S1$ and $S2$
- $\max(1 - \frac{k_1 * t}{G}, 0.5)$, if S dominates only a parent
- 0.5, if no dominance relation exists between S and $S1$, between S and $S2$

- $\max(0.5-\frac{k_2*t}{G},0)$, if S is dominated only by a parent and no dominance relation exists with the other
- 0, otherwise (if S is dominated by both parents)

where G is the maximum number of generations of the evolution, and $k1$, $k2$ are the parameters which enforce the velocity of reducing the progress during the evolution. This reduction leads to a broader proliferation of the poor individuals in the first generations than in the latter ones, when the search follows a more stable direction. Moreover, giving chances to proliferate in the population to the offspring that dominate no parent but are dominated by both parents, the population becomes more diverse.

In order to make the value $(1.0-\frac{k1*t}{G})$ belong to the segment [0.5, 1] and $(0.5-\frac{k2*t}{G})$ to [0, 0.5], the variation range for parameters $k1$ and $k2$ is bounded by 1 and $G/2$. Setting and tuning the values of $k1$ and $k2$ are made from two perspectives: a) the extent and b) the quality of the search space exploration. Hence, to get an extensive exploration (therefore a very diverse population), both parameters are set to low values (down to 0.5). The maximum exploration is attended when $k1 = k2 = 0.5$; this is the most permissive way for the poor individuals to be accepted in the evolution. Such behaviour for the genetic strategy is desirable for the very difficult instances, with abrupt search space, when one should avoid loosing the hidden solutions. An extensive exploration is not convenient however to any instance, because the population diversification and searching intensification balance may be broken.

When $k1$ and $k2$ are set to the maximum value, $G/2$, the poor individuals are the most restricted to be accepted in the population and the effects are reverse than in the previous case. It is advisable that $k1$ value to not exceed the $k2$ value, because no one will accept to lose some offspring which dominate one parent and to keep some offspring that dominate no parent.

Let x be a mutation operator applied to an individual S. The progress of x is [24]:

$$progress(x)= \quad (13)$$

- 1, if $mut(S)$ dominates S
- $\max(1-\frac{k_3*t}{G},0.5)$, if no dominance relation exists between $mut(S)$ and S
- 0, if $mut(S)$ is dominated by S

where $mut(S)$ is the candidate-solution obtained after x applies to S and $k3$ is the parameter which enforces the velocity of reducing the progress during the evolution. The second line in the above relation distinguishes this formula from the one proposed in [1].

The comments about $k1$ and $k2$ are also valid for $k3$. A low value of $k3$ leads to a vast exploration of the search space by mutation because individuals in the same Pareto front with S are allowed to enter the population.

The selection for crossover and mutation operators to be applied in each stage of the algorithm is called by the DAR control strategy based on the selection probabilities of all the nr available operators in x's class [1]:

$$\pi_x = \frac{\overline{progress(x)}}{\sum_{l=1}^{nr}\overline{progress(l)}}*(1-nr*\alpha_x)+\alpha_x \quad (14)$$

Here, $\overline{progress(l)}$ is the average progress per application of the operator l ($l = 1,..,nr$) and α_x is the minimum value of the selection probability of x. The value $\alpha_x \in (0,1)$ allows one to use all the crossover and mutation operators during the evolution, even if some of them prove to be weak for the concerned instance.

The relation (14) suggests that when the progress of an operator is high, the decision strategy provides a control to more frequently apply the operator.

In the first generation, the selection of crossover and mutation operator is randomly performed, with equal probabilities: for the crossover this probability is 0.5 and for the mutation operator is 0.33.

For the selected operator, if the result is feasible, the average progress per application is computed using [1]:

$$\overline{progress(x)} = \frac{\sum_{i=1}^{nra} progress_i(x)}{nra}, \quad (15)$$

where nra is the number of times the operator x was applied until the current generation and $progres_i(x)$ is the progress of x at application step i, computed according to the relation (12) and (13).

If the result remains unfeasible in ten trials, another operator in his class is selected to be applied and for this latter operator the progress is updated. We note that PPX and frame-shift operators do guarantee the result is feasible. For that we used the improved variant of frame-shift in [23].

As opposed to the DA (dynamical application of crossover and mutation operators) mechanism, the R mechanism, namely population partial re-initialization, is applied only when the risk to premature convergence in the current generation is considered high enough. It is considered this condition met when the average progresses of all the operators do not exceed a minimum threshold, $pMin \in [0,1]$.

The re-initialization consists in replacing a part of the new generated population with some randomly generated individuals, in a proportion set by the *pReinit* parameter.

Additionally, if the re-initialization is performed 300 times in a run, the evolution will be stopped anyway, though the a priori set stop criteria are not satisfied. The reason for this is that many re-initializations means poor chances there are to identify better solutions in the future.

The two proposed mechanisms combine their effects in the DAR strategy through multiple direct and indirect interdependences of the associated parameters (as figure 1 shows).

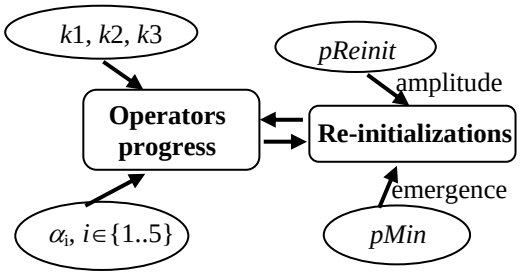

Figure 1. The interdependences between the DAR strategy parameters

The strategy achieves a dynamical adaptive parameter control from a run to another: the feedback from the current search state determines the direction and the magnitude of the genetic alteration.

Through DAR strategy, the GA permanently adapts to the population performance in order to identify the favourable regions in the search space. Hence, the algorithm is able to learn during the evolution what are the appropriate genetic operators for a particular instance, to promote the beneficial results of all the available operators, to maintain a good balance between the exploration and the exploitation of the search space and to extend the genetic search without loosing the direction, fact that avoid or delay the premature convergence of the algorithm.

NSGA-II-DAR algorithm

The NSGA-II algorithm is a GA proposed in [7] which proved to be more effective than other GAs especially for multi-objective complex optimization problems, with conflict objectives, as MOFJSSP is.

The NSGA-II algorithm contains a fast procedure for Pareto dominance based *sort* (which returns the list of non-dominated fronts in the population). Also an estimate of the density of the individuals around a particular individual named *crowding distance* is made. This measure is used to define a comparing operator, designated as "$\geq_n$", in order to identify the individuals in a Pareto front that are located in less dense regions. The reason for this stands in building a bias to uniformly distributed optimal Pareto fronts.

Over the NSGA-II algorithm the DAR strategy is inserted to attain NSGA-II-DAR algorithm. In the first generation $t = 0$ the population P_t is pseudo-random initialized then fast sorted in order to assign to each individual the dominance level (in fact the front index where it is part). By applying the genetic operators (binary tournament selection, crossover and mutation) a new population Q_t with N elements is obtained. The elitist procedure for a generation t, except the initial one, allows the parent solutions to be compared with offspring and a combined population R_t is created, having $2N$ elements. The fast sorting is applied to R_t and the list F of the non-dominated fronts is generated. The new parent population P_{t+1} is formed by adding solutions in the best fronts until the dimension N is reached. For each individual in these fronts, the crowding distance is computed and the individuals in the last accepted front are sorted based on "$\geq_n$" relation in order to select the best ones to complete the population P_{t+1}. On this parent population the genetic operators are further applied and the new offspring population Q_t is obtained. If the risk for premature convergence is noticed at the current generation, Q_{t+1} is partially re-initialized according to the mechanism R of the DAR strategy. This step, along with the new population formation step

which supposes the dynamical application of the crossover and mutation, represent the difference between NSGA-II-DAR and NSGA-II.

The solutions of the algorithm are the individuals in the first front in population P_t at the last generation (all the non-dominated schedules). To choose a final unique solution, one can use *user preferences*. Other output values of the control algorithm are: a) the number of different solutions, b) the number of performed re-initializations, c) the applying frequencies of genetic operators, and d) the average progress of the genetic operators in the last generation (in order to identify the appropriate operators for the instance).

In the current version, the computer simulation application implements four genetic algorithms: NSGA-II, NSGA-II-DAR, an elitist GA and the canonical GA.

5. Simulation Results

The NSGA-II-DAR performance evaluation was made in multiple tests on a difficult FJSSP instance from pharmaceutical industry, called *Pharm*, both mono-objective and multi-objective and the notorious *ft*10 JSSP test-instance [13].

In the *Pharm* instance, for a scheduling horizon of one month, 79 jobs of 16 different types

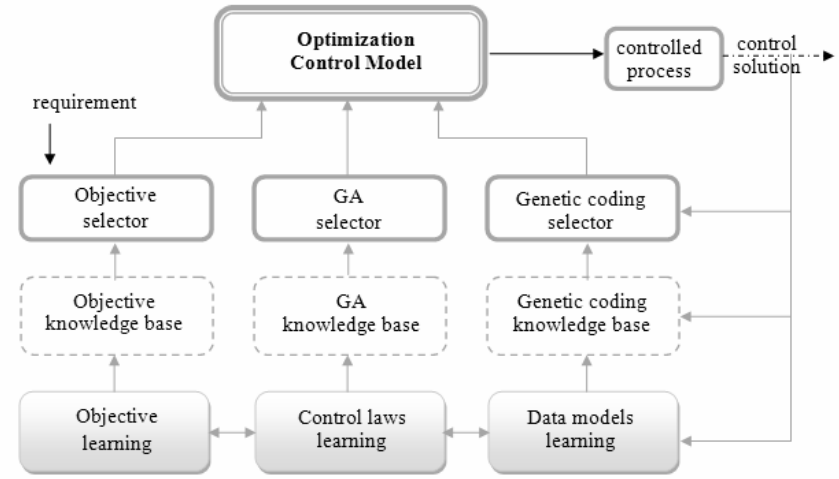

Figure 2. Autonomous system architecture specific to the designed Java application.

Computer Simulation Application

To test NSGA-II-DAR algorithm a Java application was developed. It was designed to allow the control for any optimization process, not only scheduling. It comprises modules for providing many genetic encodings and modules for implementation any number of optimization criteria. By the generality feature, the application constitutes an *autonomous intelligent control system*, whose architecture integrates many selector blocks (see figure 2), which confers to it self-organizing attributes. Hence, the application is flexible related to the variety and the complexity of the instances.

The selector blocks offers, on the one hand, a high level of control flexibility, it was noted before, and, on the other one, constitutes a good support for identification of the most appropriate encoding and the most appropriate GA for a specific instance.

have to be scheduled on 20 machines. The input data in table 1 indicate a total number of 606 operations. Here, *Prod* means the type of product, *NJ* is number of jobs corresponding to every type of product and *NOP* is the number of operations in the job. The associated scheduling flexibility is of type II.

A feasible schedule for the 606 operations may be the sequence

(6,1)(76,1)(53,1) ... (69,3) ... (54,8)(60,8)

ordered by the start times associated to the operations. In the pair (*i*,*j*) *i* is a job in *J* and *j* is an operation in the job *i*, according to the genetic encoding specified by relation (9).

The measure unit for the makespan of a schedule is the eight-hour shift. Consequently, in relation (11), the value $C_{max}(S)$, which is expressed in minutes, is divided to 480 (8 hours x 60 minutes).

Table 1. Input data for the *Pharm* instance

Prod	*NJ*	*NOP*	**Routings of the jobs on the machines**									
			Machine / machines									
			Processing times (minutes)									
1	8	6	**1**	**2**	**7**	**11,13**	**18**	**20**				
			5	10	476	200,167	800	113				
2	1	10	**1**	**2**	**3**	**4**	**8**	**9,10**	**14**	**15**	**17**	**20**
			5	15	20	30	320	685,342	394	137	253	120
3	13	8	**1**	**2**	**5,6**	**9,10**	**14**	**15**	**17**	**20**		
			5	20	150,206	325,313	684	214	341	188		
4	4	7	**1**	**2**	**5,7**	**12**	**18**	**16**	**20**			
			5	20	120,222	133	500	300	75			
5	3	8	**1**	**2**	**7**	**9,10**	**14**	**15**	**17**	**20**		
			5	20	315	263,225	560	150	239	132		
6	1	7	**1**	**2**	**5,6**	**12**	**19**	**16**	**20**			
			5	10	83,105	67	250	84	38			
7	19	7	**1**	**2**	**5,6**	**12**	**18**	**16**	**20**			
			5	10	83,105	67	108	84	38			
8	1	9	**1**	**2**	**3**	**4**	**5,7**	**13**	**18**	**16**	**20**	
			5	10	10	20	120,159	200	400	150	38	
9	3	6	**1**	**2**	**7**	**11,13**	**18**	**20**				
			5	10	360	286,222	800	150				
10	3	8	**1**	**2**	**3**	**4**	**7**	**11,13**	**18**	**20**		
			5	10	355	120	65	357,278	700	188		
11	4	10	**1**	**2**	**3**	**4**	**8**	**9,10**	**14**	**15**	**17**	**20**
			5	20	45	30	554	605,510	567	172	316	150
12	2	8	**1**	**2**	**3**	**4**	**5**	**11**	**18**	**20**		
			5	10	43	38	180	230	600	113		
13	11	9	**1**	**2**	**3**	**4**	**5**	**11**	**18**	**16**	**20**	
			5	10	28	46	280	230	290	150	113	
14	3	6	**1**	**2**	**5,6**	**11**	**18**	**20**				
			5	10	424,457	366	970	375				
15	2	10	**1**	**2**	**3**	**4**	**5**	**9,10**	**14**	**15**	**17**	**20**
			5	15	32	31	60	375,188	450	136	173	113
16	1	8	**1**	**2**	**5**	**9,10**	**14**	**15**	**17**	**20**		
			8	15	120	425,363	305	272	346	225		

In accordance with the human experience only, the minimum makespan corresponding to the given instance input data is 53 shifts.

For the multi-objective case, when three objectives are considered, the original makespan minimization objective is supplemented with the following two objectives:

- minimizing the number of late operations compared to 44 shifts value;
- minimizing the average ratio of idle times in the workshop, computed as it follows (for a given schedule *S*):

$$R_{idle}(S) = \frac{\sum_{i=1}^{ns} \frac{dreal_i - dvirtual_i}{dvirtual_i}}{ns}, \qquad (16)$$

where *ns* is the number of jobs to be scheduled, $dreal_i$ is the time spent by a job, *i*, in the workshop (calculated as the difference between ending time of the last operation of the job and input time of the first operation) and $dvirtual_i$ is the processing effective duration of job *i* (calculated as a sum of processing times of all operations in the job).

The three objectives that are to be simultaneously minimized are handled through Pareto dominance relation.

The search space of the *Pharm* instance is enormous (its dimension is of order $26*10^{388}$) and satisfies all the five difficulty criteria for the JSSP mentioned in [18] even in the case a single objective is pursued.

In the *mono-objective case*, when one tries to find the optimal schedule so that the makespan is minimized, 50 tests were run with 17 different sets of parameters values, which are combinations of values of the table 2. Here *N* is the population dimension, *G* is the maximal number of generations, r_m is the mutation rate,

pMin, *pReinit*, (*k1*,*k2*,*k3*) and (α_1, α_2, α_3, α_4, α_5) are the specific parameters to the DAR strategy.

Table 2. Parameters values for the NSGA-II-DAR

Parameter		Values	
		mono-objective	**multi-objective**
N		300	300
G		500	500
r_m		0.01, 0.05, 0.08, 0.10	0.01, 0.03, 0.05
(*k*1, *k*2)	(*k*1, *k*2, *k*3)	(100, 200) (30, 50) (4, 10) (0.5, 1)	(150,220,200) (100,200,100) (4,10,5)
pMin		0.05, 0.2, 0.5, 0.52, 0.6, 0.7	0.3, 0.5
pReinit		10%, 20%, 50%	10%, 20%, 50%
(α1, α2, α3, α4, α5)		(0.2, 0.3, 0.05, 0.05, 0.05) (0.5, 0.8, 0.05, 0.01, 0.01) (0.5, 0.8, 0.1, 0.1, 0.1) (0.8, 0.8, 0.1, 0.1, 0.1)	(0.2, 0.3, 0.05, 0.05, 0.05) (0.5, 0.8, 0.05, 0.01, 0.01)

For the *multi-objective case*, six tests were run (with the parameters values of table 2) in order to find a Pareto optimal schedule to minimize the three objectives above mentioned.

The results obtained by the proposed algorithm in contrast with the results of other three genetic algorithms (NSGA-II, an elitist GA, and the canonical GA, respectively) are analyzed with respect to several performance measures, as shown in table 3.

test) and *DivOb* is the diversity in the objective space (measured as the variance of objective values). For the multi-objective case, where *aggr.* is specified, table 3 reports the aggregation values obtained with coefficients for the fixed set of weight values of objectives importance {0.5,0.1,0.4}. For the best performance is also reported the minimum makespan value of the best solutions.

The NSGA-II-DAR algorithm obtained a makespan with 9.45% lower than in the human judgement-based methodology, meaning 40 hours in a month. Also a high rate of good solutions in both cases (mono and multi-objective) is obtained, well-distributed and diverse in the schedules space and in the objective space. The genetic operators appropriate to the instance are the inversion and PPX in the mono-objective case and the inversion and UX in the multi-objective case. Along with the frame-shift, these proved to be the most robust operators.

The comparative analysis of the results indicates the superiority of the proposed algorithm against the other algorithms in the mono-objective case from all the considered perspectives: the solutions performance, diversity in the schedule space and the objective space, the best solutions distribution in the objective space. In the multi-objective case, NSGA-II-DAR obtained a better performance compared to the elitist GA and

Table 3. The comparative performance measures values of the four GA for the *Pharm* instance

Algorithm / **Measure**	NSGA-II-DAR	NSGA-II	Elitist GA	Canonic GA		NSGA-II-DAR	NSGA-II	Elitist GA	Canonic GA
	mono-objective case				multi-objective case				
BP	**47.99**	48.16	48.56	58.25	min mksp	48.10	**48.08**	48.24	56.40
					aggr.	29.66	**29.29**	29.97	43.76
AvP	**48.48**	48.64	49.02	60.37	aggr.	31.15	29.57	**29.30**	51.16
WP	**48.99**	49.23	50.05	68.27	aggr.	34.08	**29.92**	**29.92**	58.00
VRM	**1**	1.7	1.49	10.02		3.7	3.94	**3.55**	5.89
Distr	**27%**	24%	23%	10%	mksp	**78%**	62%	64%	39%
					aggr.	15%	28%	**32%**	12%
DivSch	**235.04**	184.20	67.40	1.40		**15.66**	1.2	1.90	1.00
DivObj	**0.33**	0.40	0.48	3.29	aggr.	0.70	**0.23**	0.27	5.06

Here, *BP* is the best performance (makespan), *AvP* is the average performance, *WP* is the worst performance, *VRM* is the makespan variation range (in shifts), *Distr* is the best solutions distribution in the objective space, *DivSch* is the diversity in the schedule space (measured in number of different solutions per

canonical GA, and a pretty similar performance with NSGA-II. Notice that the maximum diversity in the schedules space and the best distribution of the best solutions is provided by the NSGA-II ADR.

The other three algorithms used for the comparative analysis were run with UX crossover, frame-shift mutation, N = 300, G = 500 and r_m = 0.01.

The superiority of the NSGA-II-DAR against the other three algorithms for the *ft*10 instance is shown in table 4. For this instance, 41 tests were run with 14 different sets of parameter values, where $N \in [20,10000]$ and $G \in [20,500]$.

Based on the above mentioned asserts, one can conclude that the proposed control strategy is able to avoid the premature convergence of the genetic algorithm to the suboptimal regions, therefore achieving the purpose it was created for.

Table 4. The comparative results of performance measures for the *ft*10 test-instance

Alg. / Measure	NSGA-II-DAR	NSGA-II	Elitist GA	Canonic GA
BP	**1013**	1216	1054	1342
AvP	**1102**	1265	1213	1384
WP	**1306**	1345	1355	1553
VRM	293	**129**	301	211

Additionally, the control model proves to be viable from the point of view of performance criteria of the GA, namely: coverage and diversity of the search space and the objective search, complexity, quality of the solutions and convergence of the algorithm.

6. Conclusions

The investigation over the main simulation-based optimization methods applied to the MOFJSSP led to designing the NSGA-II-DAR control algorithm. This optimization scheme is able to cope with scheduling flexibility and multiple objectives. It is scalable, generally applicable and inherits the advantages of the genetic algorithms, which is based on, namely [17]: it is easy to construct and extend, has a global perspective over the search space, simultaneously operates with many candidate-solutions and provides many final solutions which do not dominates each other.

The proposed algorithm reinforces the well-known NSGA_II algorithm with an heuristic adaptive control strategy, named DAR, which is apparently efficient for any optimization problem. This strategy avoids the premature convergence of the algorithm to suboptimal regions and is able to learn the beneficial operational condition over a particular optimization instance, as the experimental results showed. This task is managed by two mechanisms: a) the dynamical application of many genetic crossover and mutation operators, which accomplishes the purpose during all the evolution and b) population partial re-initialization which comes to an effect only when the risk for premature convergence occurs. The adaptation criterion for the DAR strategy is based on the average progress of the genetic operators for that assignment formulas are proposed.

The comparative analysis of the results obtained by the NSGA_II-DAR algorithm was made against the results of other three genetic algorithms for a difficult big MOFJSSP in the pharmaceutical industry and for the *ft*10 test-instance. The results show the superiority of NSGA-II-DAR over the other algorithms.

Beyond the specifics of the scheduling framework, the proposed algorithm can be viewed as a general solver applicable to any optimization problem for which one can find an appropriate genetic encoding and to use any sequential GA.

REFERENCES

1. BASSEUR, K.M., F. SEYNHAEVE, E. TALBI, **Design of Multi-objective Evolutionary Algorithms: Application to the Flow-shop Scheduling Problem**, Proc. of the 2002 Congress on Evolutionary Computation (CEC), Hawaii, IEEE Press, vol. 2, 2002, pp. 1151-1156.
2. BIERWIRTH, C., D. C. MATTFELD, **Production Scheduling and Rescheduling with Genetic Algorithms**, Evolutionary Computation 7(1), 1999, pp. 1-17.
3. BRUCKER, P., R. SCHLIE, **Job-shop Scheduling with Multi-purpose Machines**, Computing 45(4), 1990, pp. 369-375.
4. CHAN, F. T. S., T. C. WONG, L. Y. CHAN, **Flexible Job-Shop Scheduling Problem under Resource Constraints**, Intl. J. of Production Research 44(11), 2006, pp. 2071-2089.
5. CHENG, R. W, M. GEN, Y. TSUJIMURA, **A Tutorial Survey of Job Shop Scheduling Problems using Genetic Algorithms: Part II: Hybrid Genetic Search Strategies**, Intl. J. of Computers

and Industrial Engineering 36, 1999, pp.343-364.

6. DAVIS, L., **Job Shop Scheduling with Genetic Algorithms**, Proc. of the Intl. Conference on Genetic Algorithms and their Applications, San Mateo, Morgan Kaufmann, 1985, pp. 136-149.
7. DEB, K., S. AGRAWAL, A. PRATAP, T. MEYARIVAN, **A Fast and Elitist Multi-objective Genetic Algorithm for Multi-objective Optimization: NSGA-II**, Proc. of VI-th Parallel Problem Solving from Nature Conference, Paris, 2000, pp. 849-858.
8. DOLGUI, A., J.-M. PROTH, **Supply Chain Engineering: Useful Methods and Techniques**, Springer-Verlag, London, 2010.
9. DUȚĂ, L., F. G. FILIP, J. M. HENRIOUD, C. POPESCU, **Disassembly Line Scheduling with Genetic Algorithms**, Int. J. of Computer Communication and Control, 3(3) , 2008, pp. 270-280
10. FILIP, F. G., **Contributions to Hierarchical Control of Complex Systems**, Ph.D. Thesis, Polytechnic Institute of Bucharest, Romania, 1981 (in Romanian).
11. FILIP, F. G., **Decizie Asistată de Calculator: Decizii, Decidenți - Metode de Bază şi Instrumente Informatice Asociate** ("Computer Aided Decision making; Methods and Associated Information Tools"). second edition, Ed. Tehnică, Bucureşti, 2005 (in Romanian)
12. FILIP, F. G., G. NEAGU, D. A. DONCIULESCU, **Job Shop Scheduling Optimization in Real-time Production Control**, Computers in Industry 4, Elsevier, 1983, pp. 395-403.
13. FISHER, H., G. L. THOMPSON, **Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules**, Industrial Scheduling, J. F. Muth & G. L. Thompson (Eds.), Prentice-Hall, Englewood Cliffs, NJ. French, 1963, pp. 225-251.
14. FRENCH, S., **Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop**, Chichester, England, Ellis Horwood Ltd., 1982.
15. GIFFLER, B., G. L. THOMPSON, **Algorithms for Solving Production Scheduling Problems**, Operations Research 8(4), 1960, pp. 487-503.
16. GREFENSTETTE, J. J., **Incorporating Problem Specific Knowledge into Genetic Algorithms**, in L. Davis (Ed.) Genetic Algorithms and Simulated Annealing, Morgan Kaufmann , 1987, pp. 42-60.
17. HOLLAND, J. H., **Genetic Algorithms**, Scientific American 267(1), 1992, pp. 44-50.
18. JAIN, A. S., S. MEERAN, **Deterministic Job Shop Scheduling: Past, Present and Future**, European Journal of Operational Research 113(2), 1998.
19. JAIN, A. S., S. MEERAN, **A State-of-the-Art Review of Job-Shop Scheduling Techniques**, European Journal of Operations Research 113, 1999, pp. 390-434.
20. JONES, A., L. C. RABELO, **Survey of Job-Shop Scheduling Techniques**, M. Sc. dissertation, NISTIR, Gaithersburg, MSID Publications, 1998.
21. KAUFMANN, M., **Methods and Models of Operations Research**, vol. II, Ed. Ştiințifică şi Enciclopedică, Bucureşti, 1975 (in Romanian).
22. MCGOVERN, S. M., S. M. GUPTA, **The Disassembly Line: Balancing and Modeling**, McGraw-Hill, New York, 2011.
23. NICOARĂ, E. S., **Mechanisms to Avoid the Premature Convergence of Genetic Algorithms**, Petroleum – Gas University of Ploieşti Bulletin, Math. – Info. – Phys. Series, vol. LXI, 1/2009, pp.87-96.
24. NICOARĂ, E. S, **GA-based Control of Multi-objective Flexible Job Shop Scheduling Processes**, Ph.D. Thesis, Petroleum-Gas University of Ploieşti, 2011 (in Romanian).
25. PINEDO, M. L., **Scheduling. Theory, Algorithms, and Systems**, 3rd ed., Springer, New York, 2008
26. WOLPERT, D. H., W. G. MACREADY, **No Free Lunch Theorems for Optimization**, IEEE Trans. on Evolutionary Computation 1(1), 1997, pp. 67-82.