Genetic multi-criteria approach to flexible line scheduling

M.P. Fanti, B. Maione *, D. Naso, B. Turchiano

Politecnico di Bari Dipartimento di Elettrotecnica ed Elettronica Via Re David 200, 70125 Bari, Italy
Received 1 March 1997; accepted 1 November 1997

Abstract

Job scheduling in flexible production systems is a complex task even for simple cases. This paper deals with this problem using fuzzy set theory and genetic algorithms. Fuzzy techniques allow us to define global performance measures expressing different and often conflicting objectives of the production. Moreover, according to a fuzzy multi-criteria algorithm, we propose a methodology to combine various heuristics, with different weights, in a single dispatching criterion. A genetic optimization process selects the weights guaranteeing good system performances. A case study and some extensive simulations show the efficiency of the methodology. © 1998 Elsevier Science Inc. All rights reserved.

Keywords: Flexible flow lines; Fuzzy sets; Genetic algorithms; Multi-criteria; Decision making

1. Introduction

Scheduling methodologies play a key role in managing modern automated manufacturing systems, since they allow optimization of both hardware (numerical control machines, material handling devices, etc.) and software (information flow, data base content, etc.). On the other hand poor scheduling policies decrease resource utilization and increase costs, production times and inventory. According to the classical point of view, combinatorial optimi-
zation techniques are the main tools for dealing with scheduling issues. However, due to their intrinsic complexity, such optimization problems have not analytical solutions except for very simple cases [6]. Therefore, the effectiveness of classical scheduling theory in most production environment is minimal [15].

Real industrial plants require high flexibility and fast response time to face the changing market demand and the pressure of competition [14]. Thus, scheduling must have dynamic features, i.e. the policies may be rapidly varied according to changing production objectives and system conditions. For this reason scheduling procedures that tend to be time-consuming (such as many optimum seeking procedures suggested for job shops) are not appropriate.

Furthermore, constraints in combinatorial optimization often result from the need of tractability in mathematical formulation rather than from physical limits of the plant. On the other hand, production goals are imprecisely specified and conflicting with each other. Hence conventional scheduling rules usually do not yield satisfactory results but multiple objective optimization approaches appear more appropriate. In conclusion, the weakness of classical scheduling methodologies calls for new approaches to the control of automated manufacturing systems.

In recent years, among many emerging methodologies, Fuzzy Logic (FL) has shown some interesting potentiality in different aspects of the scheduling problem in flexible production floors [24]. FL has been successfully introduced as means of constraint relaxation, for determining approximate solutions and compromises and for modeling human decision-making. Referring only to papers strictly related to our approach, Hintz and Zimmermann [14] use fuzzy linear programming, taking into account the uncertainties of the planning situation, to determine the master schedule of the plant and fuzzy rules with linguistic variables as an evolution of priority rules for loading and dispatching parts in the system. In [21], Slany describes an approach combining a fuzzy multiple-constraint satisfaction algorithm with a heuristic repair method for scheduling a steel-making plant. Developing a heuristic hierarchical approach based on fuzzy logic, in [7] Custodio et al. introduce a modified version of Yager's formulation for resource allocation in the operation control (lower level) and use a fuzzy rule base for short range production planning (middle level). In [2], Angsana and Passino develop a distributed control system with performance comparable to those of conventional schedulers and with self-tuning capabilities. Finally, referring to job-shops, Roy and Zhang [18] propose another fuzzy dynamic scheduling algorithm that combines different rules through fuzzy sets and operators.

Utilizing fuzzy set theory and genetic optimization, this paper follows two main directions. The first one is the definition of a multiple performance measure expressing conflicting and not precisely defined requirements. In our approach, each single production objective has an associated performance measure (e.g. Mean Flow Time of jobs, Mean Tardiness of jobs, Makespan or Resource Utilization) and a predefined weight that quantifies its importance
in the global performance measure. In particular, we define suitable membership functions to convert the single performance measures into fuzzy values and to compute the global index.

The second direction is the synthesis of a multi-criteria control technique for the job sequencing. Notoriously heuristics are effective means to solve scheduling problems in complex manufacturing environments, where exact analytical optimization methods fail. However, since no single rule is dominant across all possible solutions, it is important to develop scheduling criteria utilizing many appropriately weighted decision rules. Following Yager's approach [23], we combine a set of heuristic rules in a single dispatching criterion, according to a simple multi-criteria decision-making algorithm. Then, we grade the incidence of each rule in the overall criterion by selecting rule weights appropriately. In this way, we improve the system behavior by adapting the multi-criteria dispatching rule to any given production system. On the other hand, the influence of each heuristic on the production system performance is generally unpredictable. So, we conduct the search of suitable weights by coding the rule weights into binary strings and using them as chromosomes in a genetic optimization process. We achieve the solution by comparing simulation runs of the given production process and by using the above mentioned multi-objective performance measure as fitness function.

Our approach appears particularly adequate to face scheduling problem in complex production environments. In fact, the fuzzy multi-heuristic policy has a very low complexity, comparable to that of the component dispatching rules, widely adopted in manufacturing practice. Moreover, while dispatching rules have a fixed behavior, our strategy incorporates a versatile and evolve method that considers multi-faceted production objectives for adapting the scheduling policy to different operating contexts.

To show the potentiality of the proposed approach, we consider two flexible production lines and compare the multi-criteria policy with classical dispatching rules. In our test cases, the optimized combination of rules leads to higher values of almost all the performance measures used in the comparison, including our fuzzy multi-objective index.

The organization of this paper is as follows. Section 2 describes the multi-criteria approach and Section 3 applies the method to set service priorities of jobs queuing at each machine in the system. Section 4 introduces the fuzzy multiple performance measure. Section 5 describes the Genetic Algorithm (GA) for the optimization of the scheduling policy and Section 6 shows the results of simulation experiments. Final remarks are drawn in Section 7.

2. The multi-criteria algorithm

Our scheduling approach is founded on a simple multi-criteria algorithm introduced by Yager [23] and briefly described in the following.
A multi-criteria decision problem consists in choosing an alternative \( A_i \), from the set of all possible alternatives \( \mathcal{A} = \{ A_1, A_2, \ldots, A_q \} \), that maximizes the satisfaction of a set of criteria or decision rules \( \mathcal{R} = \{ R_1, R_2, \ldots, R_r \} \). If \( a_{ij} \) is a measure of how much the alternative \( A_i \) satisfies the criterion \( R_j \) and \( 0 \leq a_{ij} \leq 1 \) for all \( i \) and \( j \), then the set of satisfaction degrees for a given criterion \( R_j \) can be considered a single fuzzy set \( \bar{R}_j \) with membership function:

\[
\mu_{\bar{R}_j}(A_i) = a_{ij}.
\]  

We can define a global criterion \( D \) by combining all the criteria in \( \mathcal{R} \):

\[
D = R_1 \text{ AND } R_2 \text{ AND } \ldots \text{ AND } R_r
\]

that can also be viewed as a fuzzy set \( \bar{D} \):

\[
\bar{D} = \bar{R}_1 \cap \bar{R}_2 \cap \ldots \cap \bar{R}_r
\]

with the membership function:

\[
\mu_{\bar{D}}(A_i) = \min_{1 \leq j \leq r} [\mu_{\bar{R}_j}(A_i)].
\]

At this point, we select as final choice the alternative \( A^* \) with the highest satisfaction degree in the global criterion \( D \), i.e.

\[
\mu_{\bar{D}}(A^*) = \max_{1 \leq i \leq q} [\mu_{\bar{D}}(A_i)].
\]

To stress different degrees of importance for the criteria constituting \( D \) we use a weight scale in the decision-making process. According to Yager [23], exponential weights on the membership grades represent an effective way to build a scale of importance. In fact, the values \( \mu_{\bar{R}_j}(A_i) \) decrease (increase) if raised to a power \( w_j > 1 \) (\( w_j < 1 \)). Since the membership function of the global criterion \( D \) is defined as the minimum of the membership functions of all criteria, exponential weights greater (smaller) than one increase (decrease) the influence of that criterion. In particular, exponent \( w_j = 0 \) makes the decision insensitive to criterion \( R_j \), because its membership function is equal to one for each alternative.

3. The dispatching criterion

Consider a set of \( n \) job types \( \mathcal{J} = \{ J_1, J_2, \ldots, J_n \} \) receiving service on a set of machines \( \mathcal{M} = \{ M_1, M_2, \ldots, M_m \} \). A job type \( J_j \) can be viewed as a set of \( h \) operations \( J_j = \{ o_{1j}, o_{2j}, \ldots, o_{hj} \} \) to be performed on a given subset of \( \mathcal{M} \). In the case of flexible lines the operation sequence is fixed and cannot be modified during the production. With each \( J_j \) we associate a set of parameters, as the processing time of \( i \)th operation \( t_{ij} \), the release date \( r_j \) (time at which the job is available for the first operation) and the due date \( d_j \) (time at which the job must be completed). Associated with each machine \( M_i \) there is a finite capacity
buffer $b_i$ hosting jobs in queue for operation on that machine. The task of the scheduler is to set the processing priorities in each queue of jobs, i.e. to establish the order of processing parts on each machine $M_i$, on the basis of the parameters associated with each job in the queue.

A scheduler for a job shop can be determined in various ways. Queuing networks generally do not have sufficient modeling capability to address detailed scheduling problems. So far, such models have been used to address Flexible Manufacturing System (FMS) design problems quantitatively and FMS planning problems qualitatively [17]. On the other hand, optimal and enumerative methods often clash with the combinatorial complexity of real cases. Consequently, heuristic methods still represent the most adopted class of tools to face this problem. Dispatching rules [4,13,16] as Shortest Processing Time (SPT), Earliest Due Date (EDD), Largest Number in Queue (LNQ) or First Come First Served (FCFS), are the most used heuristics in scheduling problems because they provide a good trade-off between simplicity and effectiveness of the solution procedure. Table 1 shows frequently used dispatching rules, while the aforementioned papers yield more complete lists. However, the performances of heuristic rules are highly system-dependent. Namely, each system deserves an individual and detailed simulation study of various loading and real-time dispatching strategies. Therefore the idea of combining different heuristics in a unique criterion directly originates from the need of more robust and versatile techniques for solving the scheduling problem in a flexible environment [9].

In a flexible line, whenever a machine $M_i$ completes an operation, a new job must be selected among those waiting in the buffer $b_i$. Hence, in this case the types of job waiting in queue constitute the set of alternatives $\mathcal{A}$. In this paper we select the job type, even if, in a more general case, each single job can represent an alternative.

We use a set of dispatching rules as decision criteria. The following example shows how to compute the degree of satisfaction of each alternative for each rule.

**Example 3.1.** Suppose that three different job types, $J_a$, $J_b$ and $J_c$, are waiting for service at the machine $M_i$. To combine SPT, EDD and LNQ rules in a single criterion, let the job types have the following parameters:

<table>
<thead>
<tr>
<th></th>
<th>$J_a$</th>
<th>$J_b$</th>
<th>$J_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{ij}$</td>
<td>100</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>$d_{j}$</td>
<td>600</td>
<td>300</td>
<td>900</td>
</tr>
<tr>
<td>$n_{ij}$</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

where $n_{ij}$ is the number of jobs of the same type queuing in the buffer. As indicated by the entries in boldface, in these conditions SPT rule selects a job of
<table>
<thead>
<tr>
<th>Based on arrival time</th>
<th>FCFS</th>
<th>First Come First Served: select the job arrived first at the machine.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FASFS</td>
<td>First At Shop First Served: select the job first arrived at the shop.</td>
</tr>
<tr>
<td>Based on processing times</td>
<td>SPT</td>
<td>Shortest Processing Time: select the job with the shortest processing time of imminent operation.</td>
</tr>
<tr>
<td></td>
<td>SRPT</td>
<td>Shortest Remaining Processing Time: select the job with the shortest sum of processing times of all the remaining operations.</td>
</tr>
<tr>
<td>Based on due date</td>
<td>EDD</td>
<td>Earliest Due Date: select the job with the earliest due date $d_j$.</td>
</tr>
<tr>
<td></td>
<td>MDD</td>
<td>Modified Due Date: select the job with the minimum value of $d_j^* = \max(t_{now} + P_i, d_j)$, where $P_i$ is the total remaining processing time for job $j$ at its $i$th operation.</td>
</tr>
<tr>
<td></td>
<td>MOD</td>
<td>Modified Operation Due date: select the job with the minimum value of $d_j^* = \max(t_{now} + t_{j.i}, d_j)$ where $d_j$ is the due date of operation $i$ for job $j$.</td>
</tr>
<tr>
<td>Based on number of operations</td>
<td>FOR</td>
<td>Fewest Operations Remaining: select the job with the smallest number of remaining operations.</td>
</tr>
<tr>
<td>Based on slack</td>
<td>SLACK</td>
<td>Slack: select the job with the least amount of available time before due date for remaining operations.</td>
</tr>
<tr>
<td></td>
<td>SSLACK</td>
<td>Static Slack: selects the job with the minimum value of static slack $s_j = d_j - r_j - P$, where $P_i$ is the total processing time of job $j$.</td>
</tr>
<tr>
<td>Based on setup times</td>
<td>SST</td>
<td>Shortest Setup Time: select the job with the shortest setup time for the imminent operation.</td>
</tr>
<tr>
<td>Based on queues status</td>
<td>LNQ</td>
<td>Largest Number in Queue: select the job with the largest number of jobs of the same type waiting for service.</td>
</tr>
<tr>
<td>Based on next queue status</td>
<td>WINQ</td>
<td>Work In Next Queue: select the job that requires the next operation in the machine having the minimum work load.</td>
</tr>
<tr>
<td></td>
<td>NINQ</td>
<td>Number of jobs In Next Queue: select the job that requires the next operation in the machine having the shortest queue.</td>
</tr>
</tbody>
</table>
type \( J_c \), EDD rule selects a job of type \( J_b \) and LNQ rule selects a job of type \( J_a \). To apply our algorithm we have first to transform jobs parameters in satisfaction degrees for each rule and then we must select the alternative that maximizes the combination of rules.

A simple way to obtain satisfaction degrees from real valued parameters is the normalization respect to their maximum or minimum values. Namely we have

\[
\mu_{\text{SPT}}(J_j) = \frac{t_{\text{min}}}{t_j}, \quad \mu_{\text{EDD}}(J_j) = \frac{d_{\text{min}}}{d_j}, \quad \mu_{\text{LNQ}}(J_j) = \frac{n_j}{n_{\text{MAX}}}
\]

These definitions can be easily extended to other rules. The following table shows the values corresponding to each rule, the satisfaction degrees of the Multiple Rule Criterion (MRC) and the alternative with the highest grade (in boldface). In this case we consider equal weights for all the rules.

<table>
<thead>
<tr>
<th></th>
<th>( J_a )</th>
<th>( J_b )</th>
<th>( J_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT</td>
<td>0.2</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>EDD</td>
<td>0.5</td>
<td>1</td>
<td>0.33</td>
</tr>
<tr>
<td>LNQ</td>
<td>1</td>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>MRC</td>
<td>0.2</td>
<td><strong>0.4</strong></td>
<td>0.25</td>
</tr>
</tbody>
</table>

In conclusion, the final choice of the MRC is job type \( J_b \).

The next example describes the effect of rule weights on the job type selection.

**Example 3.2.** Assume the same conditions of Example 1. We want to emphasize the influence of the rule SPT. Using an exponential weight \( w_{\text{SPT}} = 2 \) and a weight of 0.5 for the remaining rules we obtain the following weighted satisfaction degrees:

<table>
<thead>
<tr>
<th></th>
<th>( J_a )</th>
<th>( J_b )</th>
<th>( J_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT</td>
<td>0.04</td>
<td>0.16</td>
<td>1</td>
</tr>
<tr>
<td>EDD</td>
<td>0.71</td>
<td>1</td>
<td>0.57</td>
</tr>
<tr>
<td>LNQ</td>
<td>1</td>
<td>0.71</td>
<td>0.5</td>
</tr>
<tr>
<td>MRC</td>
<td>0.04</td>
<td>0.16</td>
<td><strong>0.5</strong></td>
</tr>
</tbody>
</table>

Hence, increasing the influence of rule SPT, the MRC selects the job type \( J_c \).

As we have shown, the scheduler is completely determined by the couple \( \mathcal{R}, w \), where \( \mathcal{R} \) is the set of heuristic rules considered in the multi-criteria algorithm and \( w \) is the vector of the exponential weights for the above rules. The choice of elements composing \( \mathcal{R} \) for a given problem is difficult, either because
the number of possible combinations is very high and because very few general information on the effects of the single rules are available. However, a rational choice should include few rules based on independent parameters, avoiding excessive complexity. The determination of suitable rule weights is an even harder task but, as shown later in this paper, it can be effectively tackled using guided search techniques as GAs.

4. The fuzzy multiple performance measure

Technical literature offers many ways for evaluating the performances of a scheduling policy. They include measures of costs, of inventory, of resource utilization and of throughput. Table 2 shows some example of widely adopted performance measures. However, usually primary production control goals, e.g. maximizing the profit, cannot be directly identified with values of one or more performance measures. Furthermore, analysis of these measures indicate that a scheduling procedure which works well for one criterion is not necessarily the best for some other [17]. Thus, a more realistic evaluation of a scheduling policy should at least consider a trade-off between the multiple aspects of the system behavior. For example, surveying problems related to the implementation of FMSs, Smith et al. [22] state that research should be directed toward scheduling procedures regarding satisfaction of due dates as primary objectives and considering machine utilization and minimization of WIP inventories as secondary objectives. On the other hand, to date most FMS scheduling research concentrates on criteria involving machine utilization and avoidance of the bottlenecks. Such considerations directly lead to the idea of a unique figure combining different performance measures.

Since the early developments in the field of fuzzy logic, Bellmann and Zadeh [3] used fuzzy sets to represent objectives and constraints in uncertain environments. Namely, fuzzy sets really lend themselves to combine different performance measures in scheduling issues, as described in the following.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Examples of performance measures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job parameters measures</strong></td>
<td></td>
</tr>
<tr>
<td>Completion time</td>
<td>( C_j )</td>
</tr>
<tr>
<td>Flow time</td>
<td>( F_j = C_j - r_j )</td>
</tr>
<tr>
<td>Lateness</td>
<td>( L_j = C_j - d_j )</td>
</tr>
<tr>
<td>Tardiness</td>
<td>( T_j = \max {0, L_j} )</td>
</tr>
<tr>
<td><strong>Shop parameters measures</strong></td>
<td></td>
</tr>
<tr>
<td>Makespan</td>
<td>Time spent to complete the entire batch of ( n ) jobs.</td>
</tr>
<tr>
<td>Work in progress</td>
<td>Sum of all jobs present in the shop at time ( t_{\text{now}} )</td>
</tr>
<tr>
<td>Resource utilization</td>
<td>( R_u = \frac{T_{\text{busy}}}{T_{\text{idle}} - T_{\text{busy}}} )</td>
</tr>
</tbody>
</table>
Suppose that we have to combine \( s \) performance indices \( \{x_i \in X_i \mid i = 1, \ldots, s\} \) where \( X_i \) is the spanning range of the index \( x_i \). For each index we can define a fuzzy set \( F_i \) expressing the degree of goodness of the index value. More precisely \( \mu_{F_i}(x_i) \) maps the range \( X_i \) in \([0,1]\) so that if the value of the performance is high (small) the membership grade \( \mu_{F_i}(x_i) \) is close to one (zero).

We can define a fuzzy set \( \mathcal{F} \) on the universe \( \mathcal{X} = X_1 \times X_2 \times \cdots \times X_s \) that combines all the fuzzy sets of the single measures

\[
\mathcal{F} = \bar{F}_1 \times \bar{F}_2 \times \cdots \times \bar{F}_s, \tag{5}
\]

with

\[
\mu_{\mathcal{F}}(x) = \min_{1 \leq i \leq s} \{\mu_{\bar{F}_i}(x_i)\}, \quad x \in \mathcal{X} \tag{6}
\]

Clearly, there is a close analogy between expressions (5), (6) and definitions (2), (3) leading to the MRC.

Exponential weights \( v_i \) can also be used to implement a scale of importance in the multiple measure. As their effects have been previously described, we directly introduce the final expression for the weighted multiple fuzzy measure:

\[
\mathcal{F}^* = (\bar{F}_1)^{v_1} \times (\bar{F}_2)^{v_2} \times \cdots \times (\bar{F}_s)^{v_s}, \tag{7}
\]

with

\[
\mu_{\mathcal{F}^*}(x) = \min_{1 \leq i \leq s} \left\{\left[\mu_{\bar{F}_i}(x_i)\right]^{v_i}\right\}, \quad x \in \mathcal{X}. \tag{8}
\]

In conclusion, the multiple performance measure is determined by the membership functions of the \( s \) indices \( \mu_{\bar{F}_i}(x_i) \) and by the exponential weights \( v_i \). Although a single weighting factor cannot easily express the importance of each criterion in an overall performance evaluation, we can estimate the weight \( v_i \) by considering the relative importance that experts would assign to it. When the number of evaluation criteria is high, a technique based on pairwise comparison, e.g. the Analytic Hierarchy Process (AHP) proposed by Saaty [20], could be profitably used.

The definition of the membership functions \( \mu_{\bar{F}_i}(x_i) \) is an even more complex key point. Due to the intrinsic nature of fuzzy sets, the choice of a membership function is always a matter of human judgment. However, any available information should be considered in the definition of the universes, i.e. the ranges of the single indices, and in the assignment of the satisfaction degrees. If no information about reference values for the performance indices is available in advance, simulation data analysis should be considered.

5. The genetic algorithm

This section describes how to use an Evolutionary Algorithm (EA) to optimize the weights \( w \) expressing the importance of each heuristic in the MRC.
Note that in the optimization procedure weights \( v_i \) representing objective hierarchy are fixed and defined a priori.

According to Fogel [11], EAs gather all the techniques that perform guided stochastic searches with a close analogy to natural evolution processes. Usually they are classified in four main subclasses, Genetic Algorithms, Evolutionary Programming, Evolutionary Strategies and Genetic Programming. All these techniques share the same iteration schema: the initialization of a population of solutions, the generation of new solutions by randomly altering the existing ones, the selection of "surviving individuals" according to a previously chosen objective function (the "fitness") and the production of new offspring by altering and combining the surviving individuals.

EAs have nowadays gained a primary importance in the field of scheduling, together with other stochastic search methods as Tabu Search and Simulated Annealing. Blazewicz et al. [5] provide a detailed overview and comparison of these search strategies with conventional ones in the field of scheduling. In general, EAs are used in scheduling for searching iteratively the best path in the tree of all possible decision sequences, which is often too wide for exact searches. However, EAs are not well suited for fine tuning structures, so that variations improving local search efficiency are frequently added. In our approach, we choose a GA as optimization strategy due to its ability to identify in short time regions of the solution space where the MRC has the desired behavior. Namely, as stated earlier in Section 4, the objective function is a combination of partial objectives whose satisfaction is modeled through fuzzy sets. Therefore, in such approximate context, local fine search is not a key-issue and the GA is implemented in a direct way.

The task of the GA can be formalized as follows. Given a set \( R \) of \( r \) rules to be combined on each of the \( m \) machines and an initial vector \( w_0 \) of \( u = rm \) rule weights, find the vector \( w^* \) which determines the highest value of satisfaction in the predefined fuzzy multiple performance index \( \mathcal{F}^* \). The following steps compose the GA developed for this task.

**Step 1** (Initialization of the population): a generic vector of weights represents an individual of the solution population, and must be coded with binary alphabet. Hence we must choose an upper bound \( w_{\text{MAX}} \) and a mapping relation with the relative precision for \( w_j \). According to [23], \( w_{\text{MAX}} \) should be in the magnitude order of \( r \). In the case of real numbers, an effective coding strategy is to map a binary decoded unsigned integer of length \( l \) linearly from \([0, 2^l]\) to the interval \([0, w_{\text{MAX}}]\) so that range and precision \( \pi \) are related by the following expression [12]:

\[
\pi = \frac{w_{\text{MAX}}}{2^l - 1}
\]

The multi-parameter coding can be obtained simply concatenating the single parameter codings in a string of length \( lu \).
The first individual of the initial population is

\[ w_j = 1, \quad j = 1, \ldots, u \]

that corresponds to an MRC in which all rules have the same importance in the decision. The remaining \( z - 1 \) individuals (where \( z \) is the population size) are randomly generated with uniform distribution.

**Step 2** (Fitness evaluation): as fitness of the \( k \)th individual \( w_k \), we consider \( \mu_{\mathcal{X}}(x_k) \) where \( x_k \) is the vector of \( s \) performance measures corresponding to \( w_k \) and determined by means of discrete event simulation of the production process [8].

**Step 3** (Selection of surviving individuals): at the end of the fitness evaluation phase, we assign a probability \( p_k \) to survive to each individual \( w_k \) as follows:

\[ p_k = \frac{\mu_{\mathcal{X}}(x_k)}{\sum_{h=1}^{z} \mu_{\mathcal{X}}(x_h)} \]

This mechanism is called "weighted roulette wheel" [12] because each individual has a probability of being extracted for reproduction proportional to its fitness.

**Step 4** (Generation of the new offspring): for determining a new generation of individuals we combine a set of surviving individuals, extracted with probability \( p_k \), by means of the genetic operators crossover and mutation, both occurring with previously fixed probabilities \( p_{\text{cross}} \) and \( p_{\text{mut}} \). In few words, the crossover generates a new individual joining two randomly cut portions of coded strings extracted from two different parent individuals, while mutation just randomly changes one bit in the string coding an individual. Fig. 1 describes the effect of these operators.

Steps 2-4 can be iterated until available computation time is exceeded, a suitable value of global performance index is achieved or, as in our implementation, a fixed number of simulation runs has been executed.

![Diagram](image)

**Fig. 1.** Crossover and mutation operators.
6. Case studies

In this section we test the multi-criteria scheduler and the GA optimization on two different case studies, both implemented through SIMAN simulation software [19]. All the results refer to the following GA configuration: population size $z = 50$, $p_{\text{cross}} = 0.6$ (one point crossover), $p_{\text{mut}} = 0.03$ (one bit mutation), number of bits per parameter $l = 10$, stop of the algorithm after 3500 runs.

The first case study is based on a model of the IBM Automated Circuit Card line described in detail in [1]. The system consists of four machines producing six different types of piece. Table 3 lists processing times and operation sequences. Setup and transport times are constants and included in the processing times.

The four machines are connected in series by a material handling system which allows job of type $J_j$ to by-pass machine $M_i$ if $t_{ij} = 0$. Each machine has a buffer with capacity of 30 places. A batch of 180 parts (30 for each type) is loaded into the system at the rate of 3 parts/min with a cyclic order (1, 2, 3, ..., 6, 1, 2, ...).

In this case, the set of rules $R$ that compose the MRC contains the three rules FCFS, SPT and CLB, while the fuzzy multiple performance measure $\tilde{F}$ is composed by the indices makespan, mean and maximum flow time, mean and maximum Work In Progress (WIP) and resource utilization. Using AHP scale of importance [20], lowest weight is given to maximum WIP and maximum Flow Time indices and highest weight to all the other indices. According to previous simulation studies, we define fuzzy sets $\tilde{F}_j$ expressing the satisfaction of objectives as shown in Fig. 2.

Both the initial MRC (with $w_j = 1$ for all $j$) and the final MRC optimized with GA are compared with the three component rules, each on its own. We have performed 50 optimization processes with different seeds for the GA. Fig. 3 shows the mean convergence curve on these 50 samples. All the runs have shown a regular behavior, (i.e. no remarkable deviation from the mean curve in Fig. 3).

The performance evaluation of a single individual requires a simulation taking approximately 1 s on a PC 486 100 MHz Workstation; hence each run of

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>$M_2$</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>30</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>$M_3$</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>$M_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0</td>
<td>80</td>
</tr>
</tbody>
</table>
Fig. 2. Membership functions of the indices combined in $\mathcal{F}$. 

Fig. 3. Mean convergence on 50 GA runs.
the GA reaches the predefined stopping condition in less than 1 h. Fig. 4 compares the five policies, i.e. the three component rules and the MRC before \((w = 1)\) and after (final) the optimization with the GA, for five of the six performance measures composing the multiple one (for reasons of space, maximum WIP is omitted). The figure also compares the multiple performance indices. The GA is able to drive the MRC to an extremely higher value of the multiple performance measure. The entity of this improvement, shown in the last diagram of Fig. 4, is partially related to the way the multiple index is computed. In fact, each of the component rules and the initial MRC have at least one performance index below a satisfactory value. Using the minimum as aggregation operator and given the shapes of the membership functions expressing objectives satisfaction (see Fig. 2), this leads to a very low value of the fuzzy multiple measure. Instead, the GA allows the final MRC to reach a good trade-off of performances, which furthermore in this case corresponds to an enhancement of all the single indices, as shown in Fig. 4. Different experiments on the same model confirm that, upon less than 1 h of weights tuning, the final MRC always dominates the comparison.

Fig. 4. Performance comparison for the first case study.
Table 4
Operation sequences and times of the second case study

<table>
<thead>
<tr>
<th></th>
<th>Part 1</th>
<th>Part 2</th>
<th>Part 3</th>
<th>Part 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>First Oper. (Duration)</td>
<td>M2 (120)</td>
<td>M1 (110)</td>
<td>M2 (100)</td>
<td>M3 (200)</td>
</tr>
<tr>
<td>Second Oper. (Duration)</td>
<td>M1 (200)</td>
<td>M3 (130)</td>
<td>M1 (120)</td>
<td>M2 (100)</td>
</tr>
<tr>
<td>Third Oper. (Duration)</td>
<td>–</td>
<td>M2 (130)</td>
<td>M3 (100)</td>
<td>–</td>
</tr>
<tr>
<td>Unloading</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

The second case study is a simple FMS developed in [10]. The system, consisting of three workstations and one loading–unloading station, has to process four different part types, with routes and processing times indicated in Table 4. The test process is a batch of 200 parts (50 for each type) with all raw parts ready for processing at the loading station. In this case the MRC is composed of EDD, SPT, CLB and WINQ rules, and the multiple measure \(F^r\) combines makespan, resource utilization, mean lateness and mean WIP, with membership functions similar to those showed in Fig. 2. In this case, all the indices are considered with the same importance.

Fig. 5 shows the comparison of performance in this second case study (mean WIP is omitted). Also in this context the MRC provides a better compromise of performances, this time reducing mean lateness while keeping the other indices within their best values.

![Fig. 5. Performance comparison for the second case study.](image-url)
7. Conclusions

The results confirm that fuzzy logic and evolutionary algorithms provide promising methods to face scheduling problems in flexible production environments. In fact, these production systems have some peculiarities: (1) the scheduling objectives are multi-faceted and often conflicting with each other; (2) the scheduling approaches are generally based on heuristics that influence the production performance indices in an involved way; (3) the discrete event simulation is the only means to compute the performance indices; so it is impossible to use classical methods to optimize them.

In this framework, fuzzy logic is an effective methodology to set decision mechanisms based on multi-heuristics (properly weighted) and to define performance measures combining different production objectives. On the other hand, evolutionary algorithms allow us to solve optimization problems lacking any information relating decision variables with merit figures. Indeed, extended simulation experiments can help in enlightening these relationships.

The flexibility of the methodology proposed here makes it applicable to different productive contexts, also with production objectives changing with time. The numerical experiments confirm the effectiveness of the approach and ask for further research, applying the methodology to other issues of the production scheduling (e.g. routing, part loading, lot sizing, etc.) and improving the genetic algorithm with special features for the implementation hereby proposed.

Acknowledgements

This work was partially supported by 40% MURST funds and by the Italian National Research Council under contract C.N.R. no. 96.00056.PF01.

References


