Evolutionary adaptation of dispatching agents in heterarchical manufacturing systems

BRUNO MAIONE†* and DAVID NASO†

We propose a new approach to job flow adaptive operational control in advanced manufacturing systems. The major feature of the method is the distribution of the control tasks among completely autonomous intelligent agents. Namely, agents are implicitly coordinated by a nature-analogous adaptation mechanism, which continuously tunes the free parameters of the control law of each agent. The proposed approach is effective and reactive to severe disturbances and changes in the manufacturing environment. Simulation experiments illustrate the operational distributed approach and its response to faults.

1. Introduction

Within the classical conception, shop floor control is the ‘front end’ of production planning systems that are designed to support a hierarchical information management. Namely, traditional decision making has an orientation towards centralized approaches that emphasize a deterministic view of the technological and logistical production process. In particular, control activities assume that all the system parameters are known and remain unchanged throughout the course of time between conceiving the plan and its realization. Hence, the shop floor control must execute the schedules ‘as given’ by decisions at upper levels of the hierarchy. Consequently, the shop floor control is ‘blind’ to deviation of the real behaviour from the programmed goals and cannot grasp unexpected events (machine downtimes, material shortages, demand of changes, etc). On the other hand, attempts to face uncertainty by increasing the precision and detail of mathematical planning models, in practice, have not lead to success.

In recent years, however, there has been a general trend to enhance control intelligence and reactivity to changes. The strategy to achieve this objective is to develop distributed control architectures supporting the cooperation of autonomous, distributed controllers (agents). Consequently, a departure from a centralized architecture and a move toward decentralized shop floor control are clearly emerging. In principle, indeed, the field controllers can quickly respond to local emergencies and disturbances. The new concepts have lead to Multi-Agent Systems (MAS) architectures that distribute information, decision and control tasks among a network of autonomous entities (the agents). However, in the manufacturing area, agents have often pre-programmed objectives to achieve. In other words, the interaction among agents obeys a mechanism that is designed \textit{a priori} and cannot change over time. Due to the inherent myopia of agent decisions and the continuous changing in the con-
dition of the manufacturing process, the pre-programmed decision rules cannot be the most appropriate at each moment. Hence, changing the decision rules at the right time, and making the control agents react quickly to disturbances represent interesting research directions.

In this paper, we propose a distributed system of intelligent agents that control the job flow using a set of weighted heuristic rules. An Evolutionary Strategy continuously improves the agent decision rules, so implementing an efficient adaptation to changes in the plant operating conditions.

This paper is organized as follows. Section 2 overviews related research. Section 3 describes the architecture of the proposed system. Sections 4 and 5 describe the agent decision algorithm and agent online optimization, respectively. Section 6 illustrates the simulation results for a Flexible Assembling System controlled by the proposed MAS. Finally, Section 7 draws conclusions.

2. Related research

In the last decade, the concepts of agents, MAS and heterarchies have received great attention. According to these new paradigms, control is distributed throughout the shop floor, so that each local decision unit ensures a reactive system response to abrupt disturbances. In a recent paper, Baker (1998) provides an extensive review of the research in this area. Baker notes that most MAS for shop floor control merely implement dispatching rules inspired by market-based mechanisms. The works of Solberg and Lin (1992), of Teredesai and Ramesh (1998), and of Sousa and Ramos (1999) confirm this trend. Some other researchers, among them Duffie et al. (1988, 1996), prefer to focus their attention on heterarchical architectures supporting the new control methodologies. Parunak (1994) describes many recent research applications and successful industrial implementations of the MAS paradigm.

Usually, these distributed systems require a coordinator agent that promotes dynamic collaboration of intelligent entities, with the goal of achieving both local and global objectives. Mechanisms inspired by approaches used in the Blackboard Systems (Nii 1986) and in the Contract Net (Parunak 1987, Shaw and Whinston 1988, Smith 1980) are used for communication and negotiation among the agents. However, a more recent trend seeks an increasing autonomy in control agents, by reducing global information usage and by a general loosening of, or abolishing, master/slave relationships. Following this line, we develop here a MAS structure where there is no unit coordinating the distributed control agents. These entities gather information reflecting the status of their near environment but do not directly compete nor collaborate with each other. Namely, the global functionality of the manufacturing floor emerges only as a result of the actions of individual agents obeying an adaptation mechanism. This mechanism is in charge of tuning the control laws of the agents so that objectives are achieved and the global performance is optimized. However, this task does not imply a dynamic model describing the effect of a specific control action on the system performance. Rather, the automatic adaptation mechanism, which is inspired by the design method developed by De Jong (1980), follows a nature-analogous approach imitating genetics. The mechanism continuously seeks to produce new agent decision rules (solutions) through selection, mutation and reproduction of existing solutions. In this procedure, intermediate results are selected according to relative ‘fitness’.

The application of Genetic and Evolutionary Algorithms (EAs) (Fogel 1995) in the area of manufacturing is not new. In particular, nature-analogous approaches
offer efficient procedures to solve offline complex scheduling problems (Holsapple et al. 1993, Watanabe et al. 1995, Yip-Hoi and Dutta 1996, Shi 1997, Song and Hughes 1999). EAs are also used to optimize the agent behaviour of MAS in many different contexts, e.g. colonies of intelligent robots (Dorigo and Schnepf 1993, Liu et al. 1997), Image Feature Extraction (Liu et al. 1997), and other AI benchmarks (Trenaman 1998). However, in all these applications, EAs deal with agents operating in a static or slightly varying environment. In our paper, on the contrary, agents rule the job flow in the extremely uncertain and dynamic environment of the shop floor. Moreover, to the authors’ knowledge, the proposal of using EAs as an online adaptation strategy for job flow control is new.

To implement adaptivity, we must establish how intelligent agents modify their actions, taking into account the system operating conditions. Recently, other authors have investigated similar problems. Park et al. (1997) define a mapping ‘system-state-to dispatching rule’, to implement an adaptive sequencing algorithm in production lines. The mapping is based on if–then statements such as, for example, ‘if system utilization is greater than “a” and buffer size is “b” and... then the dispatching law is SPT’. The mapping is optimized using an ‘inductive learning approach’ based on a set of training examples. Analogously, Yu et al. (1998) use fuzzy rules to associate a set of ‘environmental variables’ describing the manufacturing conditions (e.g. the current workload or the remaining production time for jobs at hand) to a set of dispatching rules. Generally, however, associating a specific dispatching rule to each status of the plant is a not an easy task. Namely, in a given operating condition, there is no evident relationship between system performance and control actions on the plant. Consequently, it is impossible to define in advance a set of general criteria to switch between dispatching policies. On the other hand, rules are case-dependent and perform differently in different plant conditions (Park et al. 1997) and in different classes of systems.

To overcome these difficulties, our agents adapt their action to the plant status using a multi-criteria decision-making algorithm that encompasses multiple weighted dispatching rules. Namely, it is well known that multi-criteria decision strategies improve workshop operation (Grabot and Geneste 1994). In particular, we use Fuzzy Set concepts to implement a trade-off among different decision rules, following an approach similar to that introduced by Custodio et al. (1994) Ben-Arieh and Lee (1995) and Kazerooni et al. (1997). However, while in all these papers rules are associated heuristically, in our approach the adaptation procedure establishes the influence of each rule by means of a weighting factor, grading the single rule contribution on the global desirability. In particular, our adaptation algorithm incorporates the rule weights into the chromosome of individuals subject to an evolutionary process. The next sections illustrate the details of the procedure.

3. The structure of the shop floor control system

Figure 1 describes the schema of the job flow control system. Figure 1(a) represents the real time control module. The control system is physically and logically distributed among a network of autonomous agents residing in different locations on the factory floor. Each agent measures the actual values of the variables characterizing the current processing status and, consequently, makes a decision among a set of possible alternatives (e.g. which part has to be processed among those waiting for service). An actuating agent’s decision influences the dynamics of the whole factory
Hence, the job-flow control results from the activity of the single controllers/agents. Many MAS for job flow control work in a similar way and can be completely described by the left-hand part of figure 1. The performance of such MAS is generally evaluated, either in simulation or on the field, by means of a set of standard performance indices (PI), as mean resource utilization, mean job throughput or makespan. The right-hand part of figure 1 describes a second loop (the dynamic optimization module) that uses the real-time values of a set of PI to modify the agent decision mechanism over time.

Typically, a job flow control must manage loading, routing and sequencing activities. Since this paper focuses on routing and sequencing problems only, we consider ‘closed’ FMS, in which the loading of a new raw part occurs only when a finished product leaves the manufacturing floor (except during plant start-up and forced changes in the workload). We also assume that the system has to produce all the different part types in equal quantities, and that a cyclical loading policy (A, B, C, ..., A, B, C, ...) selects the next part type entering the shop floor. Finally, assuming that all part types have the same average processing time on all the possible processing sequences, (as in the case study described in section 6), the cyclical loading policy can ensure a balanced workload. However, it can be easily noted that many of the mechanisms composing the proposed adaptation strategy can be easily modified to deal with different types of FMSs.

Part routing decisions select which of the available servers must provide service to a part whenever it requires a new operation. In our systems, Part Agents (PAs) are in charge of making these decisions. A PA is a software controller that is associated with each physical part as it enters the system. Each PA contains both the data
necessary to process the associated part (i.e. alternative process plans, work in progress information, etc.) and the decision algorithm. Whenever a part ends an operation and becomes ready for the next one, its PA must first determine the set of possible subsequent operations. Then, it must identify the subset of alternative machines potentially at its disposal for such operations. The agent can perform these two operations autonomously, since it retains all the necessary information. Subsequently, the PA has to gather real-time information on the eligible servers. Consequently, the PA connects to all the alternative machines to acquire the necessary real-time data. The decision process automatically ignores failed machines, since they do not respond to the information request. At this point, the PA selects the best destination by using a fuzzy weighted combination of decision rules and initializes the procedures of part transfer to the selected workstation.

At any time a machine ends an operation, the sequencing (or priority setting) decision selects the next part to be processed from among those waiting in the queue for service. Workstation Agents (WAs), which are decisional entities associated with each machine in the system, are in charge of this decision. In addition, WAs use a fuzzy weighted combination of decision rules (Fanti et al. 1998), each considering a different aspect of the available alternatives (processing times, set-up requirements, deadlines, etc.).

It is important to observe that PA and WA decisions are independent. When a PA alone, controlling the set of alternative machines, cannot decide autonomously to refuse a service request, unless the associated machine is down. This means that PAs and WAs neither negotiate nor compete for a common profit, but rather interact indirectly with each other. The adaptation strategy establishes the interaction mechanism by improving the satisfaction of the global goal of the entire control system. Therefore, in this paper, we call only those controllers that are changing the decision logic over time agents. Each agent can be viewed as a function:

\[
CA : E \times S \times \Omega \rightarrow A,
\]

where

- \( E \) is the set of all the possible conditions of the environment (i.e. the processing resources and the other agents),
- \( S \) is the set of all possible agent conditions. The position, the status of the process plan, the number of completed operations and the remaining ones describe the condition of the PAs. Similarly, the current set-up, the input and output buffer contents are the variables describing WAs conditions,
- \( \Omega \) is the space of the tuning vectors. In this paper, \( \omega(t) \) is the set of weights associated with the single PA and WA decision rules. The weights act as tuning devices for agents’ decision policy: if \( \omega \) changes, the response of the agent to the same configuration \( e \in E \) and \( s \in S \) also changes.

So if \( t \) indicates the decision time, \( CA \) maps the status of the environment \( e(t) \in E \), the internal conditions of an agent \( s(t) \in S \) and the value of a tuning vector \( \omega(t) \in \Omega \) in the set \( A \) of possible decisions.

Agents adapt their decision policies with the following mechanism. Both PAs and WAs have a limited life cycle. PAs’ cycle time can be defined in a straightforward way as the flow time, i.e. the time required to complete the processing of the associated part. For the WAs, the cycle time must be a compromise between the need of
extending the length of the observations to obtain significant indices of agent performance, and the need to measure, as rapidly as possible, the performance of the WA to guarantee a fast response of the evolutionary adaptation. According to a set of preliminary experiments on the considered case study, setting the WA life cycle as the time required to process 200 operations allows a good trade-off between the significance and the rapidity of the WA real-time performance estimation process. At the cycle completion, each agent is replaced by a new one of the same type, and the evaluation module determines its performance. The numerical value of this PI is used as a measure of agent fitness with respect to the current operating conditions of the manufacturing floor. This fitness guides the agent replacement process: the higher agent fitness, the higher the number of agents that will inherit its decision strategies. The objective of this strategy is twofold. When shop floor conditions are stationary the dynamic optimization module acts as an online optimization strategy that progressively improves the performance of the entire MAS. When unexpected perturbations of the operating conditions (i.e. a failure) occur, the genetic adaptation automatically starts the search for more effective agents for the new operating context.

4. The decision algorithm of the controller agents

Each PA must choose the next destination for the associated part when it completes a given operation. Analogously, when a workstation completes a service, the WA must choose the next part to process among those waiting in the buffer. Once the preliminary step of gathering the real time information about the available alternatives has been completed, both the PA and the WA make their decision using the same algorithm. We assume that the general decision problem involves $n$ alternatives (workstations for PAs and parts in buffer for WAs). The number $n$ may be different for each decision, since it depends on the number of workstations that actually respond to a request of service (for a PA), and on the current number of parts in queue (for a WA). We also assume that each agent must evaluate the alternatives on the basis of $m$ different heuristic criteria, taking into account different aspects of the available alternatives (e.g. processing time, slack time and set-up requirements). The number $m$ and the decision rules are selected in advance and fixed. Thus, any decision process builds on a set of $n \times m$ real time measures that can be easily arranged in a $m \times n$ matrix, referred to as a decision matrix (Zimmermann 1993). The objective of the decision is to find the alternative providing the best trade-off of satisfaction of the $m$ heuristic judgements. However, the data are non-homogeneous. Namely, the decision matrix may contain distances, numbers of jobs in queue, processing times, etc. Moreover, the meaning of matrix elements is context-dependent, since the same value of a parameter may have two different meaning in two different decisions. A simple way to aggregate the decision matrix data is defining dynamic fuzzy membership functions. More precisely, assuming that higher desirability corresponds to higher (smaller) values of the attribute, for a given criterion $C_i (i \in \{1, 2, \ldots, m\})$, a number in $[0,1]$ can be easily found normalizing with respect to the maximum (minimum) value of the attributes. Fanti et al. (1998) describe further details of this procedure.

The above fuzzification procedure leads to a fuzzy decision matrix $\tilde{S} \in [0,1]^{m \times n}$ where each element $s_{ij}$ expresses how much the $j$th alternative (a workstation for a PA or a part for a WA) satisfies the $i$th criterion. Therefore, each row of the fuzzy decision matrix is a fuzzy set $S_i$ expressing the satisfaction of the $i$th criterion in the universe of the available alternatives. Yager’s Multi-Attribute Decision-Making...
algorithm (Yager 1978) computes the global decision criterion as the fuzzy intersection of the \( m \) fuzzy sets, i.e.
\[
\hat{s}_j = s_{ij}^{w_1} \ast \cdots \ast s_{mj}^{w_m} \quad j = 1, \ldots, n, \tag{2}
\]
where the symbol \( \ast \) indicates a generic fuzzy T-norm (Klir and Yuan 1995) and \( \omega = [w_1, w_2, \ldots, w_m] \) is a vector of weighting factors grading the influence of the \( i \)th attribute in the computation of the global desirability. Yager’s algorithm uses the minimum as the intersection operator and crisp exponents as criteria weights, but the literature suggests many other ways for fuzzy criteria aggregation, e.g. a fuzzy convex sum or conjunctive aggregation (Klir and Yuan 1995). The alternative, achieving the highest satisfaction in the global criterion, is given by:
\[
\hat{s}^+ = \max_{j = 1, \ldots, n} (\hat{s}_j) \quad h \in \{1, 2, \ldots, n\}. \tag{3}
\]
This alternative is selected as the final decision. This fuzzy algorithm is particularly simple and effective, and can assume all the intermediate behaviours between the \( m \) component rules by an appropriate choice of the decision weights \( \omega \).

5. Dynamic optimization of the control agents

The following subsections contain, respectively, a brief overview of evolutionary algorithms and a detailed description of their implementation as online optimization strategies for the dispatching agents.

5.1. The evolutionary algorithms

Evolutionary Algorithms (EAs) are parallel stochastic search methods imitating Darwinian evolution laws (Fogel 1995, Michalewicz 1996). All EAs share the same basic schema. They firstly initialize a population of numerical problems and code them in an appropriate alphabet (usually strings of real or binary numbers). Secondly, they generate new populations by evaluating the fitness of each individual (the value of the objective function associated to the solution) and by selecting a group of individuals that survive and reproduce in the successive populations according to fitness-based laws. Basically, the various evolutionary techniques differ in the coding strategy, and in the mathematical operators emulating the natural selection and reproduction processes. Generally, Genetic Algorithms use binary coding and are mainly based on crossover (i.e. they generate new solutions crossing the genes of parent couples with high fitness), while Evolutionary Strategies (ES) use real-valued coding and generate new solutions mainly through mutation (i.e. applying small additive perturbations to the best individuals in the previous population). EAs can converge toward better solutions in a wide class of hard numerical optimization problems, with high-dimensional, non-convex, multi-modal or discontinuous solution spaces (Michalewicz 1996). The power of EAs lies in their total independence from any a priori information about the objective surface.

Most of the applications of EAs in many different fields, including manufacturing plant scheduling, deal with static optimization problems, where the objective surface does not change over the course of the optimization process. Technical literature has also proposed EAs in domains where the objective surface varies over time. However, to obtain useful information by applying EAs to dynamic optimization problems, the rate of change of the objective surface must be considerably slower than the rate at which the EA generates and evaluates new individuals. The online
optimization of a manufacturing process can be often viewed as a problem with a ‘drifting objective surface’ (De Jong 1980). In other words, the objective surface drifts slowly over time for small changes in the manufacturing plant conditions (e.g. changes in the production load or mix). On the other hand, manufacturing systems are also subject to various unforeseen faults that can abruptly change the system state and consequently the objective surface. For instance, the occurrence of a machine failure suddenly reduces the processing capacity of the plant, and consequently the theoretical minimum value of the mean flow time of processed parts becomes abruptly higher. In such cases, the effectiveness of the genetic approach depends on the average time between two consecutive perturbations, which should be long enough for a search to converge toward better individuals.

Evolutionary Strategies (ESs) (Michalewicz 1996) are a type of EAs particularly suited for dynamic optimization problems thanks to the self-adaptation of their configuration parameters. Namely, such algorithms not only search for better problem solutions, but simultaneously optimize their configuration parameters (e.g. the variance of the perturbations added to new individuals) in order to improve the efficiency of the convergence process. The optimization of the configuration parameters is obtained by considering them as new chromosomes of the individuals (i.e. as part of the problem-solution) and subjecting both solution and configuration parameters to the evolutionary process. This paper compares two different types of ES for agent evolution, namely a standard $(\mu + \lambda)$-ES (Fogel 1995) and a variation with lognormal mutation (Bäck 1998). According to previous studies (Bäck 1998), the latter algorithm especially has good capabilities to track optimum solutions in simple dynamic environments.

Any individual $I$ of the $j$th population $I^{(j)} = (\omega^{(j)}, \sigma^{(j)})$ consists of the weight vector $\omega^{(j)}$ of $m$ components associated with the agent’s decision policies and the standard deviation vector $\sigma^{(j)}$. The vector $\sigma^{(j)}$ defines the amplitude of the additive perturbation applied to each of the elements of $\omega^{(j)}$ if the individual $I^{(j)}$ is selected for reproduction. In the first algorithm (standard ES, SD-ES for brevity), $\sigma^{(j)}$ is adapted during the execution of the ES with the following heuristic rule (1/5 success rule, Michalewicz 1996):

$$
\sigma^{(j+1)} = \begin{cases} 
\alpha \cdot \sigma^{(j)} & \text{if } \varphi < \gamma \\
\beta \cdot \sigma^{(j)} & \text{if } \varphi > \gamma \\
\sigma^{(j)} & \text{if } \varphi = \gamma,
\end{cases}
$$

(4)

where $\alpha < 1$, $\beta > 1$ and $\gamma > 0$ represent three configuration parameters chosen empirically ($\alpha = 0.82$, $\beta = 1/0.82$ and $\gamma = 1/5$ in the standard algorithm) and $\varphi$ is the success ratio of the mutation operator (the ratio of successful mutations to all mutations in the last generation). In the second algorithm (Lognormal ES, LN-ES for brevity), the standard deviation of the algorithm mutation $\sigma^{(j)}$ is subject to the following adaptation algorithm:

$$
\sigma^{(j+1)}_i = \sigma^{(j)}_i \exp (\tau' N(0, 1) + \tau N_i(0, 1)) \quad i = 1, \ldots, m,
$$

(5)

where $\tau' = (\sqrt{2 \cdot \sqrt{m}})^{-1}$ and $\tau = (\sqrt{2 \cdot m})^{-1}$ represent the adaptation rates determining the velocity of the self-adaptation process, and $N(0, 1)$ and $N_i(0, 1)$ indicate the Normal distribution. These ESs constitute the engine of the adaptation strategy. The next subsection describes their online implementation.
5.2. The online adaptation strategy

At any time during the manufacturing process, different agents (i.e. PAs and WAs with different $\omega$) are present in the system. The agent diversity is the base of the evolutionary processes, which are separate and independent for machine and part agents. In the following, we focus our attention on the evolutionary mechanism of PAs. We make the following assumptions.

1. The manufacturing system is closed, i.e. after the maximum workload, $q$, has been reached, a new part can enter the system only when a finished part leaves it. Parameter $q$ is set by plant operators.
2. New raw parts are constantly available for loading (infinite input buffer).
3. Finished parts can always be unloaded (infinite output buffer).
4. The same quantity of each part type has to be produced.

Figure 2 describes the general schema of the adaptation strategy. A population of the evolutionary process is a set of $p$ distinct PAs. The goal of each PA is to complete the processing (of the associated physical part) in the shortest time. Thus, the flow time measures the performance of a PA. To reduce the influence of random circumstances on the fitness measure, we define, as the fitness of a PA, the average of the flow times obtained by at least $h$ identical clones of that agent.

The evolutionary process starts at the beginning of a new production cycle, when the first $q$ raw parts enter in the system sequentially. Figure 3 shows the loading sequence assigning PAs to raw parts. Once an agent is assigned to a physical part, its decision strategy $\omega$ remains fixed. Initially, the evolutionary algorithm creates a number $r > p$ of PAs. To obtain averaged performance, we create $k > h$ clones of each agent. In addition, we assign the $r$ PAs and the subsequent $r \cdot (k - 1)$ clones to the first $r \cdot k$ raw parts with the order of figure 3. Then, the first $p$ PAs that complete the operations constitute the first population, while the remaining $r - p$ agents will be placed in the subsequent one. In this way, the evaluation of the first population is completed when only the average fitness of the fastest $p$ among $r$ initial PAs has been measured. So, by assuming $r > p$ we avoid the wait for slower or blocked parts.

![Figure 2. Schema of the evolutionary process for PAs.](image-url)
Furthermore, the overlap of populations allows the evolutionary strategy to load continuously raw parts and to assign new PAs belonging to the next population, even if the current one is not completed.

Once a raw part enters the system, the associated PA controls its route autonomously. As its processing is complete, the part leaves the plant and its flow time is recorded. As figure 4 shows, to evaluate the fitness of an agent, the completion of at least $h$ of the $k$ clones is necessary. The remaining $k-h$ clones will be included in the average only if they complete their job before the end of the current population. A population ends (the iteration) when at least $h$ clones of each agent in the population have completed their job. Consequently, the small redundancy in the number of clones ($k > h$) helps in making the evolutionary adaptation more independent of random delays affecting only a minority of the clones.

![Figure 3. Loading of the first set of parts.](image)

![Figure 4. Status of fitness evaluation at a generic time.](image)
When the fitness of all the $p$ PAs of the current population has been computed, the evolutionary strategy proceeds to create a new population according to the following schema.

**Step 1.** Rank the $p$ PAs according to their fitness;

**Step 2.** Select the best $\mu$ parent PAs;

**Step 3.** Generate a subset of $\lambda$ offspring PAs (where $\mu \lambda = p$);

**Step 4.** Compose the new population with the $\mu$ parent PAs and the $\lambda$ offspring PAs and assign it to the next raw parts entering the system (with the same schema of figure 3).

In this feedback loop, the fitness measures the performance that a PA is able to obtain in at least $h$ trials, and a population represents a measure of the performance that the last $p$ PAs are able to obtain in the current operating conditions. Since only information on the last population contributes to building the new generation of PAs, $p$ can be viewed as the length of the system memory: the larger is $p$, the larger is the number of past PAs taking part into the process of computing the future PAs. In static optimization problems, the use of large populations increases the inherent parallelism of the evolutionary algorithm and generally leads to a higher efficiency of the search strategy. On the contrary, in dynamic optimization problems, smaller values of $p$ imply shorter times to complete the evaluation of a population and consequently a faster response to perturbations.

The evolution of WAs follows the same adaptation schema. At the beginning of a production cycle, a WA with unitary weights is assigned to each machine. Then, after a predefined time interval, new randomly generated agents replace the first ones. The lifetime of WAs must be sufficiently large to obtain a consistent measure of their performance. In our simulations, it corresponds to the average time necessary for processing 200 operations. The agent substitution is repeated $p-1$ times, until the first population of $p$ machine agents is completed. Finally, the generation of a new population of agents follows the same schema adopted for PAs. For WAs, the fitness is the fuzzy aggregation of two criteria, the minimization of a part in queue waiting time and the minimization of the number of set-ups.

6. A case study

This section describes the application of the evolutionary multi-agent dispatching system to control a Flexible Assembling System (FAS) producing printed circuits boards. Figure 5 shows the FAS layout. The plant has three workstations for the installation of Integrated Circuits (ICs) and a machine for assembling the circuit boards in a single final product.

The plant processes three different types of boards, each requiring the installation of some of the six different types of ICs. The three workstations are equipped with a partially different variety of ICs, as described in table 1. Each IC can be installed by at least two machines, so that the system can continue to process all the parts even with a failed workstation. Machines install ICs of the same type without set-up. On the contrary, there is a set-up delay (7 s) when machines must switch between different IC types. An automated system continuously supplies ICs to the three workstations in lots of 50 units unless a temporary lack of components occurs.

A board-handling system composed of two unidirectional Automatically Guided Vehicles (AGVs) transfers parts from and to input/output and processing stations (table 2 shows the transfer times). Through the assumption of section 5, raw boards
are immediately available for loading and finished boards can always leave the system. Moreover, the FAS is a closed system, i.e. a raw board can enter the system only when a finished board exits the FAS, except during plant start-up or forced changes in the maximum workload $q$. The loading process is therefore governed by the adaptation strategy described in the previous section.

Table 3 shows type and quantity of ICs that must be installed on the three boards. According to this table, each board has to visit at least two workstations to be completed.
Each station has both input and output buffers of capacity 5 and 2, respectively. Since the installation of ICs is free of any sequence constraint, each board can be completed with many alternative sequences of operations.

The model of the FAS has been developed with Arena 3.0 DEDS Simulation Software (Kelton 1998), whereas agents and evolutionary strategies have been implemented in C++ language. In all the experiments, we propose to use two global performance indices, namely the moving average of the flow time of the produced boards and the total throughput (the ratio number of produced parts on production time).

The aim of the first simulation study is to determine appropriately the parameter setting of both the multi-criteria algorithm and the evolutionary strategy. Figures 6(a) and (b) and 7(a) and (b) report two samples of this initial configuration phase. The four figures show the evolution of the two performance indices corresponding to different values of the parameters $h$ (figure 6) and $\sigma^{(0)}$ (the initial mutation standard deviation) (figure 7). For the sake of clarity, the figures draw only three curves for each parameter. In these simulations, the adaptation mechanism involves PAs, whilst WAs employ a fixed ‘Shortest Set-up Time’ (SST) heuristic decision logic. Figures 6(a) and (b) show that the average on a large number of clones per agent ($h = 25$) tends to produce a comparatively slow initial agent adaptation. Vice versa, with a small number of sampled clones ($h = 5$) the adaptation tends to lose efficiency on longer terms. The intermediate value ($h = 10$) provides a good trade-off between initial speed and long term optimization and is therefore a good setting for subsequent simulations. Figure 7 shows similar analyses for the initial standard deviation $\sigma^{(0)}$. The other ES configuration parameters were set using similar investigations.

On the basis of these experiments, we assume the following.

(a) Rules used by PAs: Shortest Operation Completion Time (SOCT), Shortest Distance from next station to visit (SD), Shortest Set-up with reference to the Last Part in queue (SSLP);
(b) Rules used by machine agents: First Come First Served (FCFS), Earliest Due Date (EDD), Shortest Set-up Time with reference to the last process (SST);
(c) Parameter setting of the ES for both PA and MA: $p = 10$, $\mu = 3$, $\lambda = 7$, $h = 10$, $\sigma^{(0)} = 0.2$.

The second set of simulations evaluates the efficiency of the proposed MAS and, in particular, of the evolutionary adaptation in dynamic operating conditions. We consider four different types of perturbations of nominal plant operation: an abrupt change of workload from $q = 5$ to $q = 15$ and subsequently again to $q = 5$ (figure 8), a temporary failure/repair of a workstation (figure 9), a temporary lack of a type of IC (figure 10) and a temporary partial failure at the AGV system (figure

<table>
<thead>
<tr>
<th>Board</th>
<th>IC A</th>
<th>IC B</th>
<th>IC C</th>
<th>IC D</th>
<th>IC E</th>
<th>IC F</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. ICs requirements of the three boards.
Figure 6(a). Evolutionary optimization with different numbers of clones.

Figure 6(b). Evolutionary optimization with different numbers of clones.
Evolutionary adaptation of dispatching agents

Figure 7(a). Evolutionary optimization with different initial standard deviations.

Figure 7(b). Evolutionary optimization with different initial standard deviations.
Figure 8(a). Mean flow time in case of abrupt workload variation.

Figure 8(b). Total throughput in case of abrupt workload variation.
Figure 9(a). Mean flow time in case of failure/repair of a workstation.

Figure 9(b). Total throughput in case of failure/repair of a workstation.
Figure 10(a). Mean flow time in case of abrupt lack of ICs.

Figure 10(b). Total throughput in case of abrupt lack of ICs.
Evolutionary adaptation of dispatching agents

Figure 11(a). Mean flow time in case of AGV system failure.

Figure 11(b). Total throughput in case of AGV system failure.
Figure 12(a). Evolutionary versus non-adapted workstation agents.

Figure 12(b). Evolutionary versus non-adapted workstation agents.
With the exception of the first case, we simulate all these disturbances in three different workload conditions \((q = 5, 10 \text{ and } 15 \text{ respectively})\). Since, in high workload conditions \((q = 15)\), the differences among the compared policies are significantly higher than in the other cases, for the sake of brevity we will discuss this case only. Figures 8–11 show the curves of the responses of four different agent decision policies: (a) the non-adapted fuzzy Multiple Rule Criterion with \(\omega = [111] \text{ (MRC-1)}\); (b) the single static SOCT heuristic; (c) the fuzzy MRC with Lognormal evolutionary adaptation (LN-ES); and finally, (d) the fuzzy MRC with a standard evolutionary adaptation (SD-ES). If used alone, the single heuristic rules SD and SSLP are always outperformed by SOCT and, therefore, for the sake of figure clarity, the corresponding performance curves are not drawn. Figure 8 shows that, in low workload conditions, all the compared policies yield the same performance. As soon as the workload is changed, non-adapted MRC-1 tends to have a comparatively low performance, whereas the evolutionary adaptation rapidly leads the system to the best performance, with improvements ranging from the 6\% (total throughput) to the 8\% (mean flow time) with respect to the best static heuristic (SOCT). The advantages of the evolutionary adaptation are also evident in figure 9, where the SD-ES appears considerably faster in achieving a better plant operation, a condition that is also kept after workstation repair. In this case, the improvements in terms of total throughputs are approximately 10\% higher than the values corresponding to SOCT and 24\% higher than the value of the non-adapted MRC-1. We obtain similar results in the two remaining cases, as figures 10 and 11 clearly show. The partial failure of the AGV system (figure 11) is the only case in which the non-adapted MRC-1 strategy (thanks to the contribution of the SD rule) also yields a better performance than the SOCT rule. Finally, we note that both the examined adaptation strategies (namely LN-ES and SD-ED) lead to equally effective results.

Figures 12(a) and (b) compare the performance between different WA decision logics, namely a non-optimized MRC with unitary \((\text{MRC-1)}\), two evolutionary optimized MRC \((\text{SD-ES and LN-ES})\) and three static heuristics \((\text{SST, EDD and FCFS})\), whereas PAs use a static SOCT decision rule. In this case, the heuristic SST yields the best performance, while all the other decision laws provide a comparatively inferior result. According to the simulations, the adaptation of WA is considerably slower and less influential on the overall system performance than in the case of PA. The primary reason of this result is that dynamic sequencing has a minor influence on the overall system behaviour when dynamic routing flexibility is fully exploited by PAs. The fitness surface is thus flattened and the evolutionary strategy identifies too slowly the successful components of WA multi-criteria decision logic (the SST heuristic).

7. Conclusions

In this paper, we have described a MAS structure of intelligent agents, which controls the job flow within highly automated manufacturing systems. The main requisite of the control agent is its autonomy. For instance, PAs gather information on the status of the plant, make decisions and guide the job flow independently from each other. Only the flow of parts, which have to execute their working procedures indirectly, determines relations among different agents. Hence, information exchange occurs in the form of ‘horizontal communication’, i.e. without a supervisor of agents’ activity. Similar statements hold true for WAs. Integration and coordination result from the genetic adaptation mechanism, which appropriately tunes the weights of
the rules composing the multi-criteria decision logic of individual agents. In section 6 we have reported the results of a simulation study. The analysis shows that the proposed approach is very effective in facing severe abrupt disturbances and changes in the workload, failures of workstations, AGVs and shortages of part supply. The adaptation mechanism, indeed, outperforms decision logic based on conventional decision rules.

Acknowledgement
This work was supported by 40% MURST funds.

References


