International Journal of Production Research

A coordination strategy for distributed multi-agent manufacturing systems

David Naso a; Biagio Turchiano a

a Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 200-70125 Bari-Italy


To link to this article: DOI: 10.1080/0020754042000197694
URL: http://dx.doi.org/10.1080/0020754042000197694
A coordination strategy for distributed multi-agent manufacturing systems

DAVID NASO* and BIAGIO TURCHIANO

This paper proposes a distributed multi-agent approach for dynamic part routing in automated manufacturing systems. In particular, each part in the system is associated to an intelligent software agent that must select its next destination autonomously (i.e. ignoring the actions of the other agents) and in real time (i.e. at each time it completes an operation on a workstation). Differently from other existing approaches, we overcome the typical myopia of negotiation algorithms based on dispatching rules by allowing the part agents to take decisions not only about the imminent operation, but also for the subsequent ones. The anticipated decisions are transmitted to workstation agents, which are also designated to detect and resolve conflicts by modifying part agents’ decisions. To describe the single agents and their interaction schemes in a formal way, we take advantage of DEVS discrete-event modelling tools, which also allow us to develop a detailed simulation platform for our multi-agent system. The simulation experiments obtained on a detailed model of a manufacturing system derived from the literature confirm the effectiveness of the proposed approach.

1. Introduction
In the last decade, the paradigm of intelligent agents has gained a central role in research on manufacturing control. In the context of agent-based manufacturing, multi-agent systems (MASs) are networks of autonomous software components, associated to specific hardware (part agents, machine agents) or to abstract functions (mediator agents, coordinator agents), performing pre-assigned tasks in a shared physical or virtual environment. The need for distributed multi-agent solutions is also determined by the gradual shift of enterprise information systems from centralized solutions with a main processing unit to networks of low-cost personal computers. Multi-agent technologies for manufacturing aim at supporting the development of modular hardware, operating software, and highly reactive scheduling policies. The considerable potentials of multi-agent technologies have generated a very active cross-disciplinary area of research, documented by an extremely rich literature embracing topics ranging from computer to social sciences. Multi-agent technologies had a significant impact also in the area of industrial manufacturing, giving raise to virtually countless conferences, international research projects, and technical literature. For the sake of brevity, our attention will focus on contributions that are strictly related to our research, while interested readers are referred to the various books and comprehensive surveys (see e.g. Ferber 1999, Shen and Norrie 1999, G. Weiss 1999, Shen et al. 2001, to mention a few) for a further introduction.

Revision received December 2003.
Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Re David, 200–70125 Bari–Italy.
* To whom correspondence should be addressed. e-mail: naso@poliba.it
The recent research on MASs related to our paper focuses mainly on the analysis of the complex interrelationships between the dynamical entities sharing operational tasks such as production planning, part routing, job sequencing, etc. (Duffie and Prabhu 1996). Agent-to-agent coupling and interaction mechanisms determine the overall architecture of the multi-agent network, which varies between two extreme forms, defined in the technical literature as hierarchical (agents of higher level act as supervisor for those in the lower levels) and heterarchical (no master–slave relationships, each agent can operate in full autonomy) organizations.

Heterarchical architectures certainly represent the most challenging paradigm in MASs design. In such architectures, hierarchical or master–slave relationships must be avoided or at least extremely limited (Valckenaers et al. 1998), and concepts such as cooperation and competition between agents must replace supervision and command functions. In heterarchical approaches, agents are in charge of decision and control tasks in a domain where information must also be distributed among the intelligent entities as much as possible, and consequently no global database or blackboard should be available. On the one hand, at least in principle, adopting such design paradigms should improve the modularity, the reconfigurability, the scalability, and the fault tolerance of the system (Duffie and Prabhu 1996, Baker 1998). In other words, heterarchical architectures offer a solution for all the critical issues of automated manufacturing that cannot be easily dealt with using traditional solutions based on centralized or hierarchical schemes. On the other hand, the adoption of this innovative paradigm involves other equally critical problems, such as the lack of predictability and the inherent difficulty in optimizing the global performance of the collective of asynchronous, autonomous, and concurrent agents.

The lack of generality and predictability of the global performance of heterarchical control systems also depends on the specific algorithms used by agents to organize their tasks. As pointed out in the recent survey (Baker 1998), most researches in the context of MASs adopt schemes of interaction between agents based on the metaphor of negotiations in micro-economic environments. In general terms, negotiation algorithms can be regarded as task-dispatching strategies using fictitious currency and heuristic pricing policies based on the current conditions of the tasks and the servers involved in the decision. Since negotiation algorithms are often based on dispatching rules, they inherit the typical myopia of such simple heuristics, which only use local information available at decision time. As a main consequence, negotiation algorithms often fail to provide the desired improvements of performance robustness and predictability. In fact, this criticism is supported by several comparative analyses (e.g. Van Brussel et al. 1998, Maione and Naso 2003a) showing that real-time negotiation approaches are not easy to configure, and often exhibit context-dependent performance.

This paper proposes a new multi-agent control approach that contributes to the resolution of these critical problems of heterarchical manufacturing, providing a simple but adequate solution to the emergent need of architectures where decision and control tasks are fully distributed, and at the same time satisfactory levels of global performance are guaranteed. In particular, this paper focuses on shop floor control, although the proposed ideas can be easily extended to other levels of factory operation. We describe a fully heterarchical manufacturing shop floor control system based on a network of distributed and autonomous agents. Similarly to other MASs (Lin and Solberg 1992, Maione and Naso 2003b, Heragu et al. 2002), part agents interact with resource agents (associated to workstations, loading and material
handling devices) to schedule processing or transfer tasks in real-time, i.e. as soon as the part is ready for the requested task. In particular, we focus on a part-driven scheduling approach in which part agents are in charge to autonomously select the best workstations for each operation (i.e. to control the routing of the associated physical part in process), while workstation and material handling agents are in charge of appropriately sequencing the processing and transfer operations requested by the parts.

The first important contribution of the proposed approach is that, instead of using the conventional negotiation algorithms based on single decision parameters, our agents use a decision strategy that takes into account multiple indices of the real-time operating condition of the shop floor by means of fuzzy logic. The fuzzy decision algorithm allows us to improve the agent’s ability to determine effective compromise solutions in a short decision time. With the aid of a detailed simulation comparison, we show that the proposed multi-criteria algorithm always outperforms other decision strategies widely adopted in negotiation algorithms.

On the other hand, as in all heterarchical organizations, the autonomous decisions often lead part agents to interfere with each other, e.g. to compete for the same resource neglecting other more profitable alternatives. The fuzzy multi-criteria approach does not overcome these undesirable circumstances determined by the inherently short-term evaluations of agent’s decision algorithms, which only take into account local information at the decision time. It must also be remarked that longer-term effects of local decision cannot be easily predicted due to the complexity of the discrete-event dynamics of any manufacturing system. The second main contribution of this paper addresses this specific problem. Following the recent research trends on hybrid systems (Heragu 2002), we propose a heuristic coordination mechanism that overcomes the lack of global and long-term view without compromising the desirable features of heterarchical control. The key idea consists in letting the workstation agents temporarily assume the role of supervisor when they detect a potential conflict between part agents. In such a case, workstation agents use a multi-objective decision algorithm similar to that of part agents. However, in this case the considered alternatives are evaluated on a longer-term scale, in order to identify and overcome all the possible interferences between part agents. Therefore, if necessary, a workstation agent can force parts to reconsider their decision to achieve a better overall compromise. This hierarchical relationship between workstations and parts only occurs in specific circumstances and for a limited time. As confirmed by the simulation investigation described in this paper, the proposed schema does not compromise the benefits of the overall heterarchical structure, while it allows us to significantly reduce the rate of critical circumstances, such as machine blocking, and to obtain a further improvement of global production indices such as throughput and resource utilization.

Finally, we must remark that the lack of sufficient details about the actual structure of the agents, their decision algorithms and interaction protocols used in most of the literature constitutes a further source of difficulty in evaluating the relative effectiveness of different multi-agent approaches (Cavaleri et al. 2003). In order to obtain a complete and unambiguous characterization of the operation of our MAS, we model each agent using the Discrete EVent System (DEVS) specification (Zeigler et al. 2000). The DEVS formalism allows us to describe agents as atomic dynamic systems, subject to external inputs from other agents and generating outputs directed to other agents. Finally, we obtain the global model of the entire network of agents.
by specifying the interrelationships between the atomic components (the agents). It is important to remark that the development of adequate modelling tools for multi-agent networks is a key issue involving all the application areas of multi-agent technologies, including, e.g., telecommunication networks, computer networks, communities of intelligent robots, web-based agents for information retrieval, to mention a few. Recently, considerable research efforts have been devoted to the definition of standards (FIPA 2003) and development platforms for unambiguous agent specification, especially in the context of software engineering. Two recent examples of this research are the Agent UML (AUML 2003) and the Java Agent DEvelopment (JADE 2003) projects. In the narrower context of industrial manufacturing, the DEVS formalism can be viewed as an interesting alternative to other recently proposed tools for MAS specification [e.g., the UML (Huhns and Stephens 2001) Petri nets (Lin and Norrie 2001), to mention two], because it allows us to develop models that are suitable both for discrete event simulation, and for final implementation of the software controllers on the plant’s operating system.

The reminder of this paper is organized as follows. Section 2 surveys the related literature. Section 3 describes the model of part agents and the fuzzy logic-based decision algorithm. Section 4 describes the other agents in the systems, illustrating the details of the novel coordination algorithm used by workstation agents to improve the global performance. Section 5 describes the case study used as a benchmark to validate and compare the proposed strategies with other well-known distributed approaches. Section 6 concludes the paper with some final remarks.

2. Related literature

The typical way to deal with the need of predictability and performance optimization in highly distributed control systems is to devise specific crossovers between hierarchical and heterarchical structures, in order to combine the desirable properties stemming from both paradigms. For example, in holonic manufacturing systems (Valckenaers et al. 1998), a holon is both a self-contained agent and a part of another larger entity/holon. Holons are organized in a heterarchical structure, but they can temporarily form a local hierarchy to solve specific tasks. PROSA (Product, Resource, Order, Staff holons Architecture, Van Brussel et al. 1998) is a highly distributed control architecture basically consisting of three types of holons: the product holon retaining the process knowledge for manufacturing products; the resource holon controlling the workstations, and processing the associated information; and the order holon responsible for performing the assigned tasks correctly and in time. A fourth holon, the staff holon, can also be introduced in the system as the controller determining the transition of groups of holons from hierarchical (when the plant is running in stationary nominal conditions) to heterarchical (when the system is running under severe disturbances) organizations and vice versa.

Similarly, the MetaMorph architecture (Maturana and Norrie 1997, Maturana et al. 1999) introduces the so-called mediators, i.e., special agents playing the role of coordinators by ‘promoting cooperation among intelligent agents and learning from agents behaviour’. In the manufacturing plant, when an agent makes a request, a mediator selects the most suitable receptors and broadcasts the request to them. Once it has forwarded the message, the mediator lets the calling agent and the serving receptors interact autonomously. Mediators retrieve and reuse information on the past actions by applying a technique called distributed case-based learning, and use simulation to perform a ‘look-ahead’ inspection of performance.
Recently Heragu et al. (2002) proposed a new modelling framework for hybrid architectures in which part agents, machine agents, and material handling agents represent the lower-level controllers. On a higher level, cell (or department) agents are software entities that coordinate the actions of the machine agents and material handling agents located in each department, while at the top of the organization a single system agent controls all the cell agents in the warehouse. Decisions are made at each of the three levels: the interaction between agents on the same level is ruled by a negotiation algorithm, while the higher-level agents supervise the lower-level ones by specifying or altering their decisions, objectives, or constraints. The task scheduling on the shop floor is based on negotiation algorithms similar to those used also in Van Brussel et al. (1998) and Maturana et al. (1999). However, agents residing on the higher levels, which have to explicitly allow lower-level agents to commit to their contract, always supervise each decision at a lower level. The paper by Heragu et al. (2002) also provides an extensive survey of hybrid hierarchical/heterarchical approaches for multi-agent manufacturing control.

Another emerging issue in heterarchical manufacturing control is related to the selection of the appropriate processor where each agent is hosted and executed among the various devices composing the control network of the manufacturing plant. Ideally, when agents are associated to mobile physical entities in the manufacturing facility (e.g. parts, AGVs), they should be able to move from one computing device to another so as to follow their physical counterparts along their path across the facility. In other cases, a mobile agent could be used for monitoring (Szirbik et al. 1999), fault-detection, or coordination purposes (Fletcher et al. 2001). Therefore, in the most general case, a heterarchical network of agents in a manufacturing facility may encompass permanent agents [i.e. always running on the same computing device, and therefore often referred to as static agents in the literature, e.g. Huhns and Singh (1997)] and mobile agents (i.e. software entities moving across the various computing hosts, or running on a mobile computing device). When dealing with mobile agents, several critical issues must be addressed in a proper way. In particular, in the context of relatively small-scale computing networks as those generally controlling manufacturing facilities, issues such as fault-tolerant execution (Pleish and Schiper 2003), safe migration protocols, mechanisms to locate a mobile agent (Pham and Kaemuch 1998), and more generally adequate coordination and synchronization strategies play a fundamental role. It should be underlined that the critical issues related to mobile agents within a closed environment such as manufacturing facilities are only a small subset of the various challenging issues arising in the most general case of a mobile agent running on large-scale networks (e.g. the internet). The latter include protection, secrecy, and availability of bandwidth, to mention just some [see, e.g. the survey by Pham and Karmuch (1998) for further details]. We must also remark that the main aim of this paper is to describe some significant improvements of agent decision mechanisms, and inter-agent coordination protocols. For this reason, to date, the simulation model of the manufacturing facility and the associated control network run on a single host, so the technical issues related to agent’s mobility are not addressed here, but represent one of the open topics for future research.

As mentioned in the previous section, most research on MASs for manufacturing control [see e.g. the surveys by Baker (1998) and Shen and Norrie (1999)] adopts schemes of interaction between the agents based on the metaphor of negotiation. These strategies attempt to emulate the ability of intelligent decision makers to find
rewarding tradeoffs between conflicting criteria in extremely uncertain and dynamic contexts. An increasing number of researchers has recently acknowledged the affinity of these approaches with other powerful problem-solving techniques inspired to human reasoning and other biological systems. In particular, the potential synergies stemming from the combination of computational intelligence (e.g. fuzzy logic, genetic algorithms) and distributed artificial intelligence (agents) have often been underlined in recent literature (e.g. Lakov 2002, Maione and Naso 2003a). Most research deals with communities of intelligent robots (e.g. Hagras et al. 2000, Bonarini 2001, Jacak et al. 2001), even though some early results in the context of manufacturing control have also been proposed (Aliev et al. 1997, Flake et al. 1999, Ulieru and Norrie 1999–2000, Ulieru et al. 2000). A recent survey of this evolving research area can be found in Naso and Maione (2001).

3. Part agents

In flexible manufacturing environments, the production floor consists of a set of multi-purpose workstations that can perform various operations. In general, the same operation can be executed by multiple workstations, so that parts can choose different paths (sequences of workstations) to obtain the requested services. In such manufacturing environments, where both processing sequences and workstations are extremely flexible, the global performance is mainly related to the effectiveness of part routing, i.e. the strategy assigning the parts to the workstations that perform the requested operations. The typical rate of unforeseen perturbations (faults, urgent jobs) occurring in an automated shop floor often suggests the implementation of real-time decision algorithms. Such a context is the typical environment where distributed multi-agent solutions are extremely desirable. Similarly to related literature (Lin and Solberg 1992, Duffie and Prabhu 1996, Heragu et al. 2002, Maione and Naso 2003a), in our real-time dispatching approach each part to be processed is associated to a mobile agent (the Part Agent, PA) that identifies in real-time, i.e. shortly before the part is ready for a new process, the most suitable workstation for the imminent operation. The selection is based on updated information directly obtained from the alternative workstations in real-time, through an exchange of messages with the Workstation Agents (WA), which are software entities also designated to provide PAs with the necessary information for their decisions. Other specific agents (the Loading Agent, LA, the Material Handling Agents, MHA) handle part loading and transfer between workstations. All the considered agents are software programs continuously running on one of the host processors in the network of computers controlling the manufacturing plant. The agents are executable programs realized by translating their discrete event models into executable files using commercial software (C++). Thus, the most straightforward and unambiguous way to illustrate the tasks and internal structure of each agent is directly specifying its discrete event model, while all the technical issues related to agent programming and execution will be omitted for sake of brevity. When executed, each software entity is always ready to receive external inputs, selecting the most appropriate action among a set of predefined ones. Each agent interacts with the other agents and the environment sending and receiving messages, and executing commands to the hardware it is in charge to control. Consequently, the global control of the manufacturing floor emerges from the concurrent actions of the various agents in the system. The following subsections describe the discrete event
Distributed multi-agent manufacturing systems

model of a PA, and the decision algorithm used to select the best workstation for each production step.

3.1. Discrete event model of PAs

A new PA is initialized whenever a new raw part enters the manufacturing system. Once created, the PA is in a passive state (Ready) and waits for external signals to start its task. The PA is activated by the LA with an initialization signal when the loading of the associated raw part is completed. When the PA is activated, it starts requesting information to the available workstations to identify the best server for the first operation (Querying). Once the PA ends sending the requests, it starts waiting for replies from the WAs (WaitingOffers). This condition ends either when all the messages are received, or when a timeout expires. Timeouts are introduced to ensure that indefinite waits are avoided. If the PA does not receive any reply within timeout, it returns in the Querying condition. If one or multiple replies are received, the PA starts processing the replies to identify the best destination (Selecting), using the multi-criteria algorithm described in the next subsection. At the end of the decision, the PA ranks the available alternatives in order of preference, and selects the first workstation as potential destination. Then, the PA sends a booking request to the WA, and waits for confirmation. If the request is accepted, the PA sends a transfer request to the material handling system (RequestingTransfer), otherwise it sends a booking request to the second WA in order of preference, and repeats the same sequence of operations until it obtains a confirmation. If the PA does not obtain a confirmation even from the least preferred WA in the decision rank, it returns in the Querying condition. The overall sequence of operations performed by a PA is summarized in figure 1, which specifies the sequential states and the events triggering the transitions from a state to another one (separately described in table 1). DEVS models distinguish two main types of events triggering the transitions between consecutive sequential states, namely, external and internal events (Zeigler et al. 2000). For a given agent, an event is external if its occurrence is not determined or directly influenced by the agent but it depends on exogenous factors, such as the actions of other agents. The typical external event for a PA is the receipt of a message from another agent. On the contrary, all the events whose occurrence is determined by agent’s internal dynamics are classified as internal. The occurrence of internal events is triggered by internal factors (e.g. an internal timing function), which are explicitly under the control of the agent. Typical internal events for a PA are the completion of a task (processing a decision, sending a message) or the expiration of a timeout.

Until this point, the PAs follow a schema that is similar to other models used in related research (Maione and Naso 2002, 2003b). However, the PAs proposed in this paper are significantly improved with respect to the previous versions since they attempt to anticipate their next decisions by looking ahead in their processing sequence as follows. Once the PA receives the acknowledgement of a transfer request, it does not return in the Ready condition. Rather, it restarts a new decision cycle (AdvQuering, AdvWaitingOffers, AdvSelecting) to determine in advance the best available workstation for the second operation in the process plan. After completing the anticipated decision, the PA sends a message (Advance Booking Request, ABR) to the WA of the selected workstation and waits for an acknowledgement of receipt. Finally, when the confirmation is received, the PA returns in the Ready condition. The PA performs an advance selection with the same decision criteria used in the
Selecting condition, but also taking into account the delay between the advance decision and the actual time at which they are executed (we must recall that the transfer to the first machine and the operation on the first machine still have to be performed at this time). The estimation of the effects of this delay, and of the future conditions of the workstations, is made with heuristic projections based on currently available data. Clearly, the correctness of these estimates cannot be guaranteed, and this advance decision mechanism can lead to erroneous choices. On the other hand, the mechanism of anticipated decisions allows us to obtain larger intervals of time in which these decisions can be revised to handle possible conflicts, and improve the overall results. We will resume the discussion on this aspect in the next section.

Once the PA returns in the Ready condition, it remains idle (i.e. it does not play any active role in the progress of the manufacturing process) until reactivated by a message from the WA announcing either the end of the current operation, or a workstation failure imposing the search for a new server. At this time, the PA has to send a new transfer request and enters again in the Requesting Transfer condition. From now on, the PA will repeat the advance querying-decision-transfer cycles until the associated part receives all the requested operations and is transferred to the unloading station to exit the system. As will be shown in the simulated case study, in general, the mechanism of anticipated decision leads to better performance indices, even if the PAs are not coordinated between each other. Considerably higher performance improvements can be achieved when the anticipated decisions are combined with the cooperation algorithm described in the next section.

3.2. Part agent decision algorithm

As previously remarked, the recent literature observes that the performance of real-time negotiation algorithms often adopted in heterarchical approaches are
strongly dependent on the context of the application, and on their configuration parameters. To improve the performance of agent decision and interaction schemes, this paper adopts a solution based on fuzzy logic. Similarly to the fully heterarchical approaches proposed in Duffie and Prabhu (1996), each PA performs its routing decisions independently and unilaterally using a simple and effective multi-criteria decision algorithm without interacting explicitly with each other to avoid conflicts. In fact, the high number of parts simultaneously circulating in a flexible manufacturing system imposes the design of relatively simple part agents. Furthermore, direct PA-to-PA cooperation schemes are not feasible for the excessive growth of computation and communication requirements with the number of parts in the system. The assumption of no explicit interaction between PAs overcomes these limitations, and also guarantees, at least in principle, an improved scalability of the heterarchical control system (Duffie and Prabhu 1996).

The decision algorithm used by our PAs is based on a fuzzy multi-criteria decision making approach (Zimmermann 1993). Such algorithms have been used in similar contexts such as flowshop scheduling (Fanti et al. 1998), part routing (Naso and Turchiano 1998), and adaptive agents (Maione and Naso 2001). The proposed algorithm offers the same transparency of conventional fuzzy rule-based controllers proposed in some of the reviewed literature (Aliiev et al. 1997, Flake et al. 1999, Ulieru and Norrie 1999–2000, Ulieru et al. 2000), at a reduced computational effort. In fact, fuzzy techniques represent an effective tool to build intelligent agents

<table>
<thead>
<tr>
<th>Internal events</th>
<th>External events</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i_{e0})</td>
<td>Querying completed</td>
</tr>
<tr>
<td>(i_{e1})</td>
<td>Time-out expired: no offer received</td>
</tr>
<tr>
<td>(i_{e2})</td>
<td>Time-out expired: at least one offer received</td>
</tr>
<tr>
<td>(i_{e3})</td>
<td>Selection completed</td>
</tr>
<tr>
<td>(i_{e4})</td>
<td>Confirmation Request Sent</td>
</tr>
<tr>
<td>(i_{e5})</td>
<td>Time-out expired: no confirmation received</td>
</tr>
<tr>
<td>(i_{e6})</td>
<td>Time-out expired: no confirmation received from the last workstation in decision rank</td>
</tr>
<tr>
<td>(i_{e7})</td>
<td>Transfer request sent</td>
</tr>
<tr>
<td>(i_{e8})</td>
<td>Time-out expired: no transfer confirmation received</td>
</tr>
<tr>
<td>(i_{e9})</td>
<td>(last) advance offer sent</td>
</tr>
<tr>
<td>(i_{e10})</td>
<td>Time-out expired: no advance offer received</td>
</tr>
<tr>
<td>(i_{e11})</td>
<td>Time-out expires: at least one advance offer received</td>
</tr>
<tr>
<td>(i_{e12})</td>
<td>Advance selection completed</td>
</tr>
<tr>
<td>(i_{e13})</td>
<td>ABR Sent</td>
</tr>
<tr>
<td>(i_{e14})</td>
<td>Time-out expires: no ABR ack. received</td>
</tr>
<tr>
<td>(x_{e0})</td>
<td>Activation Signal received [Loading (almost) Completed]</td>
</tr>
<tr>
<td>(x_{e1})</td>
<td>Last of all offer(s) received</td>
</tr>
<tr>
<td>(x_{e2})</td>
<td>Confirmation Received</td>
</tr>
<tr>
<td>(x_{e3})</td>
<td>Transfer Confirmed</td>
</tr>
<tr>
<td>(x_{e4})</td>
<td>All Anticipated offer(s) received</td>
</tr>
<tr>
<td>(x_{e5})</td>
<td>ABR Acknowledgement received</td>
</tr>
<tr>
<td>(x_{e6})</td>
<td>Current operation (almost) completed</td>
</tr>
<tr>
<td>(x_{e7})</td>
<td>Warning about a fault occurred to current workstation received</td>
</tr>
</tbody>
</table>

Table 1. List of internal and external events for a PA.
that make qualitative decisions based on multiple criteria with relatively simple algorithms. Our algorithm takes into account different descriptive parameters of alternative servers (measured at the decision time) such as the workload in terms of processing time, the distance of each of the alternative machines, and the current setup configuration of the workstation. Multiple attributes allow us to encompass a wide number of different aspects in the decision policy, ranging from deadlock avoidance to the minimization of material handling and machine setup delays, from the maximization of resource utilization to workload balancing.

The decision algorithm can be described as follows. Let \( W = \{W_1, W_2, \ldots, W_n\} \) indicate the set of alternative workstations for the imminent (in case of the first operation) or next (in case of advance selection) operation on a raw part. If we use \( m \) different criteria for ranking the \( n \) alternatives, we can easily organize decision parameters in an \( m \times n \) matrix, indicated as the decision matrix. Our algorithm converts descriptive variables defined on real-valued intervals (e.g. waiting time for preceding workload, earliness/tardiness, number of jobs in queue) in numbers in the interval \([0, 1]\). Depending on the type of decision parameter, the conversion procedure is based either on static membership functions or on a normalization algorithm, such as those described in Fanti et al. (1998) and Naso and Turchiano (1998). Therefore, each element \( s_{ij} \) of the decision matrix represents the fuzzy membership degree expressing how much alternative \( j \) satisfies decision criterion \( i \). The decision process is executed in two consecutive steps:

1. The aggregation of the \( m \) criteria in a unique measure. This step is performed by the fuzzy intersection of the \( m \) fuzzy sets representing the satisfaction of the \( m \) single criteria, i.e.

\[
\bar{s}_j = s_{1j}^w \ast s_{2j}^w \ast \cdots \ast s_{mj}^w \quad j = 1, \ldots, n
\]

where the symbol \( \ast \) indicates a generic fuzzy t-norm (e.g. the minimum or the product) (Zimmermann 1993) and \( w_i (i = 1, \ldots, m) \) is a weighting factor grading the influence of the \( i \)-th attribute in the computation of the global desirability.

2. The ranking of the alternatives with respect to the global criterion, and the selection of the best alternative (the one with the maximum satisfaction in the global criterion):

\[
\bar{s}_h = \max_{j=1,\ldots,n} \left( \bar{s}_j \right) \quad h \in \{1, 2, \ldots, n\}
\]

Example

A simple example can be useful to give a clear idea of the basic mechanisms of the decision. Let us consider that a PA at a given time has to choose the next workstation among four available alternatives (\( n = 4 \)) \( W_1, \ldots, W_4 \), using three criteria (\( m = 3 \)), namely, the distance, the expected waiting time for the operation, and the residual available buffer space. The parameters of the four workstations at the decision time are summarized in table 2. For brevity, each cell of the table contains three values \( a_{ij}/b_{ij}/c_{ij} \), where \( a_{ij} \) is the current value of the parameter, \( b_{ij} \) its fuzzified value, and \( c_{ij} \) the weighted value. For purposes of the example, the fuzzification is simply obtained with a standard normalization \( b_{ij} = \min_k (a_{ik})/a_{ij} \), for the first two
criteria, while \( b_{ij} = a_{ij} / \max_k (a_{ik}) \) for the third one, the \( t \)-norm operator for the aggregation of the fuzzified values is the minimum, and the weights are applied as exponential factors. Neglecting the weights, the algorithm would choose \( W_1 \), and then \( W_4, W_2 \) and \( W_3 \), in order of decreasing satisfaction. Taking into account the rule weights, which assign a higher influence to the residual buffer space, the final ranking is changed to \( W_4, W_1, W_3, W_2 \).

Clearly, the algorithm is computationally simple and effective. Furthermore, by varying the weights associated with each heuristic decision rule, it is possible to obtain a virtually infinite number of different multi-criteria routing policies. Several approaches have recently been proposed to optimize decision weights in analogous contexts. A first possibility yielding a good trade-off of simplicity and performance is to use a small set of decision rules, and select fixed rule-weights for a given production goal. In this case, a grid-inspection method (see, e.g. Hwang and Sang 1998) often provides satisfactory results. The method consists of simulating and comparing different decision policies obtained using weights from a uniformly distributed grid of samples. In this case, the collected results will lend themselves to statistical analyses, and to multi-objective optimization (Naso and Turchiano 1998). A different approach based on evolutionary computation can also be used to develop adaptive policies in which the agents change their decision weights during the production process. This method experiments continuously with new weights, and uses the feedback on their actual performance to compute new decision weights. As remarked by Maione and Naso (2003b), this strategy is significantly heavier in terms of implementation burden, but can offer remarkable improvements in both steady-state performance and system reactivity to unexpected faults.

### 4. Resource agents

Specific agents (WAs, LA and MHAs) control the hardware providing services to processed parts (workstations, loading and material handling devices, respectively). The resource agents are designed and characterized using the same modelling procedure adopted for PAs. We identify the sequential states, the internal and external events triggering the transitions from a given state to the next one. The next subsection focuses on WAs, and especially on the new mechanism of cooperation proposed in this paper. The following subsection describes the models of the LA and MHAs. Since the tasks executed by these agents do not influence system performance in a significant way, their models are only overviewed.

<table>
<thead>
<tr>
<th>Weight</th>
<th>( W_1 )</th>
<th>( W_2 )</th>
<th>( W_3 )</th>
<th>( W_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance (m)</td>
<td>1</td>
<td>80/0.63/0.63</td>
<td>100/0.50/0.50</td>
<td>50/1.00/1.00</td>
</tr>
<tr>
<td>Expected waiting time (mins)</td>
<td>1</td>
<td>3/1.00/1.00</td>
<td>15/0.20/0.20</td>
<td>20/0.15/0.15</td>
</tr>
<tr>
<td>Residual buffer space (units)</td>
<td>2</td>
<td>2/0.66/0.44</td>
<td>1/0.33/0.11</td>
<td>2/0.66/0.44</td>
</tr>
<tr>
<td>Fuzzy Aggregation Neglecting / Considering weights</td>
<td>0.63/0.44</td>
<td>0.20/0.11</td>
<td>0.15/0.15</td>
<td>0.50/0.50</td>
</tr>
</tbody>
</table>

Table 2. An example of decision matrix—best alternative in boldface.
4.1. Workstation agents

A WA is a software entity controlling a workstation. To accomplish this task, the WA has to perform several functions, e.g. processing and replying to the queries from PAs, assigning priorities to the tasks waiting in the queue, etc. If the WA is busy, the messages received from PAs are stored in different queues according to their types (offer request, confirmation request, advance booking, etc.) and served with a FIFO discipline, except for the fact that confirmation requests have higher priorities than advanced booking or offer requests. In addition to accepting/refusing new requests, the WA must assign the processing priority to parts hosted in the buffer and waiting for an operation. In this paper, this function is performed with a standard earliest due date (EDD) dispatching policy. In fact, related research (Maione and Naso 2001) already showed that more complex sequencing strategies do not yield significant improvements on system performance when flexible routing is allowed. Furthermore, the WA has to broadcast special alerts when the associated workstation fails, to let the PAs in queue at (or directed to) the workstation find an alternative server. The complete DEVS structure of a WA performing the above-specified functions was already described in related research (Maione and Naso 2002). For sake of brevity, this section only describes the new coordination function assigned to WAs, which distinguishes the agents used in this paper from those available in related references.

The main advantage of the distributed multi-agent routing is that each PA performs a decision on the basis of the current status of each potential server. In this way, the resulting dynamic reallocation of parts on the available workstations ensures a high reactivity to unexpected events such as faults and consequent bottlenecks. Even if the PAs attempt to expand their decision horizon by anticipating the decision about their next operation, their decision algorithms are inherently myopic since they cannot take into account any possible effect of the decisions on longer time intervals. In other words, even if a server appears to be the best choice according to the locally (in time and space) available information, a different decision could be more profitable on a longer time scale. For this reason, in this paper we assign to the WAs the further task of monitoring and coordinating the action of PAs to achieve an improved global performance of the MAS.

To accomplish these new functions, the WAs must firstly detect potential conflicts between the PAs. The term conflict refers to the circumstance in which two or more PAs almost simultaneously select the same workstation for the next operation, ignoring each other’s choice. As discussed in the previous section, it is difficult to handle this type of circumstance by adding specific PA-to-PA synchronization algorithms. On the contrary, the conflicts for the same workstation can be easily detected by the WA by means of the ABRs. As already stated in section 3.1, the ABR is a message sent by a PA to a WA to communicate in advance a decision about the next planned operation. Figure 2 illustrates the structure of an ABR: it contains an ABR identification string and a timing information string, which specifies the interval of time in which the advance routing decision can be revised. Namely, a routing decision can be revised in the time interval ranging from the time at which the ABR for the next operation is issued (issuing time $t_1$ in figure 2) until the expected timeout (expiration time $t_x$) estimating the time at which the PA will end the current operation and will initialize the transfer toward the next server. The time $t_x$ is computed by the PA based on the estimated processing times (contained in WA offers) and transfer times (computed according to current location).
The revision intervals \([t_i, t_s]\) are used by WAs to optimize the decisions of PAs competing for the same destination. When a WA receives two or more overlapping ABRs from different PAs, it starts to operate as coordinator. Let \(t_r(ABR_{PA_h})\) (reception time) indicate the time at which the WA receives an ABR from the \(h\)-th PA in the system \((t_r(ABR_{PA_h}) > t_x(ABR_{PA_h})\) due to communication delays): a first ABR from the \(h\)-th PA is overlapping with a second ABR from the \(j\)-th PA if the latter is received before the expiration time of the first, i.e. if

\[
t_r(ABR_{PA_j}) < t_x(ABR_{PA_h}).
\]

The ABR overlap is also described in figure 3, where, e.g. \(ABR_{PA1}\) and \(ABR_{PA2}\) are overlapping, whereas \(ABR_{PA1}\) and \(ABR_{PA4}\) are non-overlapping. Clearly, the width of the revision intervals determines the average number of overlapping ABRs received by WAs. Thus, the revision interval can also be regarded as a tuneable parameter to achieve the desired trade-off of cooperation/autonomy.

A coordination action is a unilateral decision taken by a WA about rerouting PAs with overlapping ABRs toward other workstations. This is accomplished in the following way. Suppose that at a given time, a WA has received \(p\) overlapping ABRs
from \( p \) different PAs (PA_1, PA_2, \ldots, PA_p). The latest time for revising one or more of these requests is
\[
t_{\tilde{x}} = \min_{h=1, \ldots, p} (t_x(A_{BP_{PA_h}})).
\] (4)

Shortly before time \( t_{\tilde{x}} \) (to take into account the transmission delays, and reach all the PA in time for revision) the WA runs the coordination algorithm, which is another fuzzy multi-criteria algorithm similar to the one used by PAs. The coordination algorithm uses the same decision rules of PAs, while the main difference between the two fuzzy algorithms lies in the type and number of evaluated alternatives. As already stated in section 3.2, the set of alternatives for the PA algorithm is the set of servers that replied to the request for a given operation. For what concerns the algorithm used by WAs, indicating by \( P = \{PA_1, PA_2, \ldots, PA_p\} \) the set of conflicting PAs, and with \( M \) the set of all the workstations in the system, let \( M_1 \subseteq M \) be the subset of alternative workstations for \( PA_1 \), \( M_2 \subseteq M \) the subset of alternative workstations for \( PA_2 \), and so on. The set of available alternatives for the WA is the Cartesian product \( M_1 \times M_2 \times \cdots \times M_p \), and a decision \( d_{WA} \) is a set of routing assignments:

\[
d_{WA} \in \{(PA_1, m_1), (PA_2, m_2), \ldots, (PA_p, m_p)\}, \quad m_1 \in M_1, m_2 \in M_2, \ldots, m_p \in M_p. \] (5)

In other words, the WA reformulates the routing decision about currently conflicting parts from a global viewpoint, avoiding the typical drawbacks deriving from the local decisions of autonomous PAs, such as bottleneck, machine blocking or starvation. The expanded decision process requires that the WA exchanges in real-time updated information with all the other WAs associated to the workstations involved in the decision, using querying-answering mechanisms similar to those described in the PA model. Therefore, this process may also determine a considerably higher load for the communication network, especially in flexible manufacturing environments where a large number of alternative workstations may be available for each operation. Clearly, the actual cost/benefit ratio of this schema cannot be generalized for all the types of manufacturing systems, but it must also be observed that the width of the revision interval can always be adjusted to keep the amount of supervision/cooperation actions within levels that are compatible with the capacity of data communication and processing resources.

After a coordination decision is completed, the revised routes are transmitted to all the involved PAs. To limit computational and communication requirements, once determined, a route reassignment can no longer be modified by the PA or by other WAs (e.g. taking another coordination decision). Since all the involved PAs must accept the revised route (unless they do not receive the rerouting decision within the expiration time \( t_x \) due to communication delays), the intervention of the WA as coordinator determines, in fact, a temporary and local hierarchy. However, we must also recall that the WAs assume their supervisory function only when a potential conflict is detected. In other words, differently from truly hierarchical organizations, here the PAs do not explicitly rely on the intervention of the WAs, and can operate autonomously even when the coordination function is disabled. In this way, if the temporary supervision is unavailable (e.g. for local failures or network congestions), the autonomous PAs can continue to work properly, preserving the improved fault tolerance typical of heterarchical organizations.
4.2. Loading and material handling agents

The LA rules part loading and unloading, taking into account the current work in progress (i.e. workstation workloads), and the production demand. In this paper, we focus on ‘closed’ manufacturing systems in which, once the maximum work in process is reached, a part can enter only when a finished one is unloaded. Thus, the main task of the LA is to select which part type has to be loaded when a finished one exits the system. An extensive literature is available to deal with the part loading problem in flexible manufacturing environments, whereas such aspects are often neglected or marginally treated in the literature about MASs, which is mainly focused on the interaction between part and workstation agents. Thus, the lack of sufficient details about loading policies makes it difficult to analyse the role of loading agents in related research.

Due to the closed system assumption, in this paper the LA simply uses the Highest Ratio of Remaining Requirement (HRRR) loading heuristic (Ro and Kim 1990), which selects the part type having the highest ratio of raw parts still to be produced with respect to the total requirement of the current production cycle. The LA also sends activation messages to the new PAs entering the system, and can interact with WAs when unexpected failures or congestions make it necessary to modify the loading rate.

In our system, material handling operations are executed by a set of independent AGVs. Each AGV has an associate MHA in charge of processing transfer requests. Differently from WAs, MHAs use a centralized blackboard dispatching system: all the transfer requests are queued in a single buffer and served with a FIFO discipline. The transfer acknowledgement is sent to a PA as soon as its transfer request is received and stored in a queue. Whenever an AGV ends a transfer operation, the associated MHA takes charge of the first request in the queue. The extension of the heterarchical organization to the material handling system, also taking into account multiple attributes such as distances and transfer priorities, is one of the main objectives of current research.

Figure 4 summarizes the overall structure of the proposed MAS. All the agents work on the same hierarchical level, interacting with each other through exchanges of requests and replies. The figure also throws light on the special role of WAs, which can temporarily assume a command function with respect to PAs in the case of conflict detection.

Figure 4. The hybrid network of agents: the coordination function temporarily brings WAs on a higher level.
5. Simulations

Our case study is based on a model of a fully automated manufacturing system derived from the literature (Ro and Kim 1990, Naso and Turchiano 1998). This model was explicitly chosen because it is detailed enough to allow us to perform realistic simulations of a real-world scenario, considering fundamental issues such as setup, transfer and loading/unloading times, and limited pallet availability. As shown in figure 5, the system has four CNC machining centres with finite buffer capacity (three parts), a load/unload station, and a material handling system with two AGVs carrying one pallet at a time. The system produces three different part types, and each part requires four operations that can be executed on alternative machines. Table 3 lists the alternative routes and the corresponding processing times for each part type. The DEVS models of agents are implemented using the C++ programming language, and integrated within the shop floor model developed using the SIMAN/ARENA discrete event simulation environment.

In the first batch of simulations, we compare the performance of our approach with some common distributed routing approaches based on single decision criteria. The benchmark consists of a production cycle of 300 parts (100 for each part type), and assumes that all the raw material is available at the beginning of the production process. We consider the following routing approaches:

- **Minimum Work in Queue** (MWQ, choose the machine with the minimum workload in terms of processing time),
- **Shortest Processing Time** (SPT, choose the fastest machine to perform the operation),
- **Shortest Distance** (SD, choose the closest machine) and
- the policy **Alternative Routings Dynamically Directed** (ARDD, choose the destination with the minimum sum of transport time, waiting time and processing time (Ro and Kim 1990)).

The agent’s Multi-Criteria Decision (MCD) is based on four criteria: MWQ, SPT, SD and **Deadlock Avoidance** (DA, avoid machines blocked by other parts waiting for overloaded destinations (Naso and Turchiano 1998)). Clearly, the latter heuristic aims at decreasing the risk of blocking and deadlock occurrence, but does not guarantee full deadlock avoidance. It attempts to avoid mutual waits assigning a low desirability to blocked machines and maximum desirability to all the other
<table>
<thead>
<tr>
<th>Part Type</th>
<th>No. of alternative routes</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Machines</td>
<td>1st</td>
<td>1(4)</td>
<td>2(6)</td>
<td>2(6)</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
<td>3(7)</td>
<td>2(6)</td>
<td>2(6)</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>1(3)</td>
<td>1(3)</td>
<td>1(3)</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>2(4)</td>
<td>2(4)</td>
<td>2(4)</td>
</tr>
<tr>
<td>Total route time (min)</td>
<td>2</td>
<td>2.2</td>
<td>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td>Max. Total Proc. Time (min)</td>
<td>18</td>
<td>20</td>
<td>20.2</td>
<td>21.3</td>
</tr>
</tbody>
</table>

Part on pallet loading/unloading time = 3 min  
Pallet on AGV loading/unloading time = 0.8 min

Table 3. The stations (processing times in parentheses) and alternative routes (in columns) for part types A, B and C.
workstations. To isolate and enlighten the effects produced by the proposed coordination schema, in this paper we use fixed *a priori* and constant decision weights, selected using a preliminary grid-inspection approach (Naso and Turchiano 1998). The WAs act as coordinators only in the case of the MAS using MCD decision rule.

The results of the simulation comparison are illustrated in figures 6–9. Clearly, the MCD provides the best values of all the considered performance measures, including the average queue length, not shown for brevity. Improvements with respect to the ARD strategy, which can be considered the best overall term of comparison, range between the 3.5% in the case of makespan and about 30% in the case of standard deviation of resource utilization. The latter result, which is comparable to that of the MWQ (the best policy for workload balancing), clearly indicates that the coordination contributes to achieve a better utilization of the workstations in the system.

![Figure 6. Comparison by makespan.](image)

![Figure 7. Comparison by flow time.](image)
The second set of experiments focuses on the effects of the cooperation algorithm (CA). Figures 10–13 compare the performance indices obtained simulating the MAS respectively with and without CA (setting the revision interval to zero) in two sample cases. To show that the effects of the cooperation are independent of the particular choice of rule weights, we report two cases. In the former, the agent’s rules have the same weights (all set equal to one), while in the latter the weights are set to their optimal values ($w_{MWQ} = 1.25$, $w_{SPT} = 2.00$, $w_{SD} = 0.50$, $w_{DA} = 0.25$), also used in the previous comparison. With the single exception of maximum flow time in the
case of optimized weights, the use of cooperation leads to better values for all the performance indices. Significant improvements are obtained both for the non-optimized and for the optimized weight sets. In particular, it is interesting to note that in general the relative improvements are much higher in the case of non-optimized weight set, confirming the ability of the proposed strategy to improve the global performance especially when local decisions are not optimal. Analogous results, omitted for brevity, were obtained simulating other combinations of weights. As a further confirmation of the actual effectiveness of the proposed strategy, we

![Figure 10. Effects of the cooperation strategy on the makespan.](image)

![Figure 11. Effects of the cooperation strategy on the flow time.](image)
must report that machine blocking occurs only in the 0.2% of the simulation run-time if the CA is used, whereas the incidence of this event rises up to 19% when the coordination is disabled.

6. Conclusions and future work

This paper describes a multi-agent approach for part dynamic routing in flexible manufacturing systems. The approach combines different tools offered by research in the areas of computational and artificial intelligence. In our experiments, the concurrent application of fuzzy algorithms with distributed decision architectures has shown considerable potential. Namely, fuzzy multi-criteria algorithms appear to be a promising way to design intelligent agents capable of making qualitative evaluation...
of different decision parameters in almost instantaneous decision times. For instance, results in figures 6–9 clearly show the agent’s ability to trade off between the inherent workload balancing capability of the MWQ decision rule and the other advantages of the SD and SPT heuristics.

The simulations also confirm the considerable advantages of the proposed coordination mechanism: the temporary intervention of a higher-level entity leads to significant performance improvements, whose entity seems to fully counterbalance the increased communication and computational burden involved in the cooperation functions. Since the cooperation is based on a temporary modification of the interaction schemes between part and workstation agents, it fully preserves the fault tolerance of the underlying heterarchical architecture. In the worst-case scenario, if the coordination is unavailable due to faults or other unforeseen reasons, the part agents can continue to operate autonomously as in a conventional MAS. Furthermore, by the appropriate configuration of the triggering condition for coordination actions, the proposed approach can be easily extended to manufacturing environments differing from the scenario considered in our simulations.

The proposed research leaves many interesting issues open for further research, including agent adaptation, heterarchical AGV dispatching, and development of a distributed simulation environment. More specifically, while the weights associated to each rule used by the agent are currently fixed a priori, an adaptive strategy based on evolutionary computation capable of selecting on-line (i.e. while the agents are running) the most suitable mix of decision rules for the current operating conditions is under development. The aim of this research is to obtain agents with self-tuning capability, and to improve the overall reactivity to unforeseen circumstances. Moreover, research in progress is also devoted to extend the heterarchical paradigms to the AGV dispatching system, which is at present based on a centralized blackboard. Finally, prior to the final experimentation on an automated manufacturing system, the MAS will be tested on a distributed network of computers, each hosting one or more of the agents composing the heterarchical network, to investigate the critical issues related to distributed autonomous controllers that cannot be properly examined when running the whole set of agents on a single platform.

Acknowledgement

The authors wish to thank the anonymous reviewers for their constructive comments.

References


