A soft computing approach for task contracting in multi-agent manufacturing control

Guido Maione\textsuperscript{a,1}, David Naso\textsuperscript{b,*}

\textsuperscript{a}Dipartimento di Ingegneria dell’Innovazione, Università degli Studi di Lecce, Via Monteroni, 73100 Lecce, Italy

\textsuperscript{b}Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Re David, 200-70125 Bari, Italy

Abstract

This paper describes a new task-contracting schema for multi-agent manufacturing control based on soft computing. It aims to apply fuzzy techniques to implement a real-time multi-criteria task-contracting mechanism for part flow control in manufacturing floor. For comparison purposes, the paper also considers other recently proposed evolutionary strategies to adapt and optimize agents’ decision parameters to the changing conditions of the manufacturing floor. All the considered approaches are compared on a detailed simulation model of a hypothetical manufacturing system that was recently proposed in literature as benchmark for multi-agent control systems.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Soft computing; Manufacturing systems; Multi-agent systems; Dispatching

1. Introduction

In recent years, the rapid evolution of information and communication technologies has strongly influenced the research in the area of manufacturing. The demand for wider varieties of products with shorter life-cycles calls for manufacturing system flexibility, modularity, reconfigurability, and robustness to external perturbations and faults. As a consequence, innovative approaches are emerging in manufacturing control systems. In this context, multi-agent systems (MAS) technology is one of the most promising areas.

Agents are software entities executing a set of tasks in a complex environment. They are designed for specific control functions, and are often associated to physical entities operating in the manufacturing environment (machines, raw parts, automated guided vehicles (AGV)). From a software technology point of view, agents are similar to software objects. However, while the latter run upon call by other higher-level objects in an hierarchical structure, agents must run continuously and autonomously. According to the philosophy of MAS, information, decision and control must be as much as possible physically and logically distributed across autonomous agents in the plant. The overall control of the entire manufacturing environment should result from the concurrent actions of the multiple agents operating in the plant. From the point of view of control software development, this paradigm promises reduced programming complexity, scalability, and reconfigurability. The long-term goal of multi-agent research is to obtain plug-and-play interchangeable and expandable manufacturing hardware and intelligent and reactive control software [1].
The papers [1,2] represent two authoritative and extensive surveys of this research area, which has received considerable attention in the recent literature.

MAS techniques lend themselves to work in real-time dispatching environments, which are the focus of the majority of current research [1]. In these contexts, dispatching systems take decisions on part flow in real time and represent a form of low level control based on algorithms using a limited amount of local information. Adequate coordination/cooperation mechanisms are required to achieve the desired global behavior when the decision task is distributed across concurrently operating agents. In fact, many researches on MAS focus on the development of decision strategies and protocols of interaction between agents, based on metaphors of negotiation in micro-economic environments. Since the first appearance of the earliest approaches, the main motivation of this type of research is that "invisibles hands to guide the negotiation to improve system performance" [3]. Hence, these invisible hands must also compensate for the lack of a general view in local information. In other words, a dominating opinion in multi-agent manufacturing research is that by using market or auction based task contracting, the network of interacting autonomous agents can lead to the requested characteristics of flexibility and reactivity of the manufacturing system [4–6]. On the other hand, many authors [7] also point out the inherent difficulty in designing effective forms of cooperation between local autonomous controllers, without contradicting in some way the multi-agent design principles, which suggest the absence of hierarchical forms of supervision. In fact, more than 30 years of research in the field of dispatching systems confirm that there is no explicit way to relate the local objectives of a decision entity to global system performance [8]. As a consequence, global performance becomes extremely sensitive to the definition and the fine tuning of the task-contracting rules [7]. Furthermore, in general, negotiation algorithms have many free parameters to tune, and there are few general criteria to set up agents that can effectively emulate human decision-making.

Although many reasons call for research in this area, technical literature reports relatively few attempts to cope MAS problems with modern problem-solving soft computing methodologies inspired to human reasoning and biological systems. Namely, it is known that agents derive inspiration from communities of intelligent decision makers in uncertain and extremely dynamic environments, and that fuzzy techniques are suited to model human decision-making. Therefore, this paper discusses the potentialities of the challenging combination of soft computing techniques and multi-agent paradigms in task-contracting problems for manufacturing control. In particular, the paper examines if and how much agents’ decision schemes benefit from the application of fuzzy methodologies. In principle, these approaches are more effective in finding qualitative and approximate rewarding tradeoffs between conflicting decision objectives, in obtaining more context-independent reactive behaviors, and in solving the coordination/cooperation problems.

Other tools from soft computing lend themselves to tackle multi-agent research problems. In particular, evolutionary algorithms (EA) [9,10] are a set of versatile stochastic search techniques inspired to natural evolution and genetics that have been successfully applied to a variety of manufacturing problems, including also task negotiation and scheduling in distributed environments.

To enlighten the potentiality of soft computing methodologies, this paper describes a new multi-objective task-contracting mechanism based on fuzzy decision-making and compares it with other conventional negotiation schemes for real-time part flow control. For comparison purposes, the paper considers a recently proposed evolutionary strategy [11] to adapt agent’s decision parameters to the changing conditions of the manufacturing floor. All the proposed approaches are compared on a detailed simulation model of a hypothetical manufacturing system that was recently proposed as benchmark for MAS [12]. Referring to a common benchmark is necessary. In fact, as recently pointed out by Cavalieri et al. [13], many authors often do not provide sufficient detail on their design hypotheses and on the structural characteristics of the manufacturing system. Thus, an objective view of the applicability and the performance bounds of their approaches is lacking. Hence, the adoption of this case study as common framework to evaluate the performance of MAS is fundamental for an objective understanding of the true potentials of agent technology in real-time control. Our simulation
study encompasses both steady-state performance measures, and the response of the controllers to dynamic changes as workstation (machine) and AGV failures and changes in the workload.

The paper is organized in the following way. Section 2 surveys the related literature focusing on the applications of soft computing techniques to task contracting in manufacturing. Section 3 describes agents’ pricing and buying strategies based on fuzzy decision-making techniques. Section 4 overviews the interaction protocols between agents. Section 5 discusses the simulation results. Finally, Section 6 draws conclusions.

2. Related research

The area of soft computing is an evolving collection of methodologies aiming to exploit the tolerance for imprecision and uncertainty to achieve robustness, tractability and low costs [14]. Encompassing a set of theories and algorithms inspired to human approximate classification and reasoning, fuzzy logic (FL) is a main component of soft computing that has received considerable attention in the area of manufacturing control and optimization. Since their earliest developing stages, fuzzy techniques [15] have been applied to assign priorities in job shop environments. In fact, the inherent turbulence and the lack of precise information about the real-time operating conditions make conventional planning strategies based on analytical models extremely unsuitable for these environments. Fuzzy systems, on the other hand, attempt to mimic human expertise in the form of rules as “IF work in progress of part type A is high AND average service time of part type B is low AND . . . THEN priority part A is low AND priority part type B is medium AND . . .”. Therefore, in such applications the fuzzy controller works as a numerical model of the heuristic knowledge of a human planner, and in general yields performances comparable to those of plants directly controlled by human experts. Among the others, Ben-Arieh and Lee [16], Nahavandi and Solomon [17] and Erkmen et al. [18] have proposed this typical application of fuzzy control in manufacturing environments. These approaches not only lead to higher performance with respect to conventional strategies, but rather provide a transparent evaluation and decision logic, which enlightens the mechanisms underlying the input–output behavior of the controllers. This is an extremely important feature, since many manufacturing environments still need human supervision to operate correctly.

Angsana and Passino [19] proposed a scheduling approach that is one of the earliest attempts to design a distributed network of fuzzy controllers in manufacturing. The authors underline that their scheduling system, a fuzzified version of optimal buffer clearing policies, could be viewed as a “distributed intelligent system”, since each machine is separately controlled by a local dispatcher. Each dispatcher is a fuzzy controller working on rules having buffer contents as inputs and next part type to be processed as output. The system is distributed, since each controller uses only local information to perform its decisions. However, from the multi-agent viewpoint it lacks of explicit interaction between the autonomous controllers, since the authors acknowledge that the number of parts travelling over the track from one machine to the other is the only form of communication between controllers. An interesting aspect that is considered in the paper is the capability of the controllers to self-tune themselves when machine parameters change. More precisely, the authors devise an adaptation law that changes a single parameter of the control on the basis of the performance of the policy over a shifting time-window. The parameter under tuning is the amplitude of the definition universe of the fuzzy sets, and the objective of the adaptation is to have rules always coherent with the actual operating conditions.

A recurrent idea to improve the effectiveness of dispatching strategies is to develop specific heuristics to switch in real time between different decision rules. To implement such schema in a distributed decision environment, it is necessary to understand how intelligent agents modify their actions taking into account the system operating conditions. There are recent centralized solutions to this problem. For instance, Park et al. [8] 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, e.g. “IF system utilization is greater than “a” AND buffer size is “b” AND . . . THEN the dispatching law is SPT”. The rule base is also optimized by an “inductive learning approach” using a set of training examples. Analogously, Yu et al.
propose fuzzy rules to associate a set of “environmental variables” describing the manufacturing conditions (e.g. the current workload or the remaining production time for parts at hand) to a set of dispatching rules. Recently, also Kazerooni et al. [21], Naso and Turchiano [22], Gao et al. [23], Wan and Yen [24] suggested similar approaches. However, associating a specific dispatching rule to each state of the plant is not an easy task, especially when multiple dispatchers share the part flow control duties. Namely, in most manufacturing systems, there is no evident relationship between system performance and the single decisions taken by the autonomous agents.

Real-time task scheduling methodologies for MAS employ intelligent negotiation strategies to increase system fault tolerance and robustness to unforeseen circumstances (e.g. bottlenecks and variable product demand). In these approaches, agents have to effectively balance their autonomy against simplicity of regulation of the overall system behavior. However, task-contracting schemes inherit the inherent myopia and unpredictability of performance bounds typical of dispatching systems. These limitations are widely acknowledged among the main reasons against the widespread use of agents in industry. In this specific context, the cross-fertilization of soft computing and MAS tools offers a promising research direction that has been recently investigated. For instance, to ensure that agents act coherently from a global point of view, compensating the non-local effects of local decisions, Maione and Naso [11] propose an evolutionary supervisor coordinating two types of agents, namely part agents (PA) and workstation agents (WA), which control part flow and service sequence, respectively. The objective of PA is to obtain the requested service in the shortest time. On their part, WA must minimize idle and setup times. Both type of agents use a fuzzy multi-criteria decision algorithm combining several decision rules in a unique criterion. All agents have a limited lifetime, after which a discrete-event simulation evaluates their fitness (i.e. the ability to fulfill their objective). Then, an iterative evolutionary strategy employs simulation results to compute new agents with improved fitness. Hence, the evolutionary strategy acts as an implicit supervisor continuously adapting the population of agents controlling the plant.

Evolutionary algorithms, another fundamental component of soft computing, are robust search strategies, inspired to natural evolution laws and genetics, which have been extensively applied to manufacturing problems. Most of this research focuses on off-line optimization of hard combinatorial problems (e.g. production planning, scheduling). References [25,26] are some recent examples of this research, and paper [27] offers a comprehensive survey of the research in this area in recent years.

Due to their common inspiring metaphor, centered on communities of biological entities “adapting” their behavior to changing environments, and due to the versatility and robustness of their performance, evolutionary algorithms have received an increasing attention in the area of MAS. Also in this case, the lack of adequate models, and the inherently complex, turbulent and unpredictable nature of the operating environment impede the application of conventional optimization approaches, and constitute the main motivations to use surviving-of-the-fittest-driven heuristics. A common example is the optimization of the decision strategies of a network of distributed controllers. In principle, whenever the strategy underlying agents’ decision, interaction or negotiation tactics can be parameterized (i.e. encoded in a string of binary or real parameters), it is possible to use an evolutionary algorithm to optimize the entire network of agents. Applications of this point of view include communities of cooperative robots [28,29] or artificial benchmarks [30–33]. Although size and complexity of these agent benchmarks are considerably smaller than those of manufacturing system problems, many authors conclude that the successful results obtained with intelligent agents in these simplified virtual domains will be soon extended to real-world problems. Recent literature offers some preliminary applications in manufacturing domains. For instance, McDonnell and Joshi [34] use a reinforcement learning algorithm to determine the optimal negotiation strategy between agents controlling the machine setup timing in a flexible manufacturing system (FMS). Vacher et al. [35] use a network of agents to plan operation in a job-shop type of environment. Agents associated to parts interact with each other by negotiating with heuristic rules the start and completion times of associated parts. The genetic algorithm implements a supervisor evaluating each agent’s “impact” on the production plan and optimizing the part sequence. Uliuer [36] presents an intelligent
material handling system composed of a set of independently controlled pallets. An intelligent agent associated to each pallet solves task negotiation and conflict resolution problems using a neural-fuzzy knowledge based controller. In addition, pallet optimal path planning is tackled with a genetic algorithm. Finally, also Flake et al. [37] consider the problem of autonomous vehicles control in manufacturing systems using fuzzy agents. Stations and AGV negotiate services (transportation from a workstation to another one) using a fuzzy multi-criteria algorithm that considers distance and reliability of the AGV as decision parameters.

3. Negotiation schemes with fuzzy decision algorithms

Most multi-agent scheduling systems [3] drive inspiration from metaphors of markets (e.g. the Contract Net Protocol [38]) where sellers and buyers have to reach an equilibrium between conflicting objectives, i.e. to maximize profit and to minimize costs, respectively. In micro-economic environments negotiation tasks are inherently multi-objective, and expert contractors always tend to find a rewarding compromise between conflicting aspects. The role of the “tradeoff functions” used to emulate human expert decision-making is fundamental for the interacting control agents to obtain an effective task scheduling. The area of fuzzy multi-criteria decision-making [15] provides a considerable variety of mathematical tools capable to mimic and exploit the fundamental “vagueness” of human decision-making. The work presented in this paper shows that these tools can be also profitably employed to define more effective task-contracting rules for MAS. In particular, this section introduces a negotiation algorithm using fuzzy set techniques and multi-criteria decision-making theory to distribute the part flow real-time control across a set of interacting agents, namely workstation and part agents, WA and PA, respectively.

The final price of a processing priority results from a negotiation between the WA associated to the processing workstation and the PA associated to the physical part to be processed. A PA is a software controller retaining all the information on part process plan (the sequence of operations and the set of operations that can execute each operation), and the decision algorithm. The goal of the PA is to obtain the requested operations by minimizing part flow time (i.e. the time necessary to complete the processing plan), and the cost of service.

WA are software controllers associated to the workstations executing parts in the manufacturing plants. The task of a WA is to assign a priority to each part requesting an operation on the associated machine, which must process the parts according to their priority. The processing priority is an integer ranging from 1 (highest priority) to the maximum buffer capacity L (lowest priority). WA sell operations with a pricing strategy that relates the offered priority to several factors such as current workload, setup changes with respect to the preceding and next part in queue, estimated delivery time, etc. The objective of WA is to maximize both the profit and the throughput (rate of parts processed per time unit, achieved by minimizing both setup and idle times). The following subsections describe the cost evaluation algorithm and the pricing strategy used by PA and WA, respectively.

3.1. Part agent decision algorithm

The PA decision algorithm works as follows. Let $M_s$ be the set of workstations (machines) in the manufacturing system. Suppose that a part completes an operation on a workstation. As soon as the part is ready for a new operation, the associated PA selects the subset of workstations $M \subseteq M_s$ that can execute the next operation. Then the PA sends a service request to each of the WA associated to the $m$ machines in the set $M$. Each of the $m$ queried WA responds with a message containing $h_j$ different offers, $j = 1, \ldots, m$. Hence, the total number of alternatives available to the PA for the next operation is $q = \sum_j (h_j)$.

On the basis of these preliminary definitions, we consider an offer as a couple:

$$O_{ij} = [p_i, c(p_i)]_j, \quad i = 1, \ldots, h_j \text{ and } j = 1, \ldots, m$$

(1)

where $p_i \in \{1, \ldots, L\}$ is the $i$th position in buffer, and $1 (L)$ indicates the first (last) position in queue, i.e. the highest (lowest) processing priority. The second component $c(p_i) \in [0, 1]$ is the cost of the position,
computed by the WA with an algorithm described later. Fig. 1 describes the snapshot of a generic decision problem for a PA.

Now, let \( M_0 \) be the subset of WA that communicate the availability of further offers, and \( M_0^0 \) the subset of WA that have no other places to offer \( (M_0 \setminus M_0^0 = \emptyset) \). For instance, in Fig. 1 we have \( M_0 = \{WA_1, WA_3\} \) and \( M_0^0 = \{WA_2\} \). Moreover, let \( m' \) and \( m'' \) indicate the cardinalities of \( M' \) and \( M'' \), respectively. At this point, the PA has to choose either to accept one of the received offers or to start a negotiation for a different priority with one of the WA in \( M_0 \). To make this decision, the PA formulates \( m' \) “fictitious” offers:

\[
\tilde{O}(h_j+1) = [p_j + 1, c(p_j + 1) - \epsilon], \quad j = 1, \ldots, m'
\]

where \( p_j \) indicates the priority having the smallest cost among those offered by the \( j \)th WA in \( M' \):

\[
c(p_j) = \min_i [c(p_i)], \quad j = 1, \ldots, m'
\]

and \( \epsilon > 0 \) is a tuning parameter called negotiation factor that is set according to the WA pricing strategy (the higher is \( \epsilon \), the more the PA tends to negotiate).

The aim of the fictitious offers is to estimate whether a negotiation with one of the WA could improve the cost/priority tradeoff of the offer. By adding the \( m' \) fictitious offers to the \( q \) offers from the WA, the part has \( n = q + m' \) alternative choices to evaluate. The fuzzy set theory gives effective tools to model the satisfaction of decision objectives and to combine them in a unique criterion of evaluation. In particular, each decision objective can be described with a fuzzy membership function where degree 0 (1) expresses the minimum (maximum) satisfaction of the objective, while all the intermediate values represent degrees of partial satisfaction [39,15]. The PA uses the following two fuzzy objective functions to evaluate the \( n \) offers:

minimum cost \( S_c : \mu_{S_c}(c) = (1 - rc)^\alpha, \quad c \in [0, c_{\text{max}}] \)
highest priority \( S_p : \mu_S(p) = \left(1 - \frac{p - 1}{L} \right)^{\omega_p}, \)
\( p \in \{1, 2, \ldots, L\} \) \hspace{1cm} (5)

In (4) and (5), \( c \) indicates the price of the offer (cost), ranging from zero to \( c_{\text{max}} < 1/r \) (assuming that the maximum cost of an operation is equal for all the operations, and that the maximum currency amount assignable to a PA is unitary), \( r \) is the number of operation steps to complete the part, and \( \omega_c \) and \( \omega_p \) are weighting factors grading the influence of each of the objectives in the final decision. Note that to perform efficient aggregations between the two conflicting criteria, it is desirable that the degrees of satisfaction of both criteria are always greater than zero. In fact, a null degree of satisfaction for one of the objectives completely excludes the corresponding alternative from the decision process, independently of the weight assigned to the objective. Also note that the initial amount of fictitious currency assigned to a PA is a means to implicitly assign an external priority, since “poor” PA can afford only low priorities.

The global objective is the fuzzy aggregation of the two weighted goals, i.e.:
\[
\hat{S}(p_k) = S_p(p_k) \otimes S_c[c(p_k)], \quad k = 1, \ldots, n \hspace{1cm} (6)
\]
where \( \otimes \) indicates a fuzzy \emph{t}-norm [40]. Common \emph{t}-norms include the minimum and the product, although basing on recent research results [11] our research uses a parameterized operator providing a more realistic tradeoff between the conflicting objectives. Namely, we use the “compensatory AND” operator described in [15] (p. 37) and defined as follows:
\[
\mu_S(p_k) = \left[\mu_S(p_k) \mu_S[c(p_k)]\right]^{1-\gamma} \left[1 - (1 - \mu_S(p_k)) \right] \left[1 - (1 - \mu_S[c(p_k)])\right]^\gamma,
\]
\[
0 \leq \gamma \leq 1 \quad \text{and} \quad k = 1, \ldots, n \hspace{1cm} (7)
\]
where \( \gamma \) is a free parameter indicating where the actual operator is located between “fuzzy AND” and “fuzzy OR”.

The PA selects the offer with maximum satisfaction in the global decision criterion, i.e. the offer for which:
\[
\hat{S}(\hat{p}) = \max_{k=1 \ldots n} \{\hat{S}(p_k)\}\hspace{1cm} (8)
\]

If \( \hat{p} \) is one of the \( q \) non-fictitious offers, the PA sends a confirmation message to the WA offering \( \hat{p} \) and a cancellation message to the remaining \( m - 1 \) workstations. In case \( \hat{p} \) is a fictitious offer, the PA starts a negotiation with the corresponding WA. A negotiation process is a cycle of reiteration of PA requests and WA offers, during which the WA reformulates offers at a discounted price and the PA reevaluates the received offers until the PA selects a non-fictitious offer as final decision. To limit the number of reiterations and avoid indefinite negotiation loops, we define a negotiation counter (the Negotiation Progress Index (NPI)). In case the NPI reaches the maximum number of allowed iterations, the PA accepts the best non-fictitious offer available in the last decision process.

Once the decision is made, the PA sends a confirmation message to the WA offering the chosen alternative and waits for confirmation or timeout. The timeout bounds the maximum waiting time of an agent for a response, to avoid indefinite waits due to lost messages or communication faults. When the timeout expires without receiving a response, the PA resends the confirmation request. If the reservation is confirmed, the PA initializes the transportation procedures. On the contrary, if the reservation is rejected, the PA contacts the WA associated with the second offer in rank, and if necessary iterates these steps until a reservation is confirmed.

Using an aggregated multi-criteria algorithm makes the decision strategy extremely modular and reconfigurable. In fact, it is easy to add new criteria taking into account new constraints, preferences or different priorities between the production objectives. Also, from the algorithm complexity viewpoint, the fuzzy decision process is extremely simple, and requires a modest computational cost. In any case, the effort is comparable to that of the most widespread agent contracting algorithms, even when many alternatives are available to PA (e.g. in very flexible agent contracting systems with many processing machines and/or alternative processing sequences).

3.2. Machine agent pricing strategy

As in the case of PA, also WA use a fuzzy multi-objective decision algorithm to calculate the price of the offered priorities. WA use the following six distinct criteria.
3.2.1. Offered position p

This fuzzy strategy takes into account the number of parts preceding and following the offered position, so that the more the parts preceding (following), the lower (higher) the price. A third fuzzy set accounts for the cost of priority:

\[ C_{11} : \mu_{C_{11}}(p) = \left(1 - \frac{n_{\text{bef}}(p)}{L-1}\right)^{\omega_{11}}, \quad 0 \leq n_{\text{bef}} \leq L - 1 \] (9)

\[ C_{12} : \mu_{C_{12}}(p) = \frac{n_{\text{aft}}(p)}{L-1}^{\omega_{12}}, \quad 0 \leq n_{\text{aft}} \leq L - 1 \] (10)

\[ C_{13} : \mu_{C_{13}}(p) = \left(1 - \frac{p-1}{L-1}\right)^{\omega_{13}} \] (11)

in which \( n_{\text{bef}} \) (\( n_{\text{aft}} \)) is the number of parts placed before (after) the offered position \( p \), i.e. the number of parts with higher (lower) priority and \( \omega_{11}, \omega_{12} \) and \( \omega_{13} \) are weighting factors. The three criteria can be aggregated using an operator in the large set of \( t \)-co-norms [15,40]. In this paper, we use the generalized mean operator ([40], p. 89) defined as follows:

\[ C_1 : \mu_{C_1}(p) = \left[\frac{1}{z}(\mu_{C_{11}}(p)^z + \mu_{C_{12}}(p)^z + \mu_{C_{13}}(p)^z)\right]^{1/z} \] (12)

Parameter \( z \) grades the actual location of the operator between the arithmetic and the geometric means. Note that, in contrast with the \( t \)-norms used for the PA multi-objective decisions, the generalized mean does not require degrees of satisfaction greater than zero to perform efficiently.

3.2.2. Workload preceding \( p \)

This criterion considers the total processing time (workload) corresponding to parts occupying positions preceding the offered ones. Clearly, the lower (higher) is the workload, the more expensive (cheaper) is the makespan of the longest operation on the workstation.

3.2.3. Operation processing time

This criterion takes into account the processing time \( \tau \) of the requested operation, so that longer operations have higher costs:

\[ C_3 : \mu_{C_3}(\tau) = \left(\frac{\tau}{T_{\text{max}}}\right)^{\omega_3} \] (14)

3.2.4. Adjacency of homologous operations

This criterion aims at minimizing setup time by discounting the price of positions adjacent (i.e. preceding or following) to positions hosting parts requiring the same setup. For sake of brevity, we define “homologous” all the parts in the buffer requiring the same setup configuration of the PA requesting the offer. The larger the number of homologous parts hosted in positions adjacent to the offered one, the smaller the offer. We define:

\[ C_4 : \mu_{C_4}(p) = \left(1 - \frac{n_{\text{adj}}(p)}{v}\right)^{\omega_4} \] (15)

where \( v \) is a normalization parameter and \( n_{\text{adj}}(p) \) is the adjacency function defined as follows:

\[ n_{\text{adj}} = \sum_{k=1}^{d_{\text{bef}}} \phi(k) + \sum_{k=1}^{d_{\text{aft}}} \phi(k) \] (16)

where \( d_{\text{bef}} \) and \( d_{\text{aft}} \) are the numbers of adjacent positions preceding or following \( p \) that are either empty or hosting a homologous part. Clearly, it holds \( d_{\text{bef}} \leq L - 1 \) and \( d_{\text{aft}} \leq L - 1 \). The function \( \phi(k) \) is defined as follows:

\[ \phi(k) = \begin{cases} e^{-\beta k}, & \text{if the position } k \text{ hosts a homologous part} \\ \beta e^{-\beta k}, & \text{if the position } k \text{ is empty} \end{cases} \] (17)

The function \( \phi \) gives a discount factor exponentially decreasing with the increasing distance from the offered position \( p \). Parameter \( \beta > 0 \) rules the shape of the function, and \( 0 < \beta < 1 \) accounts for the possibility that the empty position will be allocated in future to another homologous part. In principle, the discount is maximum when all the positions in the buffer host homologous parts. In this case, assuming
an infinite buffer capacity, the upper bound $v$ for the adjacency function $n_{adj}(p)$ is computed as:

$$v = \sum_{k=1}^{d_{max}+\infty} e^{-2k} + \sum_{k=1}^{d_{max}+\infty} e^{-2k} = 2 \left( \frac{1}{1 - e^{-2}} - 1 \right)$$

(18)

Clearly, in case of limited buffer capacity ($L$ finite), the normalization parameter $v$ can be computed using finite sums in Eq. (18).

3.2.5. Current buffer content

This criterion attempts to avoid machine starvation offering low prices when the buffer is empty, and at the same time to preserve a number of empty locations in the buffer to avoid buffer saturation. This is achieved by increasing the cost of the offer as the number of occupied locations $n_{oc} = n_{bef} + n_{aff}$ grows:

$$C_5 : \mu_{C_5}(p) = \left( \frac{n_{ha}}{L - 1} \right)^{c_{oc}}$$

(19)

3.2.6. Negotiation progress

This criterion aims at promoting a tradeoff between PA and WA conflicting objectives. As the negotiation progress index increases, the new offers are formulated at a cheapest cost:

$$C_6 : \mu_{C_6}(\text{NPI}) = \left( 1 - \frac{\text{NPI}}{\text{NPI}_{\text{max}}} \right)^{c_{oc}}$$

(20)

NPI$_{\text{max}}$ is the maximum number of allowed negotiations between a PA and a WA.

To sum up, the total price is obtained by combining the six pricing criteria with an aggregation operator $H$:

$$C(p, \tau, \text{NPI}) = H(C_1(p), C_2(p), C_3(\tau), C_4(p), C_5(p), C_6(\text{NPI}))$$

(21)

This paper implements the aggregation of the six criteria with the generalized mean already defined in Eq. (12). The result of the fuzzy aggregation is then used to compute the fuzzy cost of the operation:

$$c(p) = c_{\text{max}} C(p, \tau, \text{NPI})$$

(22)

where the maximum allowed expense for an operation $c_{\text{max}}$ is a normalization factor.

In all the above criteria, the weights $\omega$ allow the user to grade the influence of each rule in the final cost. Jointly with the weights $\omega_e$ and $\omega_p$, $c_{\text{max}}$, and the aggregation operators $\gamma$ and $z$, these variables constitute the set of free parameters of the task-contracting strategy. They can be selected appropriately to obtain the desired overall behavior, according to the production goals, and the domain constraints assigned by higher levels of the manufacturing management. As in most application of soft computing tools, the proper configuration of the algorithm requires a careful trial and error procedure. Certainly, the robustness of fuzzy decision laws to variation of their configuration parameters, and the simple interpretation of each separate fuzzy criterion in the heuristic decision algorithm, contribute to mitigate the complexity of this configuration task with respect to other task-contracting algorithms presented in literature. In Section 5, we also briefly describe a supervisory algorithm [11] that can be used to configure automatically the negotiation strategy to achieve a pre-specified production objective.

4. Agent interaction protocols

Each PA applies the following protocol to negotiate with one or more WA. As a part enters the system or completes an operation, it is ready for a new step. The corresponding PA requests the service from the WA associated to the workstations that can execute the next operation on the part. Fig. 2 describes the generic structure of a service request. The first portion of the message is common to all the requests for a given operation, and contains information about the PA making the request. This information includes the identification code, which univocally distinguishes each part in the system, and the current currency availability used by the WA to tender only “affordable” positions to the PA. The second portion is specific for each WA receiving the message, and contains the Negotiation Progress Index, i.e. the counter of the number of negotiation steps between the PA and the WA. The counter can also transmit exceptional messages ($-1$ indicates the final rejection of the offer, $-2$ indicates that the part has found no alternative server and is blocking the hosting workstation). After the request transmission, the PA stands by waiting for replies. To avoid indefinite waits, the PA considers only the offers received within a predefined wait timeout.

Upon receipt of the service request message, each contacted WA with available buffer locations applies the algorithm of Section 3.2 to elaborate the offer. If
the WA has multiple locations to sell, it replies with a message containing multiple offers (see Fig. 3). However, to limit the communication overhead, each message contains only a few offers (e.g. three in the simulations described in the following), while a special flag signals the availability of further offers on request. Fig. 3 describes the structure of a generic offer by a WA. When the PA receives the offer, it enters the decision phase described in Section 3.1. If the PA identifies an acceptable offer, it immediately sends a reservation request to the offering workstation. On the contrary, if no offer is satisfactory, the PA starts renegotiating. If no further offer is available, the PA evaluates once more the offers previously rejected. If no offer is available at all, a blocking of the machine hosting the part occurs. Clearly, this is a critical situation due to system malfunctions or overloads and requires special solutions. If the manufacturing system operates with high loads, it is necessary to implement congestion avoidance policies for preventing deadlocks. In our approach, the PA sends a special message (NPI = −2) to the WA signaling the blocking condition. The first contacted WA that is able to satisfy the request processes these types of messages with the highest priority. When the PA completes the negotiation and chooses the next destination of the part, it sends a final confirmation. Figs. 4 and 5 summarize the activity flow of a PA and WA, respectively. We finally remark that, in principle, a WA can be interrogated by one or more PA while it is already contracting with another PA. However, this event seldom occurs if the proposed task-contracting algorithm is implemented on a conventional network of low-cost personal computers. In fact, even considering the typical transmission delays of common data transfer protocols, the negotiation time is still orders of magnitude smaller than physical part processing or transfer times of most manufacturing systems. For this reason, the presented version of the contracting algorithm does not consider the concurrent negotiation between a WA and multiple PA (e.g. a WA that modifies its offer to a first PA due to the receipt of a request from a second PA). However, this interesting issue remains open for further research because, in principle, it could let the WA to organize their workload more efficiently.

<table>
<thead>
<tr>
<th>PART Identification Code</th>
<th>Currency Availability</th>
<th>Negotiation Progress Index</th>
<th>Requested Operation Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>C</td>
<td>e</td>
<td>P</td>
</tr>
</tbody>
</table>

- Π is the set of part identification codes
- CA| |_max| is the initial amount of currency assigned to each PA
- N| |P|i| |max| is the maximum number of negotiation cycles
- Ω is the set of operation codes

Fig. 2. Structure of a generic service request by a PA.

<table>
<thead>
<tr>
<th>Machine Identification Code</th>
<th>Requested Operation Code</th>
<th>Offer .1</th>
<th>Offer .2</th>
<th>Offer .n</th>
<th>(Further offers Availability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>IC</td>
<td>∈</td>
<td>M</td>
<td>o</td>
<td></td>
</tr>
</tbody>
</table>

- M is the set of workstation (machine) identification codes
- Ω is the set of operation codes
- (p| |i| ,c(x| |i| )) is a generic offer where p| |i| is the offered position and c(p| |i| ) the offered price

Fig. 3. Structure of a generic offer by a WA.
5. A simulated benchmark for multi-agent part flow control

This section compares the performance of the proposed fuzzy negotiation architecture (FNA) with three different multi-agent control networks for real-time part flow control in flexible manufacturing systems. The comparison uses a model of a hypothetic FMS that was conceived to be both generic and detailed to allow a fair benchmarking between alternative MAS approaches [12,13]. Fig. 6 describes the system layout. The FMS has multi-purpose machines and two AGV for part transfers between stations. For sake of clarity, we give technical
details of the FMS model in Fig. 7 and Tables 1–3. Fig. 7 shows the alternative processing sequences for the three parts processed in the system. Thanks to the processing flexibility of the workstations, the parts can complete their working procedures by visiting many alternative sequences of machines, so that PA can chose in real time the servers for their next operation.

![Diagram](image-url)

Fig. 5. Workstation agent decision flow.

![Diagram](image-url)

Fig. 6. Layout of the experimental benchmark.
The model is built using a commercial discrete-event system simulation software (ARENA), while all the controlling agents and the interaction protocols are implemented in C++ routines continuously interacting with model simulator. To give an idea of the computational complexity, each of the simulations described in the following takes approximately 10 minutes on a Pentium IV based personal computer. The task-contracting algorithms take a negligible percentage of the processing time, mostly spent to perform the discrete-event simulation of the manufacturing model. Such modest computational requirements make the proposed strategy fully transferable to a real industrial environment.

In this benchmark, we first consider a multi-agent network that is a conventional market-based architecture where PA and WA negotiate operations on the basis of the expected operation delivery time. The PA request an offer to WA in terms of estimated delivery time of the operation and choose the machine that offers the earliest delivery time (EDT). Since the workstation agents organize the workload in queue with an earliest due date (EDD) discipline to accommodate urgent parts, we will refer to this first architecture as EDT-EDD.

The second multi-agent network is an extension of the previous one, in which the PA evaluate the possible destinations using several criteria (buffer contents, current setup, availability of transportation), rather than taking uniquely into account the delivery time. The decision mechanism uses a fuzzy multi-attribute
decision algorithm, which grades the influence of each decision rule in the global criterion. Recently, as an effective alternative to other task-contracting schemes, Maione and Naso [41] have proposed this fuzzy agents architecture, which offers an interesting tradeoff between simplicity and robustness of the performance. The main difference between the FA architecture and the scheme proposed in this paper (i.e. the FNA) is the amount of interaction between PA and WA. In the FA architecture, whenever a part is ready for a new operation, the associated PA requests to the available WA only real-time information regarding their operating conditions. Upon receipt of the requested information, the PA autonomously selects the server that will process the associated part. In contrast with the FNA architecture, the WA does not play any active role in the decisions of a PA. Moreover, in the FA architecture, also the WA autonomously establishes the service priority among the parts waiting in the buffer of the associated workstation, by evaluating real-time parameters such as setup requirements, part due date, etc. Hence, also in the WA decision process, PA are passive entities that do not interfere or modify WA choices.

Since the lack of explicit interactions between part and workstation agents may lead to unsatisfactory performances, Maione and Naso [11] propose a supervisory mechanism based on evolutionary computation to adapt in real time agents’ decision policy to the actual operating conditions of the plant. The evolutionary supervisor continuously experiments new weights affecting the rules composing the decision algorithms, and uses the feedback on their actual performance to compute new decision weights. The so obtained evolutionary fuzzy agent (EFA) architecture is the last and most complex scheme considered as benchmark in this case study. As remarked in [11], EFA can also offer significant improvements both of steady-state performance and of system reactivity to unexpected faults.

The first part of the comparison refers to stationary conditions (machines and AGV running at full capacity). We use mean flow time and mean throughput as performance indices to compare the four architectures. Since deadlines are assigned using the total work content strategy [42], part earliness/tardiness concur with part flow time and hence are omitted for sake of brevity. Simulation intervals are long enough to ensure the proper level of statistical confidence. Results presented in the bar-charts are rounded averages on 20 replications, while curves of indices over time during perturbations describe one sample run over the 20 replications. Fig. 8 (flow time) shows that all the three architectures using fuzzy decision algorithms obtain shorter flow time with respect to the conventional EDT-EDD. The FNA improves flow time (throughput, see Fig. 9) of +5% (+4%) with respect to the EDT-EDD and provide −2% (−2%) worse indices with

![Comparison of strategies in stationary conditions](image)

Fig. 8. Mean flow time in stationary conditions.
respect to the EFA. The last result has to be expected since EFA runs a continuous monitoring and supervision of agent decision rules. On the other hand, the EFA is considerably heavier in terms of programming complexity, so that the actual cost/benefit ratio between EFA and FNA should be evaluated for each case. Summarizing the main result of this first set of simulations, the FNA is the best unsupervised strategy, providing a good tradeoff between local autonomy of the single agents and global system performance.

Fig. 9. Mean throughput in stationary conditions.

Fig. 10. AGV failure: comparison of strategies by throughput (moving average).
The second part of the comparison analyzes the response of the four architectures to three different perturbations of the stationary conditions, namely the failure of an AGV, the failure of a workstation and a workload variation (+50%). Distinct simulation runs enlighten the effects of each single perturbation. In all cases, the perturbation occurs in the central part of the simulation, and lasts for one third of the total time horizon. The objective of this second set of simulation is to evaluate the reactivity and the robustness of the compared task-contracting algorithms. More precisely, the simulations must verify that the proposed strategy with fixed configuration parameters guarantees the same relative performance in presence of changing operating conditions. Figs. 10–12 describe the case of AGV failure/repair process. We note the

---

**Fig. 11.** AGV failure: comparison of strategies by mean flow time.

---

**Fig. 12.** (a and b) AGV failure: comparison of strategies by produced parts.
Fig. 13. Workstation failure: comparison of strategies by throughput (moving average).

Fig. 14. Workload variation: comparison of strategies by throughput (moving average).
similar behavior of the three multi-criteria policies (improvements of FNA with respect to EDT-EDD reaches 13% during the perturbation, due to the use of a specific criterion minimizing the AGV use). The FNA has a more nervous behavior in terms of flow time (Fig. 11), which does not directly affect the throughput as shown in Fig. 10 (EFA and FNA have a throughput 10% higher than EDT-EDD in both nominal/perurbed conditions). The number of produced parts confirms the superiority of FNA (+9%/+11%) and EFA (+10%/+12%) with respect to the other architectures. Subjected to the other two perturbations, the FNA improves the behavior of the conventional architectures (+5%/+8%) and also performs remarkably closer (−2%−3%) than the other policies to the EFA architecture (see Figs. 13–16). Since the relative performance of the compared policies does not vary in the different cases, we can reasonably conclude that this result will hold true also when the disturbance occur randomly, as in most real-world industrial environments. For what concerns the cost/benefit ratio between the EFA and the FNA, this second set of simulation leads to conclusions analogous to the results of the first set. In particular, the fact that the improvement in performance obtainable with the evolutionary supervision is even less significant in these dynamic cases confirms the remarkable robustness of the proposed task-contracting strategy.

6. Conclusions

In the area of MAS, real-time task scheduling methodologies employ intelligent negotiation strategies to increase system fault tolerance and robustness to unforeseen circumstances (e.g. bottlenecks and variable product demand). In these approaches, agents have to effectively balance their autonomy against simplicity of regulation of the overall system behavior. However, task-contracting schemes inherit the inher-
ent myopia and unpredictability of performance bounds typical of dispatching systems. These limitations are widely acknowledged among the main reasons against the widespread use of agents in industry.

In this paper, we show that soft computing can contribute to mitigate these problems. We compare various multi-agent control schemes using soft computing with different degrees of complexity and of agent interactions. The simulation results indicate that, in general, the relative performance improves if the complexity of the multi-agent decision algorithm increases. In particular, the three approaches based on soft computing have better performance than the conventional delivery-time contracting algorithm.

The differences between the three schemes based on soft computing also enlighten the fact that the agent cooperation/coordination problem is still open. Namely, the higher the interaction between agents, the higher the number of free parameters to tune. Consequently, the relationships between each of these parameters and the performance of the plant become less transparent. In highly interactive agent networks, also evolutionary computation showed limited possibilities to improve performance. In fact, the FNA architecture subject to the same evolutionary supervisor used in the EFA has not provided the expected improvements with respect to the FNA with fixed decision weights. Here, we have omitted the results of this schema (evolutionary FNA) since they are indistinguishable from those of the hand-tuned FNA.

Finally, comparing performance indices shows that strategies with a considerably smaller amount of agent interactions can be tuned on line more effectively. In particular, EFA performs significantly better than FA, even if the programming overhead of a schema implementing the feedback of the effectiveness of agents’ decision logic is higher than that of an unsupervised schema. In this respect, the new FNA schema proposed in this paper may be considered the best tradeoff between performance and implementation complexity.

Acknowledgements

The authors wish to thank the anonymous reviewers for their constructive comments, which helped to improve the quality of this paper.

References


Guido Maione was born in Naples, Italy. He received the Laurea degree with honors in electronic engineering in 1992 and his PhD degree in electrical engineering in 1997, both from Polytechnic of Bari, Italy. He joined the University of Lecce in 1996 where he is currently assistant professor in automatic control at the Department of Innovation Engineering. He was visiting scholar at the Rensselaer Polytechnic Institute of Troy, New York, in 1997. His research interests are in the field of: modeling, control and simulation of discrete-event systems, with
particular reference to automated manufacturing systems; soft computing techniques applied to multi-agent systems; Petri nets and digraph models; systems and control theory; deadlock detection/recovery, prevention and avoidance in manufacturing systems.

David Naso received his first degree (Laurea) in electronic engineering in 1994, and his PhD degree in electrical engineering in 1998, both at the Polytechnic of Bari, Italy. Currently, he is assistant professor in automatic control with the Department of Electric and Electronic Engineering of the Polytechnic of Bari. His research interests focus on multi-agent control systems, manufacturing system control, computational intelligence and its application to control engineering.