Int J Adv Manuf Technol
DOI 10.1007/s00170-008-1387-7

ORIGINAL ARTICLE

Implementation of a supervised learning technique in a multi-agent system for building production orders

Israel Villar-Medina · Omar López-Ortega · Roberto Hernández-Gómez

Received: 23 June 2007 / Accepted: 4 January 2008

Abstract The authors describe the implementation of a supervised learning algorithm within a multi-agent system, whose general objective is to build *production orders*. Although this task has been carried out traditionally by the *production management system*, the classic approach lacks adaptive techniques and intelligent behavior. It is acknowledged that the combinatorial problem underlying the construction of production orders belongs to the NP hard complexity class. Therefore, flexible computational solutions are needed. We claim that by using intelligence and collaboration in a multi-agent system (MAS), a correct solution is reached more efficiently. Intelligence is emulated by both learning and decision-making, achieved through a feed-forward artificial neural network (FANN). The FANN is embedded in a *machine agent*, which determines the appropriate machine to manufacture the product. Collaboration is obtained by employing a sound protocol based on FIPA-ACL messages. We illustrate the approach by designing and implementing a MAS, which is already in use in a company that produces *labels*.

Keywords Multi-agent system · Production activity control · Supervised learning · Production orders

I. Villar-Medina · O. López-Ortega (✉) · R. Hernández-Gómez
Instituto de Ciencias Básicas e Ingeniería,
Centro de Investigación en Tecnologías de Información y Sistemas, Universidad Autónoma del Estado de Hidalgo,
Carretera Pachuca - Tulancingo Km. 4.5, Pachuca,
Hidalgo, Mexico
e-mail: lopezo@uaeh.reduaeh.mx

I. Villar-Medina
e-mail: villar_israel@yahoo.com

R. Hernández-Gómez
e-mail: armando@uaeh.edu.mx

1 Introduction

Manufacturing companies crave for supporting software that helps achieve flexibility to compete in the globalized marketplace. Multi-agent systems (MAS) provide attractive features to obtain a better performance of manufacturing activities. Mönch [19] employs the term *agent-based manufacturing* to emphasize the impact of this paradigm because activities such as process planning [13, 14], holonic control, and production management are being supported by multi-agent systems. Production management systems are particularly important since they link planning to manufacturing by building and dispatching production orders [5]. However, the classical approach is mostly unfit in dealing with the exponential number of combinations to generate the *correct* production order as input data is not necessarily homogeneous. Hence, flexible approaches open immense opportunities to improve this manufacturing activity as reported in [25], where a genetic algorithm has been employed to dispatch orders in a dispersed manufacturing setting.

Even though agent-based manufacturing can enhance flexibility of production management systems by incorporating artificial intelligence techniques (i.e., supervised learning), this potential has yet to be fully exploited. Supervised learning is achieved primarily through feed-forward artificial neural networks (FANN), with the back-propagation algorithm [8]. Consequently, modeling and implementing *learning agents* to manage more efficiently changing market conditions contribute to the current state of the art in production management systems.

In this paper we elaborate on how to model and implement a FANN that has been trained with the back-propagation algorithm. This intelligent agent, known as *machine agent*, is part of a multi-agent system, of which we describe briefly its global design and operation. This article

is organized as follows: A discussion of the related work is presented in Sect. 2. The transition from a classical production planning system to an agent-based approach is detailed in Sect. 3. The case-study is presented in Sect. 4. The description of the implementation process of the learning agent is presented in Sect. 5, and finally, the conclusions of the project are drawn.

2 Related work

2.1 Multi-agent systems in production environments

Multi-agent systems in production environments have been used in varied forms. Julka [9] reported an agent-based supply chain management system. Symeonidis et al. [23] employ a multi-agent system in order to plan enterprise resources through data mining techniques. Optimization on production planning is also achieved with a MAS [10]. Similarly, Kornienko [11] accomplished a significant reduction of time and movement by using intelligent agents that assign products to the appropriate machine. Wang [24] reports the use of intelligent agents to determine a process plan and investigates the necessity to combine process planning and scheduling [26]. A MAS aimed at planning enterprise-wide resources is depicted in [12]. Park [20] also introduces the notion of a coordinator agent to smoothen the activities being performed in a manufacturing setting. Agent systems have been also developed to optimize the utilization of resources within existing manufacturing systems and deal with changing demand and products [1]. An agent-based architecture is assigned to support the scheduling of distributed manufacturing resources in a virtual enterprise [27]. Finally, recent work focuses on how to mix Web services and agents to integrate inter-organizational business processes [6, 22]. Mahesh [15], shows a generic structure of a MAS to be used at the production planning level.

2.2 Neural networks in production systems

The application of neural networks in production systems has been done successfully. Paternina [21] reports learning algorithms to schedule multiple products on a single server. The solution is validated via simulation. Fichtner [7] describes an unsupervised learning technique to determine the appropriate NC machine, having as input unknown design features coming from a CAD system.

2.3 Order dispatching approaches

Order dispatching systems have been an interesting topic of research since they comprise the link between production planning and manufacturing execution. This type of system can be seen embedded on MRP systems or as supporting software for the supply chain. Such is the case of [16, 17]. The author describes object-oriented models for the design and implementation of the order dispatcher system. Furthermore, its findings are implemented as ready-to-use software (http://orion.com).

Although research papers are highly valuable, our proposal differs on three different areas. One is the proposal of an agent-based production management system, as opposed to the classical approach. Secondly, we pioneer on how to incorporate supervised learning in software agents to deal with complex decisions in a manufacturing environment. The third difference consists of the communication of asynchronous decisions rather than using top-down algorithms, as in the case of other order promising systems.

3 Agent-based production planning

Marík [18] acknowledges the inclusion of agent technology on the production planning level, stating that the *agentification* process associated with installing agents into the production planning department provides an elegant mechanism for system integration and supports the migration from centralized planning towards distributed and flexible architectures. To asses this transition, Figs. (1) and (2) show the classical approach and the agent-based approach, respectively.

The classical approach to production planning is a quantitative measure of requirements determined by the time-phased explosion of a top-level production plan. This includes the determination of material and capacity requirements, balancing incoming orders and forecasts taking into account available material and capacity. The primary

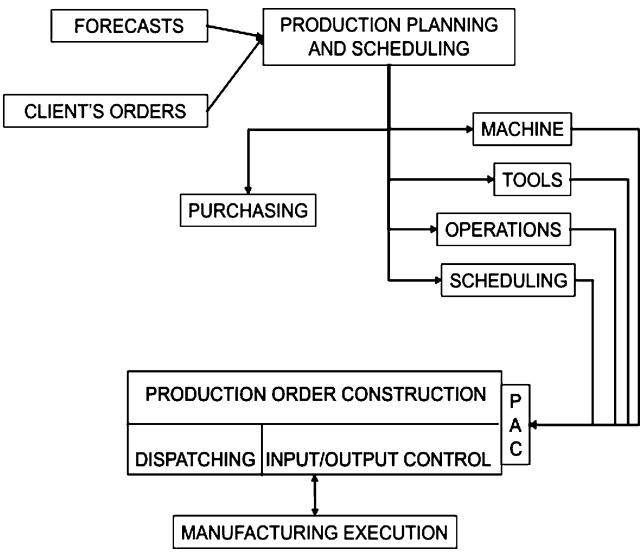

Fig. 1 Production planning: the classical approach

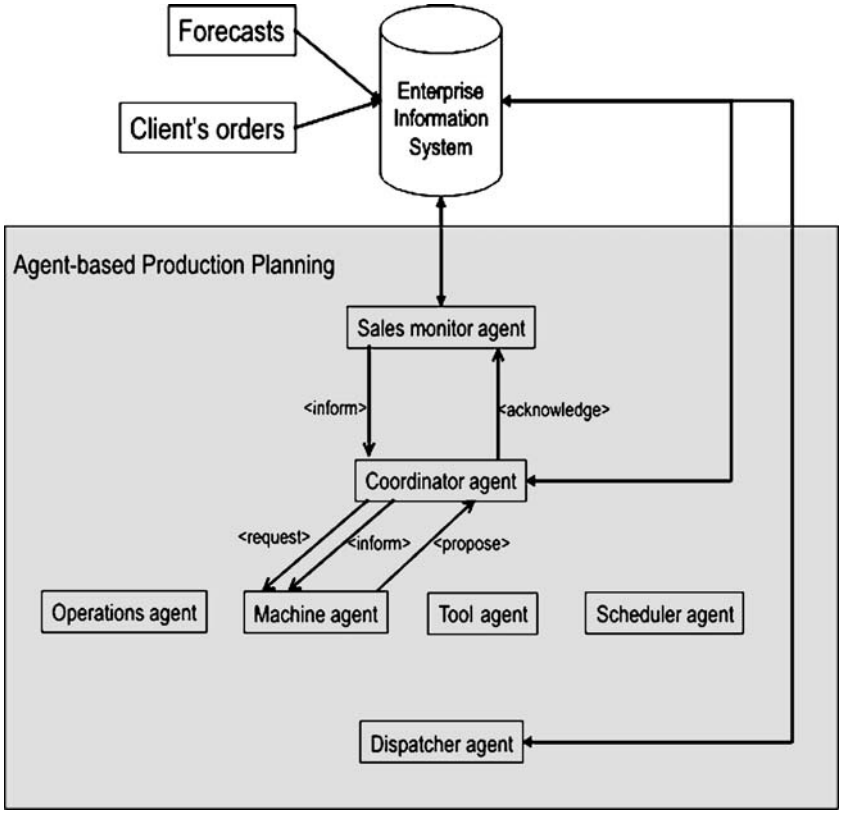

Fig. 2 Production planning: the agent-based approach

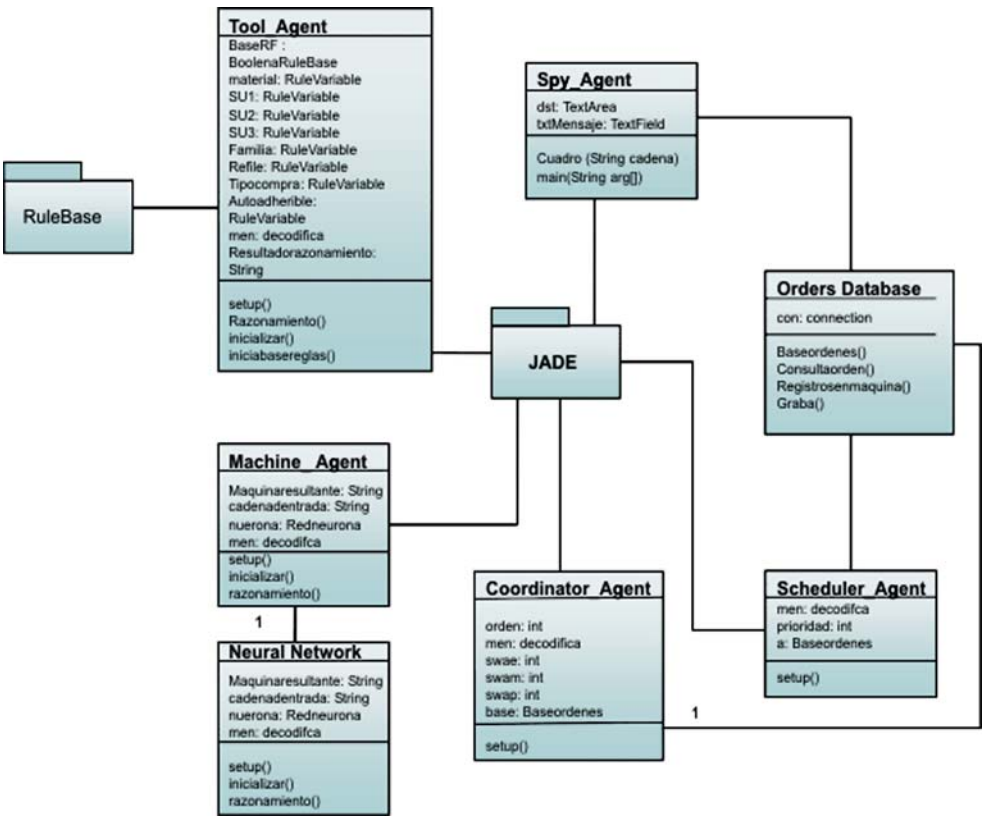

Fig. 3 Structure of the multi-agent system

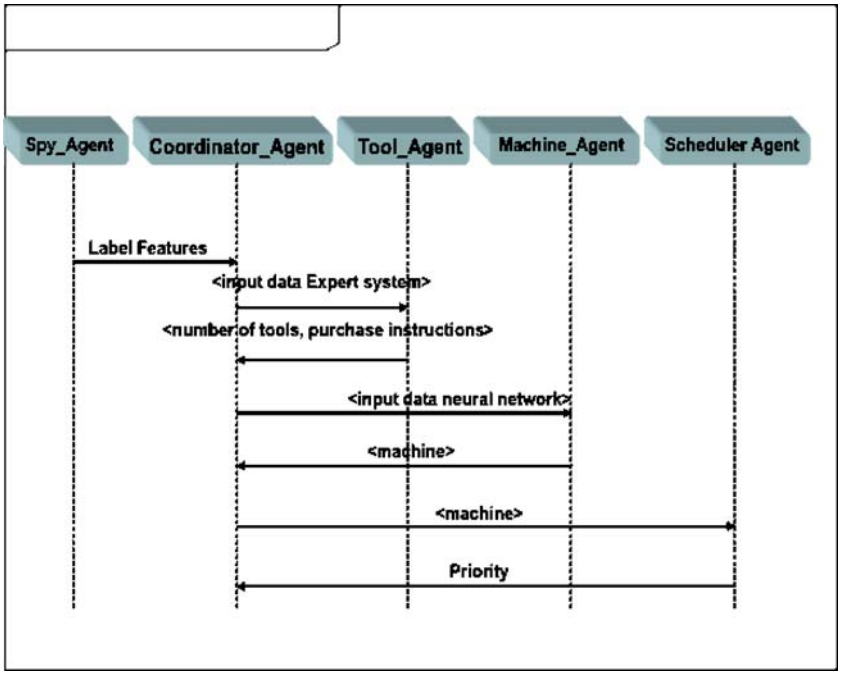

Fig. 4 Communication protocol

objective is to balance two opposing forces: what is planned to be produced and the actual capacity to produce. Despite the set of well-structured algorithms to plan production, any minor change in the environment (a new incoming order, a cancellation, or the inability to acquire raw material at the defined time) provokes re-running a series of time-consuming processes, from shaping a new top-level plan to accommodating changes in the material and capacity requirement planning.

On the other hand, an agent-based production management system facilitates communication among decision-makers (i.e., the software agents). Also, the process of making a decision is distributed, which also permits asynchronuous and parallel processes to be executed.

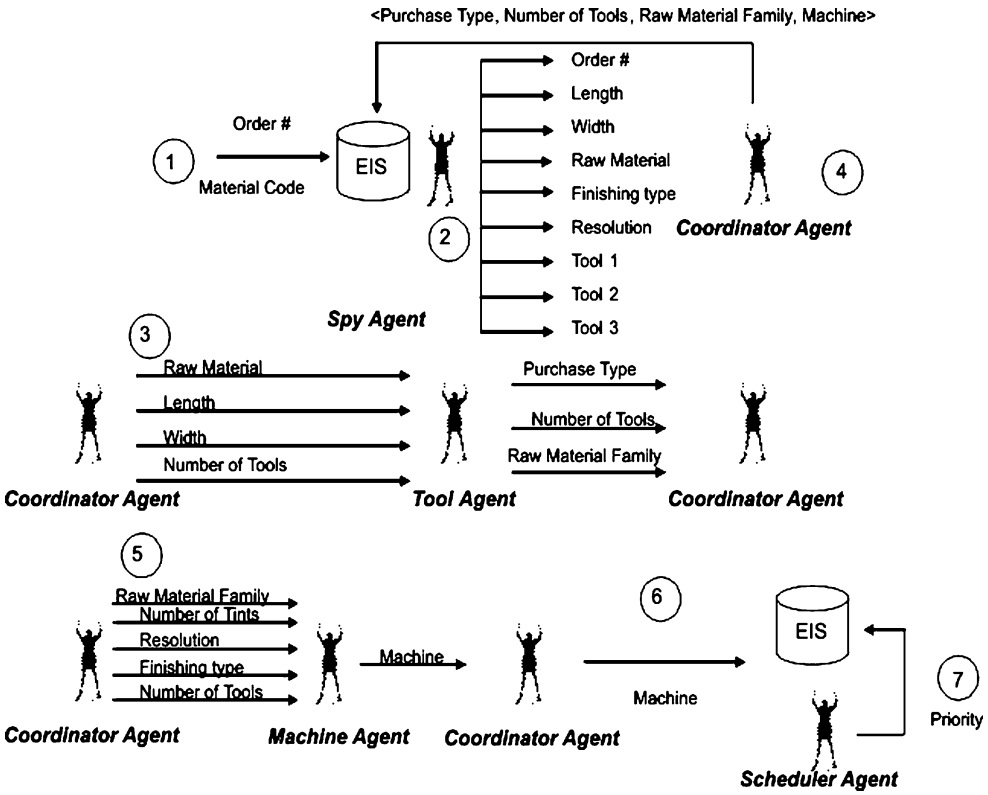

Fig. 5 The case study

Collaboration among autonomous software units is possible by means of FIPA-ACL *performatives*.

Let a communication act between two agents be a tuple consisting of:

```
<Performative       Sender       Receiver      Message_Content>
```

```
      <Request      Coordinator_Agent   Machine_Agent       machine>
<Inform       Coordinator_Agent   Machine_Agent       product_requirements>
     <Propose      Machine_Agent       Coordinator_Agent   machine_id>
```

Therefore, the communication within agent-based production planning possesses the following structure:

Another appealing feature is the promise to have intelligence as part of the agents' inherent capabilities. The mechanism to create a meaningful message content is not only a query to a database (which can also be used, though) but also a process that emulates intelligent decision-making. Moreover, data communication can be achieved by speech acts, giving the impression that two or more software agents are performing human-like communication. The multi-agent

Fig. 6 Training matrix for the FANN

INPUTS					OUTPUT
RAW MATERIAL FAMILY	NUMBER OF TINTS (DPI)	RES TYPE	FINISH	NUMBER OF TOOLS	MACHINE
CLOTH	0	0	NONE	1	830
CLOTH	0	280	NONE	1	830
CLOTH	1	280	NONE	1	830
CLOTH	2	360	NONE	1	830
CLOTH	3	360	NONE	1	830
CLOTH	3	400	LAMINATED	1	2200
CLOTH	3	400	LAMINATED	2	NILPETER
CLOTH	3	400	LAMINATED	2	NILPETER
CLOTH	4	700	NONE	2	2200
CLOTH	4	700	NONE	2	2200
CLOTH	4	700	NONE	2	2200
CLOTH	7	800	LAMINATED	2	NILPETER
CLOTH	7	800	NONE	2	NILPETER
CLOTH	7	800	NONE	1	NILPETER
CARDBOARD	0	280	NONE	1	830
CARDBOARD	1	360	NONE	1	830
CARDBOARD	3	360	NONE	1	830
CARDBOARD	4	500	LAMINATED	1	2200
CARDBOARD	4	360	NONE	1	2200
CARDBOARD	5	600	LAMINATED	1	2200
CARDBOARD	6	700	NONE	2	NILPETER
CARDBOARD	5	800	NONE	3	NILPETER
CARDBOARD	6	700	LAMINATED	2	NILPETER
CARDBOARD	5	800	LAMINATED	2	NILPETER
KIMDURA	0	360	NONE	1	2200
KIMDURA	1	400	LAMINATED	1	2200
KIMDURA	1	400	LAMINATED	2	2200
KIMDURA	1	400	LAMINATED	1	2200
KIMDURA	5	360	NONE	1	NILPETER
KIMDURA	6	360	NONE	1	NILPETER
BOPP	0	280	NONE	1	830
BOPP	1	360	LAMINATED	1	830
BOPP	3	400	NONE	1	2200
BOPP	3	400	LAMINATED	1	2200
BOPP	5	500	NONE	1	2200
BOPP	6	700	NONE	1	2200
BOPP	7	800	NONE	2	NILPETER
PAPER	0	280	NONE	1	830
PAPER	0	280	NONE	1	830
PAPER	1	360	NONE	1	830
PAPER	2	360	NONE	1	830
PAPER	3	500	NONE	1	2200
PAPER	3	600	SULFATED	1	2200
PAPER	3	700	LAMINATED	2	2200
PAPER	3	600	LAMINATED	2	2200
PAPER	6	360	LAMINATED	1	NILPETER
PAPER	5	700	NONE	2	NILPETER
PAPER	6	800	NONE	3	NILPETER
OTHER	0	0	NONE	1	2200
OTHER	4	500	NONE	3	NILPETER

system we propose contributes to achieve this task. To the best of the authors' knowledge, this is the first report on using supervised learning as a mechanism to improve production planning. In the following sections we provide details on the design and implementation of the agent-based production planning system illustrated in this paper.

4 Implementation of the MAS

4.1 The multi-agent system: structure and communication

Figure (3) presents the actual structure of the multi-agent system. The Agent Unified Modeling Language [2] has been employed to describe the architecture and the communication protocol. On the implementation side, JADE [3] is employed as the agent platform. The following agents integrate the architecture: (i) coordinator agent, (ii) machine agent, (iii) tool agent, (iv) spy agent, and (v) scheduler agent.

In the proposed design, both the *machine agent* and the *tool agent* possess intelligent capabilities. The *tool agent* has a rule-based system to determine the right tooling, whereas the *machine agent* has a FANN. Previous work on JADE agents with a rule-base system can be consulted in [14]. The activities performed by the multi-agent system are explained succinctly. The *coordinator agent* is in charge of directing the messages among the agents assemble. The *spy agent* has the responsibility to read the Enterprise Information System at fixed intervals. The *scheduler agent* decides the sequence to release the production orders. The communication protocol illustrates how the set of agents communicate partial decisions at every phase of the process to build production orders (Fig. 4).

Communication between the *coordinator agent* and the *machine agent* is not initiated until the *tool agent* informs its results. The *coordinator agent* informs five pieces of data to the *machine agent* (see Fig. 7). The following code illustrates how the *coordinator agent* informs the *machine agent* the value for the raw material family and appends a message in a GUI.

```
//the local name of the machine agent is am
if ((men.Variable.compareTo("FA=")==0)&&(men.Valor.compareTo("")!=0) && am==0 )
{
        msg2.setContent(men.Mensaje);
        msg2.addReceiver( new AID( "am" , AID.ISLOCALNAME) );
        send(msg2);
        esp.GUI_de_la_orden.areaMensajes.append("\n<COORDINADOR>        to        MACHINE_AGENT:
        "+msg.getContent());;
        Raw_Material:Family = men.Valor;
}
```

The *machine agent* initiates its reasoning process and then informs the *coordinator agent* about what machine to use. The actual content that is uttered by the *machine agent* is a valid conclusion obtained by the FANN attached to it.

```
ACLMessage msg2 = new ACLMessage(ACLMessage.INFORM);
inititate_back_pro();
the_machine = reasoning_back_pro();
msg2.setContent("MR="+ the_machine);
msg2.addReceiver( new AID( "coordinator", AID.ISLOCALNAME) );
send(msg2);
```

The *machine agent* first initiates the neural network by setting it to execution mode. This happens when the initiate_back_pro() method is reached. The resultant machine is attached to a string variable the_machine, which receives the decoded value obtained by the neural network when the reasoning_back_pro() method is called. The resultant machine is set as a content of the INFORM performative, which is sent to the *coordinator agent*.

When the machine, tools and other data are established, a priority is assigned to the production order. The production order is stored in the Enterprise Information System when a priority is assigned. Then a message is transmitted to the manufacturing department to start production. Subsequently, we describe the design and implementation process of the supervised learning mechanism.

4.2 Dynamics of the MAS in the case study

The agent-based production planning system is implemented as supporting software in a factory dedicated to produce *labels*. Labels are used on diverse products such as wines or sodas, candies, jeans, bar-coding, CD's, etc. They are a highly demanded product in the marketplace. However, their characteristics make them prone to changes. Variation in size, colors,

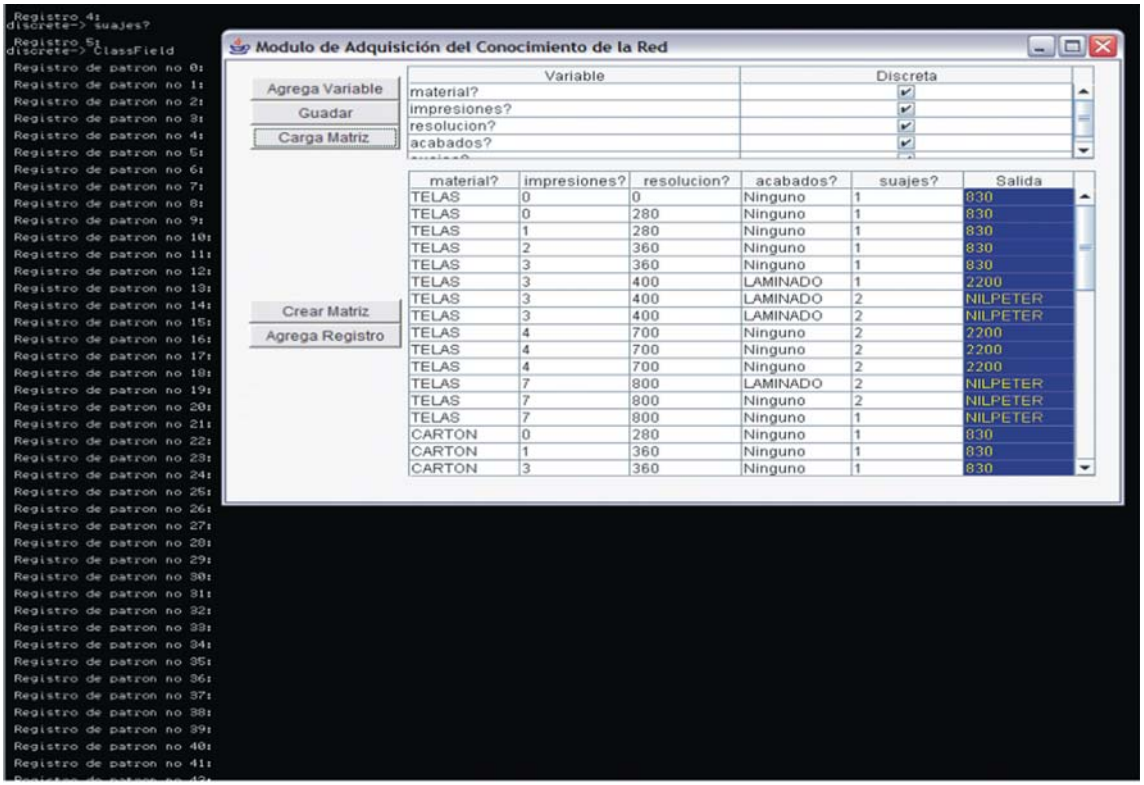

Fig. 7 Training module for the FANN

or raw material impacts the entire manufacturing system. Few companies have the capacity and expertise to produce them, and those that manufacture labels face tremendous production planning problems, because systems based on the MRP approach can not handle efficiently such complexity.

The labels are produced according to clients' requirements. Once the client's order is entered by the sales department, the production planning department must elaborate a *production order*, on which the information to produce the label is comprised. The entire communication process is presented in Fig. (5).

There are many combinations of input data to produce a *label*. Raw material can be chosen from the following options: glued-paper, non-glued paper, several types of cloth, plastic, cardboard, to name but only a few. Also, every raw material is purchased differently, either on continuous cylinders or in batches (also called master). Another complexity arises when a client sets the colors to be used. Thus, the combinations of tints is fixed according to the client requirements. The production planning system must establish what tools (SU1, SU2, and SU3) are to be executed to comply with the client's required design of the label. On such information, the manager must determine the number of tools (up to three).

One of the key elements needed to construct a correct production order is the determination of the machine. The production planning system must provide the right machine (out of three options) which will produce the label. The machine depends on the following variables: raw material family, number of tints, resolution (dots per inch), type of

Table 1 Codification of the *Raw Material Family* set

Input stream						Raw Material Family
0	0	0	0	0	1	Paper
0	0	0	0	1	0	Fabric
0	0	0	1	0	0	Kimdura
0	0	1	0	0	0	Cardboard
0	1	0	0	0	0	BOPP
1	0	0	0	0	0	Other

Table 2 Codification of the *Tints* set

Input stream							Number of tints
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	2
0	0	0	0	1	0	0	3
0	0	0	1	0	0	0	4
0	0	1	0	0	0	0	5
0	1	0	0	0	0	0	6
1	0	0	0	0	0	0	7

Table 3 Codification of the *Resolution* set

Input stream								Resolution
0	0	0	0	0	0	0	1	Not specified
0	0	0	0	0	0	1	0	280
0	0	0	0	0	1	0	0	360
0	0	0	0	1	0	0	0	400
0	0	0	1	0	0	0	0	500
0	0	1	0	0	0	0	0	600
0	1	0	0	0	0	0	0	700
1	0	0	0	0	0	0	0	800

finishing, and number of tools. In spite of the production manager expertise, production orders normally contain imprecise machine data. The production manager either assigns a machine that is not suitable to process the raw material, or s/he launches a production order for a machine that is not available within the time framework. This situation might occur due to fatigue, or the inability of a human being to manage a large number of combinations. Therefore, the process of establishing the right machine is critical. The following section provides details on the solution presented.

5 Learning to make decisions

As it has been stated before, obtaining the appropriate machine depends on the combination of five different data sets as follows:

Raw_Material_Family **X** *Tints* **X** *Resolution* **X** *Finishing_Type* **X** *Number_Tools -> Machine*

Each input data set contains the following elements:

Raw_Material_Family = {Fabric, Paper, Kimdura, Cardboard, BOPP, Other}

Number_ Tools = {1, 2, 3}

Tints = {1, 2, 3, 4, 5, 6}

Resolution = {not specified, 280, 360, 400, 500, 600, 700, 800}

Finishing_Type = {not specified, laminated, sulfated}

The output data set is formed by:

Machine = {830, 2200, Nilpeter}

Table 4 Codification for the *Finishing Type* set

Input stream		Finishing type
0	0	Not specified
0	1	Laminated
1	0	Sulfated

Table 5 Codification for the number of tools set

Input stream		Number of tools
0	0	1
0	1	2
1	0	3

We chose the feed-forward artificial neural network architecture because such nets are known to be good classifiers. In order to train the net, a training matrix is employed (Fig. 6). Such a training matrix is actually a subset of the entire range of possibilities to set a machine.

The FANN was provided to the *machine agent* by means of Java code, adapting the work of [4] on Java coding of back-propagation. To facilitate the learning stage, we developed a training module, see Fig. (7). The values associated with the training matrix are stored in text file, which contents are read by the neural_network class attached to the *machine agent* (Fig. 3). The resultant weights of the net are kept in a binary file, which is used every time the neural network is set into execution mode.

The actual configuration of the FANN is related to the training matrix. Specifically, one processing unit (p.u.) is created for each value contained in the training set. For example, as Input 1 of the training matrix has six different values, six processing units are built. However, the actual value is stored on a variable as a string of characters, which is not a suitable input for the FANN. Thus, such a string is converted into a stream of 0s and 1s. Table 1 illustrates the codification for the Raw Material Family:

The prior codification is necessary because FANNs only handle values within the closed interval [0,1]. Therefore, the actual input and output values that the net receives and obtains are 0s and 1s. Therefore, the FANN has six processing units in the input layer, which are in charge of dealing exclusively with the Raw Material Family. The totality of discrete values contained in the input and output sets were codified in a similar way. Tables 2, 3, 4, 5 and 6 show the resultant codification.

Consequently, the number of processing units in the input layer of the FANN equals the number of codified input values. For this case, 25 processing units in the input layer are set. Once the processing units for the input layer

Table 6 Codification of the *Machine* set

Input stream			Machine
0	0	1	830
0	1	0	2200
1	0	0	Nilpeter

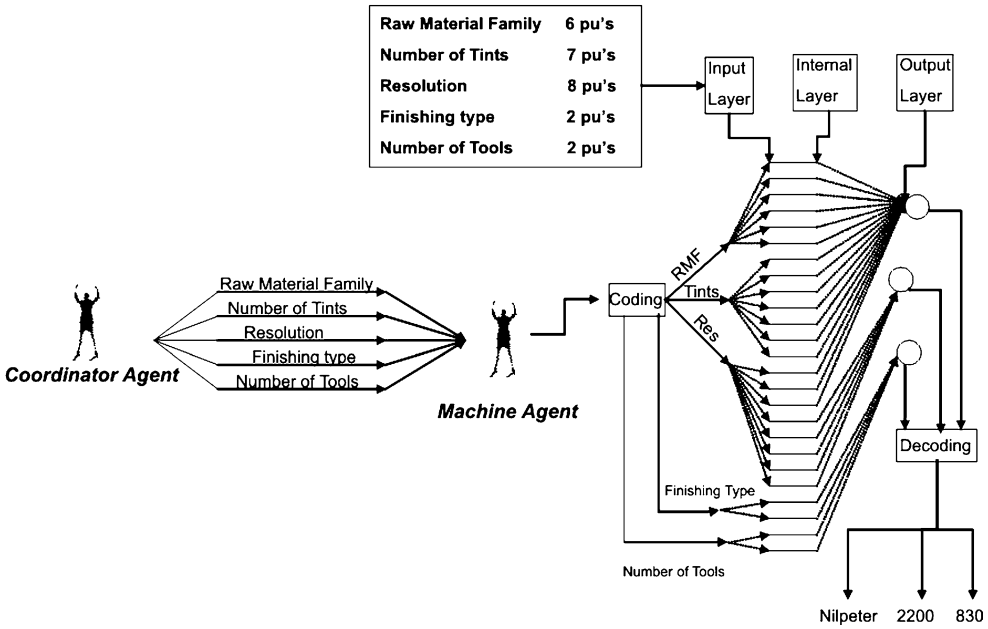

Fig. 8 Communication and inference to obtain a machine

are generated, then the intermediate layer is set. The Java code assigns the same number of processing units in the input layer to the intermediate layer. Only one intermediate layer is created since a FANN with a single intermediate layer is powerful enough to deal with non-linear problems. The output layer is then constructed.

The number of processing units in the output layer also depends on the number of values that are provided as valid

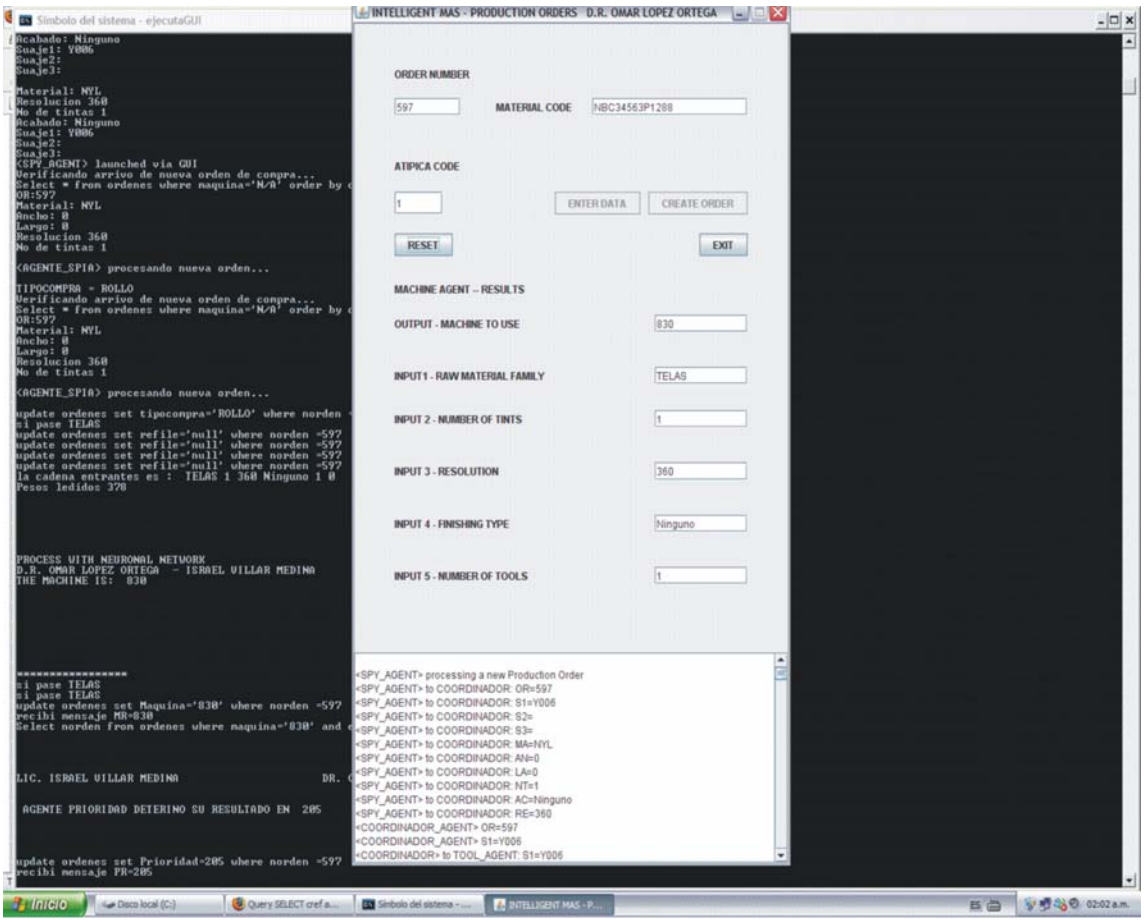

Fig. 9 GUI of the MAS

outcomes in the training set. In our training matrix, the output layer is given three different values, and three processing units are constructed.

According to the previous codification, the resultant FANN consists of 3 layers. The input and intermediate layers are comprised of 25 processing units each, whereas the output layer possesses three processing units. This fact is represented in Fig. (8). The processing units on the input layer receive streams of 0s and 1s, which result after codifying the incoming symbol. The reverse process is carried out for the output layer, where the processing units provide 0s and 1s, all of which are further converted into the symbol that represents a machine.

Figure (9) presents the GUI attached to the *machine agent*. The order number, the material code, and the *atipica* code (i.e., a control code) are the data to be entered into the system. Once the user inputs the data, the EIS is updated. At this moment, the *spy agent* is reading a new incoming order, and it then notifies the *coordinator agent*. After the user clicks the button CREATE ORDER, the *coordinator agent* changes status from idle to active. As soon as the *coordinator agent* is active, it initiates the communication process with the *tool agent* and the *machine agent*.

The *machine agent* receives the five pieces of data necessary to obtain a valid machine. This can be seen in the GUI. The text area shows the exchange of messages that actually takes place. When the machine is set, the *coordinator agent* acknowledges the reception of the last bit of data to form a production order. Shortly afterwards, the *scheduler agent* is ready to give the order a place in the dispatching queue.

6 Conclusions

In this paper the authors have described the implementation of a supervised learning technique in a multi-agent system aimed at improving the construction of production orders. This activity represents the link between manufacturing planning and execution, and it is an NP-hard combinatorial problem due to the ample range of possibilities to actually construct a valid production order. The use of a FANN has given positive results. The resultant net was trained with a subset of the entire range of combinations, and the resultant weights have been further used to make inferences. Input data to the network is transmitted via a *coordinator agent*, in charge of directing messages among the entire ensemble of agents. The resultant MAS fully automates the process to acquire client's data and build the production order. We envision future work in several directions. One line of research is the incorporation of unsupervised learning techniques. With this approach, the training phase of the net could be avoided, and the network can adapt itself to changes in the manufacturing setting, i.e., when a new machine is added to the manufacturing system. Another line of research is the incorporation of intelligent agents to forecasting. This will help the transition to agent-based manufacturing from classical software modules without compromising the actual organization of manufacturing companies.

Acknowledgements The authors thank Francisca Tanca for proof-reading and corrections.

References

1. Anosike AI, Zhang DZ (2007) An agent-based approach for integrating manufacturing operations. Int J Prod Econ (in press)
2. Bauer B, Odell J (2005) UML 2.0 and agents: how to build agent-based systems with the new UML standard. Eng Appl Artif Intell 18:141–157
3. Bellifemine FL, Caire G, Greenwood D (2007) Developing multi-agent systems with JADE. Wiley, New York
4. Bigus J (2002) Constructing intelligent agents using Java, 2nd edn. Wiley, New York
5. Browne J, Harhen J, Shivnan J (1992) Production management systems. A CIM perspective. Addison-Wesley, UK
6. Feng SC (2005) Preliminary design and manufacturing planning integration using Web-based intelligent agents. J Int Manuf 16:423–437
7. Fichtner D, Nestler A, Dang T, Schulze A, Carlsten U, Schreiber S, Lee S (2006) Use of agents and neural networks for acquisition and preparation of distributed NC information to support NC planning. Int J Comput Integr Manuf 19:581–592
8. Jang J-SR, Sun C-T, Mizutani E (1997) Neuro-fuzzy and soft computing. a computational approach to learning and machine intelligence. Prentice-Hall, Englewood Cliffs, NJ
9. Julka N, Srinivasan R, Karimi I (2002) Agent-base supply chain management -1: framework. Comput Chem Eng 26:1755–1769
10. Karageorgos A, Mechandjiev N, Weichhart G, Hämmerle A (2003) Agent Based optimization of logistics and production planning. Eng Appl Artif Intell 16:335–348
11. Kornienko S, Kornienko O, Priese J (2004) Application of multi-agent planning to the assignment problem. Comput Ind 54:273–290
12. Lea BR, Gupta MC, Yu WB (2005) A prototype multi-agent ERP system: an integrated architecture and a conceptual framework. Technovation 25:433–441
13. López-Morales V, López-Ortega O (2005) A distributed semantic network model for a collaborative intelligent system. J Intell Manuf 16:515–525
14. López-Ortega O, López-Morales V (2006) Cognitive communication in a multi-agent system for distributed process planning. Int J Comput Appl Technol 26:99–107
15. Mahesh M, Ong S-K, Nee A-Y-C, Fuh J-Y-H-, Zhang Y-F (2007) Towards a generic distributed and collaborative digital manufacturing. Robot Comp Integr Manuf 23:267–275
16. Makatsoris HC, Chang YS (2004a) Design of a demand driven collaborative supply-chain planning and fulfillment system for distributed enterprises. Prod Plann Control 15:256–269
17. Makatsoris HC, Chang YS, Richards HD (2004b) Design of a distributed order promising system and environment for a globally dispersed supply chain. Int J Comp Integr Manuf 17:679–691
18. Maríк V, Lazanský J (2006) Industrial applications of agent technology. Control Engeneering Practice, DOI 10.1016/j.conengprac.2006.10.001. (in press)

19. Mônch L, Stehli M (2006) ManufAG: a multi-agent system framework for production control of complex manufacturing systems. Inf Syst Elec B DOI 10.1007/s10257-005-0030-5
20. Park S, Sugumaran V (2005) Designing multi-agent systems: a framework and application. Expert Syst Appl 28:259–271
21. Paternina-Arboleda C-D, Das T-K (2005) A multi-agent reinforcement learning approach to obtain dynamic control policies for stochastic lot scheduling problem. Sim Mod Prac Theory 13:389–406
22. Shen W, Qi H, Shuying W, Yinsheng L, Hamada G (2006) An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. Robot Comp Integr Manuf 23:315–325
23. Symeonidis A-L, Kehagias D, Mitkas P (2003) Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques. Expert Syst Appl 25:589–602
24. Wang L, Shen W (2003) DPP: an agent-based approach for distributed process planning. J Intell Manuf 14:429–439
25. Wang Q, Yung K-L, Ip W-H (2005) An order dispatcher for dispersed manufacturers solved by a mixed variable hybrid genetic algorithm. Int J Comp Integr Manuf 18:41–52
26. Wang L, Shen W, Hao Q (2006) An overview of distributed process planning and its integration with scheduling. Int J Comput Appl Technol 26:3–14
27. Wang D, Nagalingam S-V, Lin G-C-I (2007) Development of an agent-based Virtual CIM architecture for small to medium manufacturers. Robot Comp Integr Manuf 23:1–16