19 OPTIMIZATION ALGORITHM FOR DYNAMIC MULTI-AGENT JOB ROUTING

Leonid Sheremetov[1], Luis Rocha[1], Juan Guerra[2], Jorge Martinez[2]
[1] *Mexican Petroleum Institute,*
Av. Lazaro Cardenas, 152, Col.San Bartolo Atepehuacan, México, D.F., CP 07730,
[2] *Computer Science Research Centre of National Technical University (CIC-IPN),*
Av. Juan de Dios Batiz esq. Othon de Mendizabal s/n Col. Nueva Industrial Vallejo,
México, D.F., C.P. 07738, MEXICO
e-mail: sher@cic.ipn.mx

Current research in multi-agent heterarchical control for holonic systems is usually focused in real-time scheduling algorithms, where agents explore the routing or process sequencing flexibility in real-time. In this paper we investigate the impact of the dynamic job routing and job sequencing decisions on the overall optimization of the system's performance. An approach to the optimization of local decisions to assure global optimization is developed within the framework of a Neural Collective Intelligence (NECOIN). Reinforcement learning (RL) algorithms are used at the local level, while generalization of Q-neural algorithm is used to optimize the global behaviour. A simulation test bed for the evaluation of such types of multi-agent control architectures for holonic manufacturing systems integrating discrete-event simulation facilities is implemented over JADE agent platform. Performance results of the simulation experiments are presented and discussed.

1. INTRODUCTION

The worldwide competition and the highly specified customers' requirements towards product quality, delivery time, and services force the industry to a permanent optimization of the production. As a consequence, logistics gets a new focus on optimization of the production process in a very dynamic environment. Current research in multi-agent heterarchical control for holonic systems is usually focused in real-time scheduling algorithms, where agents explore the routing or process sequencing flexibility in real-time (Denkena et al., 2002, Heragu et al., 2002, Sheremetov et al., 2003, Usher, 2001). Though there are a lot of results on scheduling heuristics and dispatching rules, few researchers have studied the influence of these approaches on the overall optimization of the production system performance. Since most of these solutions and techniques are based on local optimization criteria, these decisions do not assure the overall business optimization at the global level because of the conflicts between the local goals (Julka et al., 2002). Traditional centralized techniques usually cannot assure global optimization either due to the inherent complexity of the problem.

In this paper, the problem of job routing (JR) is addressed within the context of the NEural COllective INtelligence (NECOIN) theory (Wolpert & Kagan, 1999) and the adaptation of the Q-neural algorithm (Rocha-Mier, 2002). According to our approach, agents construct previously unknown model of the environment through learning and interaction between them in a distributed fashion. The approach looks for balancing the agents' efforts to achieve the short-term or local goal (shortest path selection) and a long-term or global goal – overall production optimization (Shimbo & Ishida, 2003). According to our definitions, a production system within the NECOIN framework is a large multi-agent system where:

- Its objective is the decentralization of control and communication.
- Each entity of the system is represented as an agent with autonomous behaviour and a local utility function.
- The learning process consists of adapting the local behaviour of each entity (agent) with the aim of optimizing a given global behaviour.
- The agents execute Reinforcement Learning algorithms at the local level while generalization of Q-neural algorithm is used to optimize the global behaviour.

In this paper we investigate the impact of the dynamic job routing on the overall optimization of the system's performance. A simulation test-bed for the evaluation of such types of multi-agent control architectures for holonic manufacturing systems integrating discrete-event simulation facilities and implemented over JADE agent platform (AP) is described. This test-bed can be also used to compare different approaches to job routing on a common basis (Brennan & O, 2000). The case study deals with production of hypothetic products on the shop-floor level. Performance results of the simulation experiments are presented and discussed.

2. NECOIN framework for the JR problem

This work proposes a model of the production system (PS) within the framework of the NECOIN theory. In our approach, an agent can represent any entity of the PS. In contrary to the model described in (Sheremetov et al., 2003), the materials in the PS are represented as objects forming part of the environment. Therefore, every agent can change or influence these environment objects. The details of the objects are stored as attributes. We define the following elements within the NECOIN framework for the JR problem:

- Order-agent that has the knowledge on final products orders: PO
- Set of n machine-agents (MA): $M = \{M_1, M_2, ..., M_n\}$
- Set of s operations executed by machine i: $OP_i = \{O_1, ..., O_S\}$
- Vector of non-negative values of r features for each operation $O_i : \vec{V_i} = < v_1^i, ..., v_r^i >$, *e.g.* v_1^i = average time. These features vary from one machine to another.
- Set of n storage-agents (SA) denoting raw material providers: $S = \{S_1, S_2, ..., S_n\}$
- Set of s objects corresponding to a type of raw material: $MP = \{MP_1, ..., MP_S\}$
- Set of n final product storage agents (FPSA): $FP = \{FP_1, ..., FP_n\}$

- Set of n objects corresponding to a type of final product: $P = \{P_1,...,P_n\}$
- Vector of non-negative values of r features for each product $P_i : \overrightarrow{PV_i} =< pv_1^i,...,pv_r^i >$, *e.g.* pv_1^i - product priority.

In this work, each agent has the following features:

- The set of environment states $X = \{x_1, x_2, x_3,...\}$. Knowledge (usually incomplete) about other agents is considered to be part of the environment state. For example, in some cases a MA might make decisions without knowledge that a supplier has frequently failed on due dates.
- The capacity of agent to act is represented as a set of actions: $A_i = \{a_1, a_2,..., a_k\}$.
- The relationships between the agents in the PS are defined by: $R = \{r_1, r_2, r_3,...\}$. For each neighbour agent, the following parameters are considered: a) its relationship to the current agent (customer, supplier), b) the nature of the agreement that governs the interaction (production guarantees), and c) the inter-agent information access rights (the agent's local state to be considered during the decision-making process).
- The priorities of every agent are represented by: $Q = \{q_1, q_2, q_3,...\}$ These priorities can help in sequencing incoming messages for processing.
- The local utility function (LUF) is represented as follows: $Q_{(x(t),a_{x(t)})}(t+1) = Q_{(x(t),a_{x(t)})}(t) + \alpha\left[r(t+1) + \gamma \min_{a_{x(t+1)}} Q_{(x(t+1),a_{x(t+1)})}(t+1) - Q_{(x(t),a_{x(t)})}(t)\right]$, where: α is learning rate, γ is reduction rate.
  This equation represents the Q-learning (Sutton et al., 1998) equation used in RL. The Q-values $Q_{(x(t),a_{x(t)})}$ give an estimation of the PS. The way in which the Q-values are updated can be considered as one of the most important problems to solve in our framework. The reinforcement for the performed action is represented by $r(t+1)$. This function of reinforcement represents the partial time of product production and is composed of: a) transition time, b) waiting time, and c) operation time.
- The set of control elements: $C = \{c_1, c_2, c_3,...\}$. A control element is invoked when there is a decision to be made while processing a message. For example, in order to determine each destination of materials, a routing-control algorithm would be utilized.
- Every agent has a message handler responsible for communication.

3. Q-Neural Algorithm for Job Routing Task

To address the JR problem, the adaptation of the Q-neural algorithm (Rocha-Mier, 2002) is proposed and described. The behaviour of the Q-neural was inspired by the Q-routing algorithm (Littman & Boyan, 1993), the theory of NECOIN and the algorithms based on the behaviour of the colonies of ants. Learning is done at two levels: initially, at the agent's level locally updating the Q-values by using a RL rule, then, globally at system level by the utility function's adjustment. The control messages allow updating knowledge of the PS by updating the Q-values, which are

approximated by using a function approximator (look-up table, neural network, etc.). In Q-neural, there are 5 types of control messages:

- *An 'environment-message' (flag_ret=1)* generated by an intermediate MA after the reception of a raw material if the interval of time ω has already passed.
- *An 'ant-message' (flag_ret=2)* generated by the FPSA according to the interval of time *w_ants* when a final product arrives at the final product storage.
- *An 'update-message' (flag_ret=3)* generated in the planning phase every *ε__update* seconds to ask the neighbouring MA for their estimates about the operations of the FP.
- *An 'update-back-message' (flag_ret=4)* generated after the reception of an update-message in order to accelerate learning of the environment.
- *A 'punishment-message' (flag_ret=5)* used to punish a MA using a congested resource.

The Q-neural algorithm includes 3 parts: planning, ant-message and punishment algorithms working as follows. Exploration can involve significant loss of time. In Q-neural, a mechanism of planning (within the meaning of this term in RL) was developed at the local level of each agent. This mechanism consists of sending an update-message every ε_update seconds. This update-message will ask for the Q-values estimates of all the products, which are known at that moment by the neighbours.

When an environment-message arrives at the FPSA, an ant is sent in return if the period of time ω_ants has already passed. This ant exchanges the statistics obtained on its way. When it arrives to the SA, it dies. The ant updates the Q-value of each MA through which the raw material passed before arriving at the FPSA.

In some cases, different MAs from the same tier can have the same best estimate (prefer the same route). If they act in a greedy way, congestion occurs in the queue. To avoid congestions, a MA must sacrifice its individual utility and use another route. In order to address this problem a punishment algorithm is developed forcing an agent who receives a punishment message to calculate the second best estimate.

Finally, the Q-neural algorithm is defined as follows:

Initialize at $t = 0$*: All the Q-values* $Q_{x(t),a_{x(t)}}$ *with high values, the RL parameters:*

$\alpha, \gamma, \text{exploration}, w, w\_ants$

REPEAT

Update the instant t

if *a raw material is received by machine* M_i

Read the input vector x *from the raw material header and environment variables*

Send the message to the agent M_x *where the raw material arrives with the value of the reinforcement function* $r(t+1)$ *and the estimation* $Q^{\mu}_{(x(t),a_{x(t)})}(t)$

Execute the operation O' *and choose the action* $a_{\bar{x}(t)} = M'$ *in function of the input vector* x *by using the strategy* ε_greedy *derived from* $Q_{(x(t),a_{x(t)})}(t)$

Send the raw material $a_{\bar{x}(t)} = M'$

At the next time step, receive the message from the machine M' *with the value of the reinforcement function* $r(t+1)$ *and the estimation* $Q_{x(t+1),a_{x(t+1)}}(t+1)$

Apply the Q-learning update rule:

$$Q_{(x(t),a_{x(t)})}(t+1) = Q_{(x(t),a_{x(t)})}(t) + \alpha\left[r(t+1) + \gamma \min_{a_{x(t+1)}} Q_{(x(t+1),a_{x(t+1)})}(t+1) - Q_{(x(t),a_{x(t)})}(t)\right]$$

REPEAT

Algorithm of planning ε_update

Algorithm of punishment

Algorithm of ants

Planning, ant-message and punishment algorithms are described in more details in (Rocha-Mier et al., 2004).

4. Implementation of the Q-neural Algorithm in the Multi-agent Framework

The above-described model has been implemented using JADE AP in order to test the performance of the developed algorithms. A generic layout of the simulated PS is shown in figure 1. The first tier consists of suppliers of raw materials and is represented by the central storage. Raw materials are distributed among the machines organized into several different tiers. For simplicity, we consider that the operation lists corresponding to each machine of the tier are identical. Finally, all the processed parts are stored at the storage of the final products.

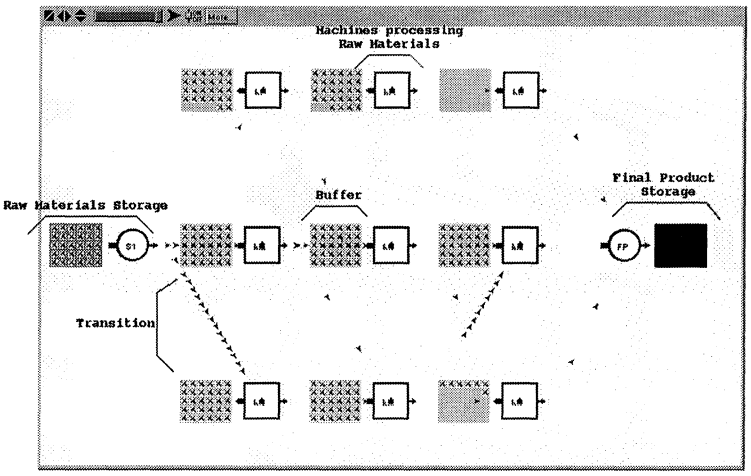

Figure 1 – A generic shop-floor layout scheme

An agent represents each entity in the model; there are *n* MAs at each layer, one SA and one FPSA. Also, there is a Scheduler Agent (SCHA) whose role is that of a synchronization ticker for the discrete event simulator organizing events during the simulation. Nevertheless, the event queue behaviour (currently owned only by this agent) can be transferred to the bulk of agents for an actual implementation in a

physical environment. A User Interface Agent (UIA) is used to define experiments, start, resume, and finish system's operation.

The simulation begins when the SCHA broadcasts a *startUp* message to the whole group. This way, agents can perform internal initialization tasks mainly related to the tables (known as Q-tables) and variables to be used by the Q algorithms. In the case of the SA, it will load the *Technological Processes* and *Orders* lists that will be processed. Also, it will reply to the SCHA with the initial *Raw Materials* list to be released.

Raw materials, intermediate products and final products are not physically present. The information concerning each of them is passed via message interchange between the SA and MA. A raw material becomes a final product after having travelled along the machine network. Initially, it is created with an operation vector that decreases at each step, until it gets empty and an FP is located at the corresponding storage.

Fig. 2 shows a Collaboration Diagram illustrating the planning phase of the Q-neural algorithm, which involves the SCHA and three MAs. Also, a sample raw-material transfer and the corresponding arrival to the FPSA are shown.

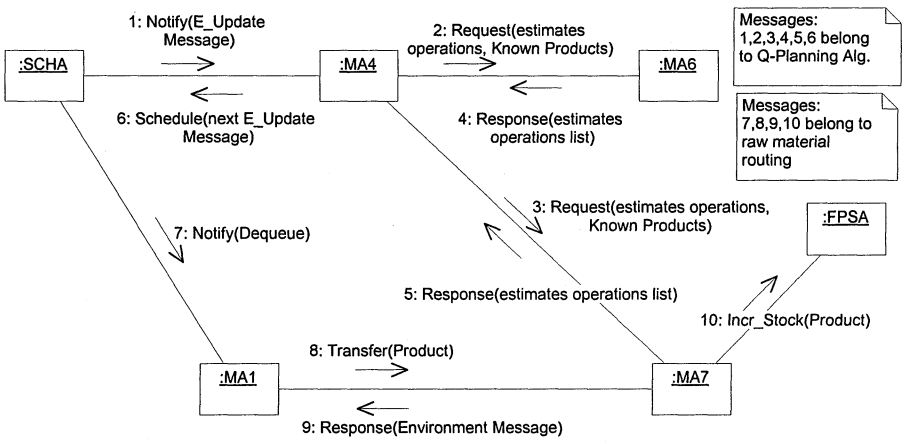

Figure 2 – Collaboration Diagram: planning phase of the Q-neural algorithm

As it can be noticed, both events are first dispatched by the SCHA. The SCHA warns MA each time they must take some product from their internal queue and get it processed. Before forwarding the modified product, an MA will consult the Q-tables in order to decide the best route to follow. Also, there is a ping-like operation that ensures the selected machine is still alive. Otherwise, the second best alternative will be chosen and the corresponding MA will be notified to queue the product for further processing. In consequence, the Q-tables are changed since a new product has been set in the link between the current machine and its neighbour.

Data back-propagation to the previous stage is achieved after a MA commits to queue a product. This mechanism helps ensuring that each machine has information to optimize product routing. Routing algorithms are embedded in MA's body. This results in information updating aimed in optimizing the decision making process on the best route selection for each stage.

It must be noticed that the SA works partially as a MA for the following reason: the SA also keeps a Q-table for making decisions on where to send the raw material after this is released. If the MA detects that there are no pending operations to accomplish for the processed product, it will notify the FPSA. In turn, this agent will modify the stock numbers for such product.

Under a centralized approach, a single entity evaluates continuously the best alternative for every agent's strategy within the system. However, computing the numbers for a large group could be unfeasible under time and resources constraints. Under our distributed approach, each agent shares its local estimates with the neighbours, thereby laying the ground for a collective solution with fewer resources than those required by a centralized entity.

The optimization is reached by informing the neighbourhood status; this information is used by reinforcement rules and conjoined with the collective algorithms. This way, the Q-tables are continuously updated to reflect the agent's perception of the world. They improve their decision-making as more raw materials flow through the system. Agents only have to activate the corresponding behaviour according to a schedule that gets adjusted during the system execution.

5. Case study description and performance results

In the previous section, we have shown how the different agents can interact with each other to carry out the control of the production processes. We distribute agents and product objects over the network, and have them interact with each other to simulate the communication and cooperation of the actual controllers distributed in a production plant. In this section, we will present an example to demonstrate the interaction model of the agents and the resulting optimization of the production processes. We present an example of some of the performance analysis that has resulted from the model described in this paper to investigate the impact of the dynamic JR decisions on the overall systems performance.

For the experiment configuration, we used one SA, 3-tier production scheme and a FPSA. Tier A is composed of MA_1 and MA_2 performing operations O1 and O2 taking 18 and 34 time units and 19 and 20 time units at different machines respectively. Tier B is composed of MA_3 performing operations O3 (36) and O5 (19), MA_4 with O4 (30) and MA_5 with O5 (14) and O6 (29). Finally, tier C is composed of MA_6 with O7 (30) and O8 (18) and MA_7 with O8 (19). During the initialization stage, MAs search their local Directory Facilitator. They receive a list of partners from the next tier of the PS. Also, MA asks next-tier neighbours about their capabilities. This information is used to initialize the Q-tables.

There are three different products over which, three operations must be accomplished: P1, operations O1, O4 and O5; P2, operations O2, O4 and O7; P3, operations O2, O5 and O7. The corresponding demand for each of them is: between time units 1 - 50, P1-type raw materials for 30 products must be released from the SA. Between 21 - 60, P2-type raw material for 40 products and between 31 - 70, P3-type raw material for 60 products must be released from the SA. All of them travel through the PS until they reach the FPSA at the other side of the network.

Starting simulation, the SA sends message indicating the MA_1 to add a new product to the buffer's queue. However, this is only a notification; the actual

processing will not take place until the SCHA informs the MA_1 to do so. After the operation is completed, the agent is responsible for routing the product to the next tier where two events are triggered: a neighbour's request to add the intermediate product and a SCHA's request to add a new event for the corresponding machine to check its buffer for pending jobs.

For communication purposes, domain ontology is used. This encodes different types of events that agents must be aware of. The ontology consists of a set of numerical constants for events such as: "raw material release" (RELEASE_MP), "final product report" (INCR_STOCK) and "product leaves machine buffer" (DEQUEUE). Message structure between MA and SA is slightly different from conversations with the SCHA. This latter manages only events notifications from and to the group of agents. MA and SA, on the other hand, implement action requests for operations. As shown in fig. 3, MA_1 requests MA_2 to add a product to MA_2's queue. That is the way products travel from tier to tier. In the last conversation, MA_3 requests MA_4 and MA_5 to inform their capabilities in order to update MA_3's Q-tables. Each MA responds to the query with the list of operations it is able to perform and associated processing time using FIPA-request protocol. These operations are specified as "operations concept" using domain ontology.

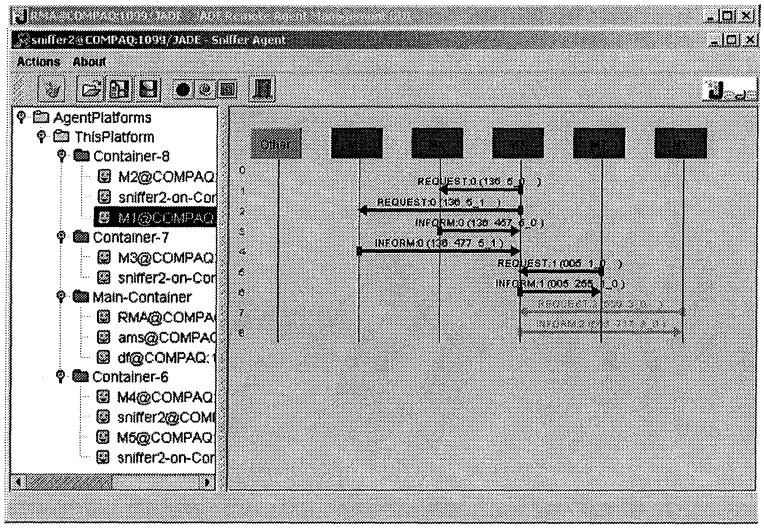

Figure 3 - Message interchange among a set of MAs (Sniffer Agent screenshot).

As shown in fig. 3, MAs are distributed along the JADE containers according to the production tier they belong to. Also, container facilities work as a bridge for experimenting with distributed locations. There is an option to connect physically distant machines that resembles the common layout of a real PS (as shown in Sheremetov et al., 2003). In other words, presented implementation is two-folded. It can be used as a test-bed for trying different configurations, and if required, it is intended to function as an implementation in a real scenario.

Planning and punishment sub algorithms were applied each second of simulation time and ant sub-algorithm was applied each 0.5 sec. The general parameters were: learning rate = 0.8 and exploration rate = 0.08.

At the second stage of the experiments, an adaptation of the Q-routing algorithm (Littman et al., 1993) within the framework of the JR problem was compared with the Q-neural algorithm, described in this paper. The comparison of these two algorithms can be found in Fig. 4. This figure shows the number of products produced (arrived at the FPSA) vs. the average production time.

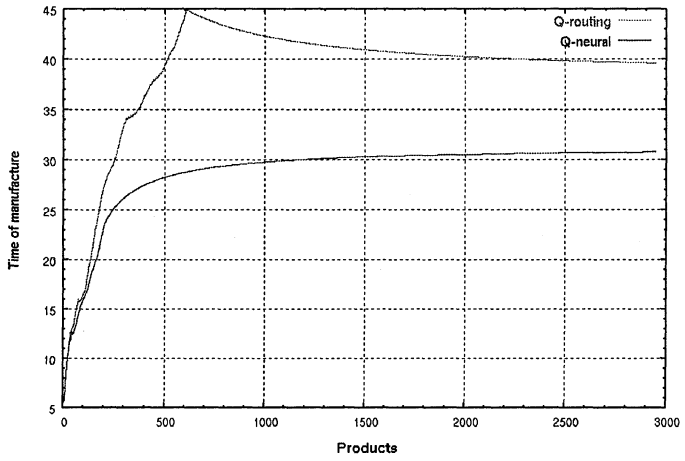

Figure 4 - Comparison of the results of the Q-routing and Q-neural algorithms

Fig. 4 shows the adaptability and better performance of the Q-neural algorithm due to the following. MAs based in Q-routing, make their decisions in a greedy way to optimize their local utility functions. This conduces to a buffer's saturation and a decrement of the global utility function as a result of this greedy behaviour. However, the MAs based in Q-neural, make their decisions taking into account both local utility and the global PS performance. As a result, the performance of the scenario is improved thanks to the adaptation to the changes of the PS environment.

6. DISCUSSION AND CONCLUSIONS

Today's challenge is to optimize the overall business performance of the modern enterprise. In general, the limitations of traditional approaches to solving the JR problem are due to the fact that these models do not correspond to the reality because of incomplete information, complex dynamic interactions between the elements, or the need for centralization of control and information. Most of heuristic techniques on the other hand, do not guarantee the overall system optimization.

In this paper, JR problem is addressed within the framework of NECOIN theory. In order to optimize the global behaviour of PS on the shop-floor level, learning process using RL algorithms to adapt agent's local behaviour is used. This model is implemented in the agent-based parallel modelling and simulation environment over the JADE platform. Being the agglutinating centre of the enterprise information infrastructure, an AP also serves as an experimental test-bed for the implementation of the models developed in this paper. By means of this, we can easily implement the algorithms tested in the simulated environment into the real-world applications.

The experiments on the comparison of this approach with that reported in (Sheremetov et al., 2003) on a common platform is under development. The tested control systems will have varying production volumes (to model the production system with looser/tighter schedules) and disturbance frequencies, so that the impact of the JR and sequencing decisions in various manufacturing environments can be evaluated. The communication protocol's behaviour between agents is under investigation using communication network simulation tools like Network Simulator (NS-2). We also pretend to compare our algorithms with other classic optimization methods using the developed multiagent test-bed.

Though we do not have yet the results of these experiments, we conclude that the JR problem is well situated for the application of the NECOIN theory. In addition, the adaptive Q-neural algorithm provides better throughput and reliability than other algorithms. In future work, an adapted model of the CMAC Neural Network will be used for the Q-values approximation. More complicated punishment algorithm will be developed to adjust the local utility functions.

7. ACKNOWLEDGMENTS

Partial support for this research work has been provided by the IMP, within the project D.00006.

8. REFERENCES

1. Brennan RW, O W. A simulation test-bed to evaluate multi-agent control of manufacturing systems, In Proc. of the 32nd Winter simulation conference, December 10-13, Orlando, Florida., 2000.
2. Denkena B, Tonshoff HK, Zwick M, Woelk PO. Process Planning and Scheduling with Multiagent Systems. In V. Marík, L. Camarinha-Matos, H. Afsarmanesh (Eds.) Knowledge and Technology in Product and Services. Selected papers of the IFIP, IEEE International Conference BASYS'02, Kluwer Academic Publishers, 2002; 339-348.
3. Heragu SS, Graves RJ, Byung-In K, St-Onge, A. Intelligent agent based framework for manufacturing systems control. IEEE Transactions on Systems, Man and Cybernetics, 2002; 32(5): 560–573.
4. Julka N, Srinivasan R, Karimi I. Agent-based supply chain management-1: framework. Computers and Chemical Engineering, 2002: 26.
5. Littman M, Boyan J. A Distributed Reinforcement Learning Scheme for Network Routing. School of Computer Science, Carnegie Mellon University, 1993.
6. Rocha-Mier, Luis. Apprentissage dans une Intelligence Collective Neuronale: application au routage de paquets sur Internet. PhD thesis, Institut National Polytechnique de Grenoble, 2002.
7. Rocha-Mier L, Sheremetov L, Contreras M, Osuna C, Romero M, Villa L, Hernandez A L. Global Supply Chain Management based on Collective Intelligence. In Proc. of the 2nd World POM Conference and 15th annual POM Conference. Cancun, Mexico, April 30 - May 3, 2004.
8. Sheremetov LB, Martínez J, Guerra J. Agent Architecture for Dynamic Job Routing in Holonic Environment Based on the Theory of Constraints. Honolic and Multi-Agent Systems for Manufacturing (HoloMas 2003), V. Marik, D. McFarlane, P. Valckenaers, eds.: Springer Verlag, 2003; LNAI 2744: 124-133.
9. Shimbo M., Ishida, T. Controlling the learning process of real-time heuristic search. Artificial Intelligence, 2003; 146: 1–41.
10. Sutton R, Barto A. Reinforcement Learning: An Introduction. The MIT Press, 1998.
11. Usher, John M. Negotiation-based in job shops via collaborative agents, Third International ICSC Congress on World Manufacturing WMC'2001, Rochester, N.Y., Sept. 24-27, 2001.
12. Wolpert D, Kagan T. An Introduction to Collective Intelligence. Technical Report NASA-ARCIC-99-63, NASA Ames Research Center, 1999.