Dynamic scheduling in Petroleum process using reinforcement learning

Nassima Aissani[1], Bouziane Bedjilali[1],

[1]Oran University, BP 1524 El M'nouer, Oran, Algeria
aissani.nassima@yahoo.com,bouzianebeldjilali@yahoo.fr

Abstract. Petroleum industry production systems are highly automatized. In this industry, all functions (e.g., planning, scheduling and maintenance) are automated and in order to remain competitive researchers attempt to design an adaptive control system which optimizes the process, but also able to adapt to rapidly evolving demands at a fixed cost. In this paper, we present a multi-agent approach for the dynamic task scheduling in petroleum industry production system. Agents simultaneously insure effective production scheduling and the continuous improvement of the solution quality by means of reinforcement learning, using the SARSA algorithm. Reinforcement learning allows the agents to adapt, learning the best behaviors for their various roles without reducing the performance or reactivity. To demonstrate the innovation of our approach, we include a computer simulation of our model and the results of experimentation applying our model to an Algerian petroleum refinery.

Keywords: reactive scheduling, reinforcement learning, petroleum process, multi-agent system.

1 Introduction

Current oil and gas market trends, characterized by great competitiveness and increasingly complex contradictory constraints, have pushed researchers to design an adaptive control system that is not only able to react effectively, but is also able to adapt to rapidly evolving demands at a fixed cost. The system does this by using the available resources as efficiently as possible to optimize this adaptation. [4] presented an analysis of the needs of production systems, highlighting the advantages of adopting a self-organized heterarchical control system. The term, heterarchy, is used to describe a relationship between entities on the same hierarchical level [6]. Initially proposed in the field of medical biology, it was then adapted for several other domains [9; 10; 7]. In the multi-agent domain, the term, heterarchy, is relatively close to the concept of "distribution", as used in "distributed systems". However, from our point of view, the fact that the decisional capacities are distributed does not mean that the multi-agent system is organized heterarchically, even though this is often the case [15;17]. Nonetheless, the heterarchic organization of distributed systems is the assumption that we make in this paper. From our point of view, this assumption is justified by the system dynamics and the volatility of the information, which make a

purely or partially hierarchical approach inappropriate for creating an effective reactive system [4].

In this paper, we focus on the dynamic control of complex manufacturing systems, such as those found in the petroleum industry. In this industry, all functions (e.g., planning, scheduling and maintenance) and resources (e.g., turbines, storage systems) are automated.

2 BRIEF DESCRIPTION OF UNIT3100 IN RA1Z REFINERY

This unit is designed to produce oil from the base oil treated in the units HB3 and HB4 and imported additives, the base oil is received in Tank TK2501 to TK2506. Each docking Tank stock defined grade of oil (SPO, SAE10-30, BS) (Production of 132,000 t / year for an amount of 10% additives) if the type of oil stored in a tank must be changed, the tank must first be rinsed for hours which is often avoided. This unit produces two major oil: engine oils 81% of the production (gasoline, diesel, transmission oils) and industrial oils (hydraulic (TISK), turbines (torba), spiral (Fodda), compressor (Torrada) and various oils). To do this, two methods are used: continuous mixing (mixing line) and mixing in discontinuous (batch) (see Figure 1). In this article we focus on the mixing line. To produce finished oil, a recipe must be applied:

```
X1% Hb1 + X2% Hb2+ X3% Additif1
```

Where : Xi is the rate and HBi is the base oil.

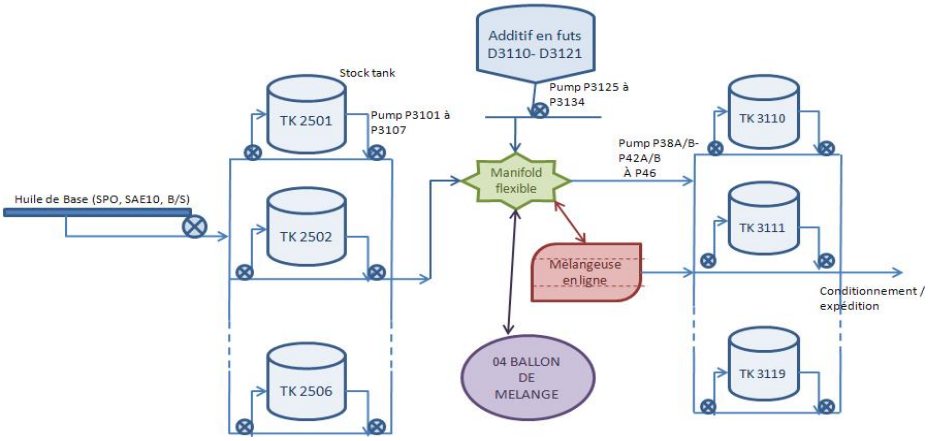

Fig. 1. Unite 3100 model

The mixing line its base oil from the docking Tanks, which produce this decade plan (see figure 2):

In this paper, we aim to develop an adaptive control system for Unit3100 which will produce dynamically efficient scheduling solution using resources in optimal way.

We consider each resource and Oil in tank as a decisional entity, and we model them as agents.

Finished oil		Base oil needed				
Grades	quantity to produce	SPO	SAE10	SAE30	BS	Total
NAFTILIA 20W50	1000		126,3	776,2		902,5
CHELIA 10W	1000		956			956
NAFTILIA 40	800			616,8	149,6	766,4
CHIFFA 40	2000			1520	424	1944
CHELIA 40	1000			766,6	200	966,6
CHELIA TD 20W40	2000					0
TISKA 32	1000		992			992
TISKA 68	2000					0
CHELIA VPS 20W40	2000		475	1057	300	1832
TASSILIA EP 90	1200		240	120	780	1140
TASSADIT A2	1200					0
Total	15200	0	2789,3	4856,6	1853,6	9499,5

Fig. 3. Production plan

3 STATE-OF-THE-ART

We conducted a state-of-the-art review of the dynamic scheduling problem in the literature. This section highlights the studies that reflected our point of view.

3.1 Dynamic scheduling

In manufacturing control, scheduling is the most important function. In this paper, we focus on dynamic scheduling.

[5] Have classified dynamic scheduling into three categories: predictive, proactive, and reactive. The first, predictive, assumes a deterministic environment. Predictive solutions call for a priori off-line resource allocation. However, when the environment is uncertain, some data (e.g., the actual durations) only becomes available when the solution is being executed. This kind of situation requires either a proactive or reactive solution. Proactive solutions are certainly able to take environmental uncertainties into account. They allocate the operations to resources and define the order of the operations, though, because the durations are uncertain, without precise starting times. However, such solutions can only be applied when the durations of the operations are stochastic and the states of the resources are known perfectly (e.g. stochastic job-shop scheduling) [3]. The third type of dynamic scheduling, reactive, is also able to deal with environmental uncertainties, but is better suited for evolving processes.

Reactive solutions call for on-line scheduling of resources. In fact, the resource allocation process evolves, making more information available and thus allowing decisions to be made in real-time [16; 11; 5; 1]. Naturally, a reactive solution is not a simple objective function, but instead a resource allocation policy (i.e., a state-action mapping) which controls the process. In this paper, we focus exclusively on reactive solutions.

3.2 Reinforcement learning

Over the last few decades, scheduling researchers were inspired by artificial intelligence whose methods were based exclusively on operational research algorithms of exponential complexity. Taking into account performance effectiveness and efficiency, which means optimizing several criteria, will increase problem complexity even more. Artificial intelligence has allowed such complex problems to be solved, yielding satisfactory, if not always optimal, solutions.

[9] used genetic algorithms (GA) to adapt the decision strategies of autonomous controllers. Their control agents use pre-assigned decision rules for a limited amount of time only, and obey a rule re-placement policy that propagates the most successful rules to the subsequent populations of concurrently operating agents. However, GA do not provide satisfactory solutions for reactive scheduling. Therefore, a reactive technique must be integrated into GA to allow the system to be controlled in real time.

Reinforcement learning (RL) might be an appropriate way to obtain quasi-real-time solutions that can be improved over time. Reinforcement learning is learning by trial and error dedicated to agents learning. In this paradigm, agents can perceive their individual states and perform actions for which numerical rewards are given. The goal of the agents is thus to maximize the total reward they receive over time.

[8] used reinforcement learning to optimize resource use in a very expensive electric motor production system. Such systems are characterized by a variety of products that are produced on re-quest, which requires a great deal of flexibility and adaptability. The assembly units must be autonomous and modular, which makes performance control and development difficult. [8] considered these units as insect colonies able to organize themselves to carry out a task. Self-organization can reduce the number of resources used, allowing production risk problems to be solved more easily.

The most used reinforcement learning algorithm is Q-learning. [18] extended this algorithm by using a reward function based on EMLT (Estimated Mean LaTeness) scheduling criteria, which are effective though not efficient. [2] pro-posed an intelligent agent-based scheduling system. They employed the Q-III algorithm to dynamically select dispatching rules. Their state determination criteria were the queue's mean slack time and the machine's buffer size. These authors take advantage of domain knowledge and experience in the learning process.

But in this paper, we are exploring a more developed algorithm "SARSA algorithm" in a heterarchical organisation of agents. In conclusion, we are trying to experiment reinforcement learning by using **SARSA** algorithm to conceive an ***adaptative*** and ***reactive*** manufacturing control system for petroleum process based on **heterarchical** multi-agent architecture. In the next section, we will present our system architecture and motivating our choices.

4 THE PROPOSED CONTROL SYSTEM

A multi-agent system is a distributed system with localized decision-making and interaction among agents. An agent is an autonomous entity with its own value system and the means to communicate with other such entities. For a general survey of the application of multi-agent systems in manufacturing, see the review by [1]. In order to develop multi-agent system with a reactive decision capability in an uncertain environment, they may be modelled as Markov Decision Process (MDP) [12]. And to improve the system performances and learn optimal policy in Markov environment, If the transition function T (modelling the system's evolution from state to state) is unknown while an objective can be identified a learn-by-trial process such as RL [12;13] can be designed.

4.1 The proposed manufacturing control system

We consider that a petroleum refinery exists in a dynamic, uncertain and unpredictable environment, since it is subject to internal stress (e.g., production risks) and external constraints (e.g., forced markets, unexpected orders). According to[12], the decisions made in such environments involve Markov decision processes (MDP). Clearly, in such a Markovian context, it is necessary to consider the transition function T, modelling the system's evolution from state to state as an unknown. According to [12] and [13], a learn-by-trial process, such as reinforcement learning, should be used determine the optimal policy. This modelling approach is widespread. Figure 1 shows the main functions embedded in each agent.

4.2 SARSA (Stat, Action, Reward, new Stat, new Action) algorithm to resolve dynamic scheduling problem

An MDP is a tuple < S,A,T,R >, where *S* is a set of problem states, *A* is a set of actions, $T(s, a, s') \rightarrow [0, 1]$ is a function defining the probability that taking action *a* in state *s* results in a transition to state *s'*, and $R(s, a, s') \rightarrow R$ defines the reward received after such a transition.

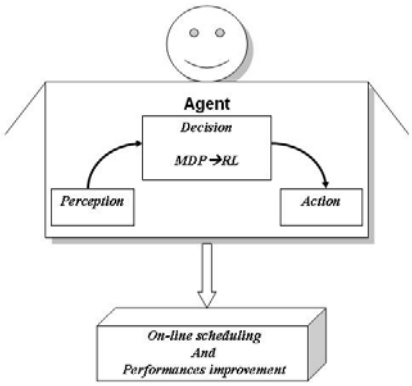

Fig.1. MDP→ RL →improvement of on-line scheduling Performances

If all the parameters of the MDP are known, an optimal policy can be found by dynamic programming. If *T* and *R* are initially unknown (which is commonly the case when considering industrial case studies), *Reinforcement learning* (RL) methods can learn an optimal policy by direct interaction with the environment. RL is learning to act by trial and error. Agents perceive their individual states and perform actions for which numerical rewards are given. The goal of the agents is thus to maximize the total reward received over time. This technique is often used in robotics, in order to teach a robot the behavior to achieve its goals and to overcome obstacles.

The ***SARSA*** algorithm is used to learn the function $Q^{\pi}(s, a)$, defined as the expected total discounted return when starting in state ***s***, executing action ***a*** and thereafter using the policy $\boldsymbol{\pi}$ to choose actions:

$$Q^{\pi}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma Q^{\pi}(s', \pi(s'))] \quad (1)$$

The discount factor $\gamma \in [0,1]$ determines the relative importance of short term and long term rewards. For each *s* and *a* we store a floating point number *Q(s,a)* for the current estimate of $Q^{\pi}(s,a)$.

As experience tuples < *s,a,r,s',a'* > are generated through interaction with the environment, the table of Q-values is updated using the following rule:

$$Q(s,a) = (\alpha - 1)Q(s,a) + \alpha(r + \gamma Q(s', a')) \quad (2)$$

The learning rate $\alpha \in [0,1]$ determines how much the existing estimate of $Q^{\pi}(s,a)$ contributes to the new estimate.

If the agent's policy tends towards greedy choices as time passes, the ***Q(s,a)*** values will eventually converge to the optimal value function ***Q*(s,a)***. To achieve this, we use a Boltzman probability which determines the probability of choosing a random action.

Figure 2 shows the steps of the SARSA algorithm

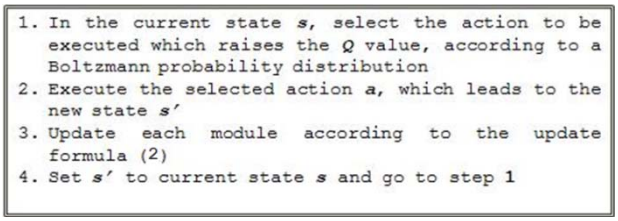

Fig. 2. The SARSA algorithm

In our case, this algorithm will make the Resource Agent learn its action policy π, which in turn makes it able to choose the best action for each state (accept task/request, or not). This algorithm works with the following data:

State parameters are the current time $\mathbf{t} \in \mathbf{0\ldots T}$; the inventory of pmps $\mathbf{p_1\ldots p_n}$ and their states $\mathbf{Sp_1\ldots Sp_n}$ (e.g., maximum capacity, feeding, receiving); the list of Storage Tanks $\mathbf{T_1\ldots T_m}$, and their states $\mathbf{ST_1\ldots ST_m}$ (e.g., Capacity). **Action** concerns the reception or not of the product, stop or start pumping.... **Reward** function assigns no reward to most of the states and positive rewards to a specific goal state. For more precision and to obtain a proper convergence, the reward function is a state combination engendered by an action. One idea was to take into account the volum in tanks and ($\mathbf{C_i}$) and feeding and uploading stream ($\mathbf{Fd_i}$) ($\mathbf{Ud_i}$) in the reward function:

$$R_{Part-Ag} = \begin{cases} 1 \text{ if } C_i(t) = C\max_i \\ 0 \text{ if } C_i(t) \geq C\min_i \\ -1 \text{ if } C_i(t) < C\min_i \end{cases} \qquad R_{Resource-Ag} = \begin{cases} 1 \text{ if } \sum_{i=1}^{6} Fd_i = 1500 \text{ m}^3/h \\ -1 \text{ if } \sum_{i=1}^{6} Fd_i = 0 \end{cases}$$

4.3 Multi-agent interaction

As shown in Figure 3, the MCSR (Manufacturing Control System using Reinforcement learning) architecture consists of “resource agents” for the pumps, “parts agents” for the tanks containing oil and an "observer agent" to control the process.

Based on Alaadin modelling (Ferber and Gutknecht, 1998), the resource and parts agents have certain properties, roles and groups. Initially, each agent must have knowledge about its properties (e.g., tank number, capacity, characteristics… or pump reference, flow stream…), its role (i.e., storage or pumping) and its group (e.g., tanks containing the same product). The observer agent has a global view of the system, and the state variables that it observes are the indicators of performances.

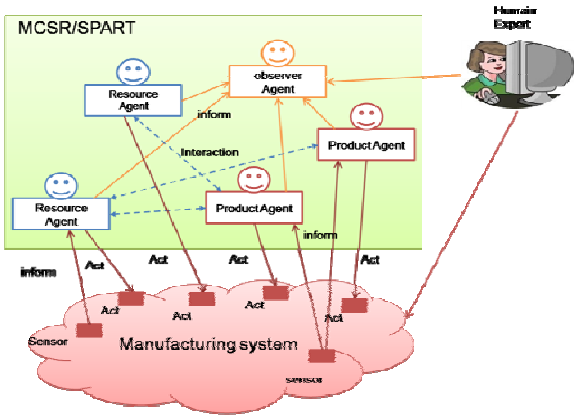

Fig. 3. MCSR architecture (Manufacturing Control System using Reinforcement learning)

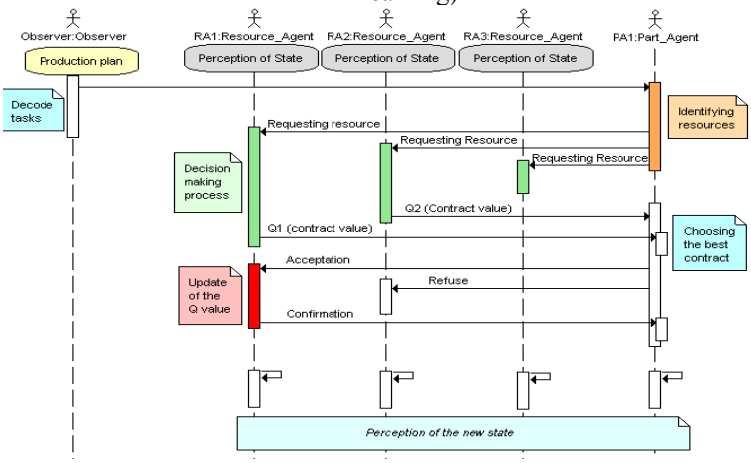

Fig. 4. MAS interaction (Sequence Diagram)

The Observer Agent receives the decade production demand. It sends the relevant set of tasks to each agent. Each part agent (i.e., tanks containing oil) and resource agent (i.e., pump) perceives its state which is a combination of its individual state (e.g., stopped, busy) and the set of tasks that they must execute.

To deal with agent interaction, we used the well-known Contract Net protocol [14] to determine the task allocation to resource agents.

The idea is roughly the following: a part agent has a task request that it proposes to resource agents, and then the resource agents give their propositions. The part agent chooses the best proposition and establishes the contract. A detailed illustration of the agent interaction is provided in figure 4.

5 IMPLEMENTATION AND EXPERIMENTS

Our model was simulated in the Borland Jbuilder environment because of its potential for facilitating communication and thread programming and because of its compatibility with the chosen MADKIT platform architecture for SMA development (visit http:// www.madkit.org/downloads). One of the advantages of the reinforcement learning algorithms is that they allow evaluation during learning. To permit this evaluation, we selected the following criteria.

5.1 Description of the process & constraints

A petroleum refinery is subjected to many operational constraints. Operational constraints include the requirement that only one tank at a time can receive oil, but several can simultaneously feed mixing line, and another that states a tank cannot receive and send oil at the same time. Problem inputs include the base oil arrival schedule, which describes the volumes and qualities of the base oils and additives that will be received in the refinery during the desired time horizon; the finished oil demands, and the current levels and qualities of the base oil in the storage tanks. The major constraints considered can be formalized as follows (see parameter definitions given in 4.2):

C1: Tank storage level can never be less than a given threshold $C_i(t) \geq C\min_i$

C2: Tank storage level can never be greater than a given threshold . $C_i(t) \leq C\max_i$

C3: mixing line must always contain oil $\sum_{i=1}^{n} Fd_i(t) > 0$

C4: Tank cannot feed and receive at the same time $\begin{cases} Ud_i(t) > 0, Fd_i(t) = 0 \\ Ud_i(t) = 0, Fd_i(t) \geq 0 \end{cases}$

The base oil is stored in specific storage tanks (TK2501-TK2506 (see figure 5)). The total time horizon spans 160 hours, during which completely defined oil parcels have to be received from the pipeline. Six oil tanks are available; all of them have the same capacity, but different amounts of oil at the beginning of the time horizon (figure. 6.)

Tank	Pump	Flow stream M3/h	Base oil	TONNAGE	initial tonnage
2501	P3101	40	SPO	2343	312
2502	P3102	40	SAE10	2371	2182
2503	P3104	40	SAE30	2399	625
2506	P3105	40	SAE30	1924	1553
2504	P3106A	40	B/S	1968	1789
2505	P3106B	40	B/S	1968	1334

Fig. 5. Tank setting

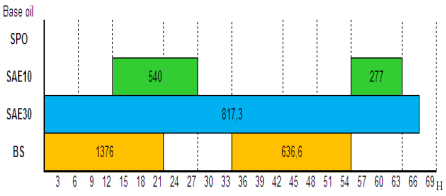

Fig. 6. base oil arrival

Aims are to receive all base oil using available pumps feeding Tank with sufficient capacity, and to produce exactly the requested quantity with the available quantity of bases oil in the range of the decade. For this reason, we consider as an evaluation criterion the *Cmax* (Maximum duration time to produce the requested products).

5.2 Experimental results

The experiment was conducted as follows: we launch the system with data explained above. The graph (Figure 7) shows the results for the first phase of the learning algorithm. As this graph shows, before 5000 iterations, the *Cmax* variation is rather high. It varied in the interval [100h, 1500h], which is a modest result. This can be justified by the fact that the results are from the *exploration* phase, in which actions are executed randomly according to the Boltzmann probability [1]. The second phase is the *exploitation* phase, in which the choice of actions is based on Q values (just before and after 5000 iterations), and the results are better. This phase produced solutions with a very interesting *Cmax* of 45 h. Thus, we can state that our system converges towards optimal solutions by minimizing the total time of production even with maintenance tasks.

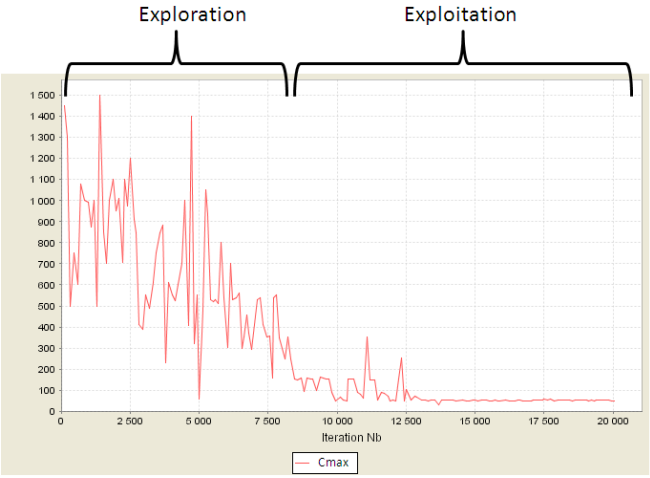

Fig. 7. Cmax graph

5.3 Reactive behavior

Despite being relatively under control, thanks to the preventive maintenance plans, perturbations are always possible in a refinery. To test our system faced with such random events, we caused system perturbations in order to observe the system's behavior.
We caused the same perturbation (a breakdown of *P3102*) in the *exploration* phase at the *2000th* iteration and again in the *exploitation* phase at the *15000th* iteration. When such perturbations occur in the current system, some production tasks have to be cancelled to allow the maintenance tasks to be performed. The human expert then has to manually find a solution to replace the cancelled production tasks. However, in our experiment, the disturbance in the exploitation phase was quickly compensated for without any *Cmax* variation over *49h*, and the system was brought back to the level of its best performances. These results show that our system is able to learn how to establish a continuously improving optimal control policy to schedule maintenance tasks within a production plan without reducing the production rate.

6 CONCLUSION AND FUTURE WORKS

In this paper, we have presented a multi-agent model for the dynamic scheduling of in petroleum process. In this model, agents simultaneously insure effective scheduling and continuous improvement of the solution quality by means of reinforcement learning, using the SARSA algorithm. We have also provided an overview of the research done in the field of manufacturing control, focusing on dynamic and reactive scheduling. The results of our experiments with this model show that our approach can generate on-line scheduling solutions and improve their quality by minimizing *Cmax*. Nevertheless, we want to widen the time horizon of our experimentation, taking into consideration more complex production units. Last, we are going to work on a holonic version of our model for future comparison with the multi-agent model.

References

[1] Aissani. N, D. Trentesaux and B. Beldjilali, 2008, Use of Machine Learning for Continuous improvement of the Real Time Manufacturing control system performances. *IJISE: International Journal of Industrial System Engineering,* Vol 3, No 4, p 474-497
[2] Aydin. M. E, Öztemel. E, (2000), Dynamic job-shop scheduling using reinforcement learning agents, *Robotics and Autonomous Systems*, Vol 33, No 2, p 169-178
[3] Bidot J, T. Vidal, P. Laborie and J. C. Beck, 2007, A General Framework for Scheduling in a Stochastic Environment. *Proc International Joint Conference on Artificial Intelligence* IJICAI07, P. 56-61

[4] Bousbia, S and D. Trentesaux, (2002), Self-Organization in Distributed Manufacturing Control: state-of-the-art and future trends*, IEEE International conference on Systems, Man & Cybernetics*, Hammamet, Tunisia, Vol 5, 6 p.
[5] Csaji B. C and Monostori L.. 2006. Adaptive algorithms in distributed resource allocation. *Proc of the 6th International Workshop on Emergent Synthesis*, August 18–19, The University of Tokyo, Japan, p. 69-75
[6] Duffie, N.A., Prabhu, V.V. (1996) 'Heterarchical control of highly distributed manufacturing Systems', *International Journal of Computer Integrated Manufacturing*, Vol. 9, No. 4, 1996, p. 270-281.
[7] Haruno. M, Kawato. M (2006),'Heterarchical reinforcement-learning model for integration of multiple cortico-striatal loops: fMRI examination in stimulus-action-reward association learning', *Neural Networks,* Vol 19, (2006), p 1242–1254
[8] Katalinic. B and Kordic. V (2004) 'Bionic assembly system: concept, structure and function' *Proc of the 5th IDMME 2004*, Bath, UK, April 5-7, 2004
[9] Maione. G and Naso. D, (2003), 'Discrete-event modeling of heterarchical manufacturing control systems', *Systems, Man and Cybernetics, 2004 IEEE International Conference*, Vol 2, 10-13 Oct. 2004, p 1783 - 1788
[10] Prabhu. V.V, (2003). "Stability and Fault Adaptation in Distributed Control of Heterarchical Manufacturing Job Shops," *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 1, p. 142-147.
[11] Pujo. P and Brun-Picard. D, 2002, Pilotage sans plan prévisionnel ni ordonnancement préalable , Méthodes du pilotage des systèmes de production, *Hèrmes*, 2002. p 129- 162.
[12] Russell S. Norvig P. (1995) 'Artificial Intelligence: A Modern Approach', *The Intelligent Agent Book. Prentice Hall Series in Artificial Intelligence*.
[13] Singh. S and Sutton R., (1996), Reinforcement learning with replacing eligibility traces. *Machine Learning*, Vol 22, p1-3
[14] Smith. R. G., (1980), The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *IEEE Transactions On Computers*, Vol. C-29, No. 12, p 1104-1113
[15] Trentesaux D., Dindeleux R. and Tahon C. (1998), A MultiCriteria Decision Support System for Dynamic task Allocation in a Distributed Production Activity Control Structure, *Int. Journal of Computer Integrated Manufacturing*, Vol. 11 n°1, 1998, p. 3-17.
[16] Trentesaux D., Gzara M., Hammadi S., Tahon C. and Borne P. (2001). D-Sign: un cadre méthodologique pour l'ordonnancement décentralisé et réactif. *Journal Européen des Systèmes Automatisés*. p. 933-962
[17] Trentesaux D., Les systèmes de pilotage hétérarchiques : innovations réelles ou modèles stériles ?, *Journal Européen des Systèmes Automatisés*, vol. 41, n°9-10, 2007, pp. 1165-1202.
[18] Wei Y-Z and Zhao M-Y, (2005), A reinforcement learning-based approach to dynamic Job-shop scheduling, *Acta automarica sinica*, Vol 31, No 5, p 765-771