Australian Journal of Computer and Communications
ISSN: 2203-9492 (Print) ISSN: 2203-9500 (Online)

MAS Approach Based on a Temperature-Equilibrium Model for Simulating Scheduling Requirements in Manufacturing Processes

Antonio Gordillo-Sosa, Miguel Barrón, Joel Quintanilla
Tecnologías de la Información y Comunicación
Universidad Tecnológica del Suroeste de Guanajuato.
Carr. Valle-Huanímaro km1.2, 38400 Valle de Santiago, Gto. México.

Adriana Giret
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia.
Camino de Vera s/n 46022 Valencia,España.

Rafael Guzmán-Cabrera
Departamento de Ingeniería Eléctrica.
Universidad de Guanajuato. México,
Carr. Valle-Salamanca comunidad Palo Blanco.
e-mail: guzmanc@ugto.mx

Abstract

In many real world manufacturing environments the work balance problem needs to be solved in order to obtain an optimal solution for resource allocation. The main objective to be accomplished consists on the adequate distribution of every incoming order to every machine with processing capability. Once an efficient task distribution process is achieved, the entire plant workload can be incremented. This paper focuses on presenting an approach of a multi-agent system (MAS) based algorithm for automatically processing task allocation, particularly in manufacturing applications. The developed simulation environment integrates many of the key critical working conditions of a classical job-shop scheduling problem, such as: dynamic environment, order rescheduling, and priority change. The results section presents the performance analysis for the algorithm, showing that the proposed specific developed MAS environment approach is a viable solution for properly manage many real manufacturing working instances with good quality.

Keywords: Multi-agent Systems, Intelligent Manufacturing Systems, Task Allocation

1. Introduction

The manufacturer's success is no more measured by their ability to cost-effectively produce a single product; success now seems to be measured in terms of flexibility, agility and versatility [1]. In order to survive, manufacturing systems need to adapt themselves at an ever-increasing pace to incorporate new technology, new products, new organizational structures, etc. The above trends have motivated researchers in academia and industry to create and exploit new production paradigms on the basis of autonomy and cooperation because both concepts are necessary to create flexible behavior and thus to adapt to the changing production conditions. Such technologies provide a natural way to overcome such problems, and to design and implement distributed intelligent manufacturing environments [1]. One of the more complex problems in manufacturing systems is scheduling, an essential activity focused on distribute orders to machines ensuring an efficient production scheme. Many efforts have been developed aiming to address the tasks orders distribution: event though it cannot deal with all the real world problems, the Resource Constrained Project Scheduling Problem (RCPSP) is a standard model for NP-hard problems belonging to project scheduling [2], this same model have been extended using multi-agent systems (MAS) [3]

Koestler [4] set the basic concepts to establish a holonic organization model to be applied particulary in manufacturing. This kind of system is basically cooperative and distributed. In fact, a holonic organization is aimed to feature the robustness of descentralized organizations, and the efficiency of hierarchical control architecures [5]. From the manufacturing control viewpoint, an holonic system is focused mainly on scheduling, shop floor control, reactive scheduling and multi-agent negotiation. The

objective of this organizational framework is to assign tasks to the resources for optimizing certain performance criteria like lead time, or tardiness. For example, from a feature based milling process [5] perspective, the product holon specifies the feature to be processed: pocket shape, size and tolerances. The resource holon executes this task by selecting appropriate tools. The resource holon carries out a tool length measurement. From the manufacturing viewpoint, several order holons are launching tasks on resources. Holonic manufacturing systems are tightly related to the domain of multi-agent systems and, more precisely, of autonomous cooperative agents for distributed manufacturing. Modern manufacturing systems have to deal with dynamic situations such as machine break-down, emergency orders, priority changes and other kinds of disturbances. The holonic concept has been proposed as an efficient paradigm for developing such an adaptive manufacturing system.

Nevertheless, despite the large list of works reported in this field there is almost no resource or scientific study on the performance measure of this type of approaches under very common and critical execution situations. Even so, Bench4Start (http://www.univ-valenciennes.fr/bench4star/) must be pointed out as an interesting approach for defining benchmarking solutions for manufacturing systems. Nevertheless, Bench4Start does not take into account multi-agent based approaches for manufacturing scheduling.Our goal is to simulate a real manufacturing system and design and analyze the performance of an agent based system under key critical situations such as: dynamic environment, tasks rescheduling, and priority change.

1.1 Task allocation in intelligent manufacturing systems: state of the art

Scheduling is one of the most important fields for manufacturing optimization: involves determining the allocation of plant resources. Tasks must be assigned to the process units, and the duration and amount of processed material related to those assigned tasks must be determined [6].

A well-known manufacturing-scheduling problem arising in this area is the job shop scheduling problem with n jobs, each one consisting of m operations. Here, we have a certain number of machines (m) which can be equipped with different tools each of which can handle exactly one of the operations of each job. Each operation needs a certain amount of processing time, the aim is to find an order x of the operations on each machine such that the finishing time c(x) of the last job (the overall processing time for all jobs) is minimized. This is also known to be a very difficult problem such that some test problems of moderate size are still unsolved.

Many important practical problems require efficient allocation of resources in order to complete goal activities over time in presence of complex state-dependent constraints. For example, consider the problem of managing the supply chain of a manufacturing enterprise, involving a large and dynamic amount of job allocation tasks in the assembly lines. For the enterprise to operate efficiently, supply chain functions must be executed in a coordinated manner. On this scenario, the coordination problems can be addressed using a network of cooperating intelligent agents [7,8], each of them performing one or more supply chain functions, and coordinating their actions with other agents. This kind of network is capable of, among other functions: (a) develop efficient communication and coordination scenarios to properly manage change and solve problems (b) create intelligent information infrastructures that keep agents permanently aware of relevant information and (c) develop a set of decision structures in order to properly distribute the incoming job tasks.

By definition, rescheduling is the process of updating an existing production schedule in response to unconsidered disturbances. In dynamic manufacturing environments, the schedules must not only be generated with enough quality, they also have to allow a quick reaction to unexpected events during the process in a resource-effective manner. For these disturbances, rescheduling has to be considered as a practically mandatory design factor. There are many types of these kind of events that can alter all the planned scheduling process, including rush orders, machine failures and time delays, among others.

1.2 Agent technology

With the possibility of creating autonomous entities, capable of communicate their preferences and negotiate among them for achieving individual or system goals, [26] the software agents represent a robust system able to dynamically react to unexpected events. They have been vastly and successfully applied into the manufacturing field [27] and recently, tested and fully operational in industrial environments [28].

1.3 Temperature-equilibrium based mas approach for dynamic scheduling in manufacturing

The proposed solution algorithm, dubbed Temperature Equilibrium Algorithm, is implemented designing a Multi-Agent System (MAS) under the JADE platform, and applies some of the intrinsic capabilities of the software agents, such as cooperation and communication. The simulation with agent-based systems integrated to complex systems has already been verified [9,10], as well as their applicability on manufacturing [11]. In short, the general characteristics that integrate the simulator are:

The job orders amount and their corresponding reference values are generated randomly. Being the latter the priority level associated with every task. Each temperature value, representing the workload associated to every agent in the system is generated randomly. A high value represents a very active ("hot") agent. In addition, a low temperature value indicates a "cold" agent, i.e., an agent with less workload level, and under such conditions , more capable for receiving incoming jobs.

Once generated and assigned the values described above, the internal procedure inside the system for reaching general equilibrium is as follows: the "hottest" agent shares a certain volume of its workload with a "colder" agent, which helps towards a better workload distribution inside the overall system. The values generated previously to the distribution process and the subsequent temperature values, as well as the negotiation status are shown to all agents in order to properly visualize their internal state and to have the ability to interchange workloads whenever required. The next figure depicts this scenario.

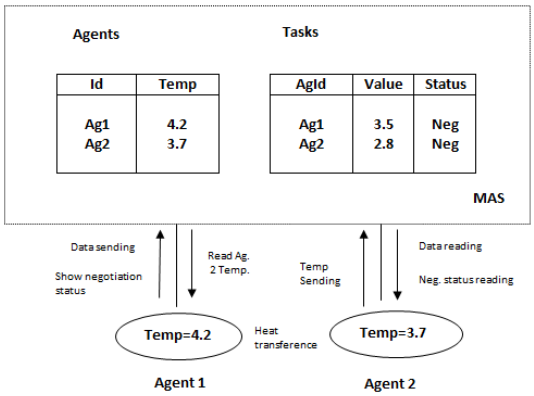


Fig. 1 MAS environment under Temperature Equilibrium

2. Materials and methods

The proposed multiagent architecture for scheduling requirements includes two levels. The first one is formed by those entities related to the real environment yet to be simulated. The second comprehends the responsible agents designed for controlling and handling the aforementioned entities. Furthermore, the multiagent architecture requires more elements, such as information and communication management for each agent, as well as coordination and execution control for the assigned jobs.

Similarly, in the development of the simulation environment, necessary assumptions were taken into account, in order to comply with a proper Computer Integrated Manufacturing System [11]:

- The system can collect and process the necessary information to make decisions at production time.
- The possibility of detecting and reacting to critical situations and react to them is intrinsic to the simulator.
- The real working environment consists in an assembly line system, with three job stations receiving constant manufacturing assignments from different categories. The integrated agents are autonomous, each one representing a physical entity into the system.

2.1 Jade platform

The software framework on which the simulated environment is depending on is JADE (Java Agent

Development Framework). In the sector of software programming science, this platform is the most popular. It has been used in many application domains such as supply chain management, auctions, etc. [18-20] This platform facilitates the development of intelligent agent systems and simplifies the development process under a standard compliance comprising a set of services and autonomous agents. [12], incorporating, as well, the main features of agency: (a) autonomy, (b) social ability (c) reactiveness and (d) pro-activeness [13]. Also, JADE integrates a huge amount of ready-to-use infrastructure components, providing a reliable mechanism for creating customized interaction protocols and ontologies. From a software engineering perspective, JADE incorporates some valuable features [21], such as: a) Interoperability: JADE agents can interoperate with other agents, given that all of them comply with the FIPA standard [22]. b) Uniformity: JADE provides a homogeneous set of APIs that are independent from the network and Java version c) Easy to use: consists on a simple and intuitive set of APIs.

2.2 Dynamic environment

Nowadays, it is impossible to manage and control the production levels without the incorporation of intelligent tools capable of decision making at each stage of the process. One of the basic properties characterizing a modern manufacturing system is dynamism, defined as the set of changes in the structure and behavior during operation. This concept expresses different competencies, responsibilities and relationships between entities [14]. The first scheduling requirement in the manufacturing area is for it to be dynamic [15]; in it, the job orders arrive to the assembly line with diverse characteristics like different priorities, manufacturing intervals, etc. The system must be able of collecting and processing information from different sources in order to properly make decisions at all stages during the process. In this kind of environment, the work doesn´t stop, it is only prioritized in different forms. Applying a proper algorithm for distributed control, the individual agents can make their own decisions over manufacturing control relating to resource allocation, prioritization,etc. The main benefit of this approach is that it improves the robustness to change inside the system, managing the sudden disruptions or reorganizations in a better way [23]. In order to recreate a simulation environment for this specific requirement, the software elements depicted on Figure 2 were included.

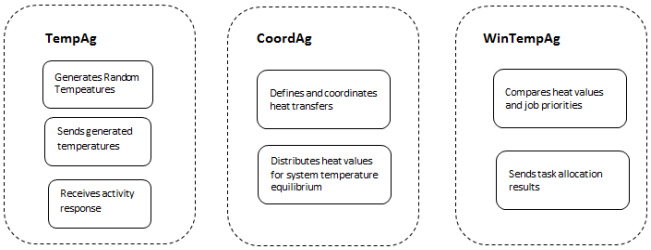


Fig. 2 Dynamic Environment Classes using Temperature Equilibrium Algorithm

The next requirements were considered for the integration of the simulated Dynamic working environment:

- Every agent can process only one job at a time.
- The individual priority and temperature values corresponding to the set of incoming tasks are generated randomly, in the form of double precision numbers rounded up to one decimal place. This consideration, as detailed later, contributes to a more uniform application of the protocol equilibrium into the agents integrating the MAS.
- Once initiated the equilibrium protocol into the system, the task order is allocated to the most capable agent.

The Table 1 describes the operational flow into the MAS environment set for this test.

Table 1 Temperature Equilibrium Algorithm for task allocation in a Dynamic Environment

(1) Agent T_i sets the priority value V_i for the corresponding resource to every generated job assignment. V_i is a numeric floating point value rounded up to one decimal place, considering $0.0 \leq V_i \leq 9.9$.

(2) The coordinator agent A_j, gathers the temperature values t_j, specific to every agent, constituting a set defined by: $T_j = \{t_j, t_{j+1}, .. t_{j+n}\}$ for assigning each job order. Every T_j value is a double precision random numeric value rounded up one decimal place, such that $0.0 \leq t_j \leq 9.9$ for all the elements in the

set.

(3) A_i verifies every temperature value in order to obtain the highest element, defined by: M=max (T_{ij}).

(4) Once the highest value is obtained, $M \geq 8$ is evaluated. When valid, the agent is considered "hot in first degree". 8 tenths are substracted from its original value and distributed equally to the other agents for searching a more equilibrated heat distribution into the MAS.

(5) $6 \leq M < 8$ is evaluated; when valid, the M agent is considered "hot in second degree". 6 tenths are substracted from its original value and distributed equally among the other agents for searching a more equilibrated heat distribution into the MAS.

(6) Whenever $M < 6$ is valid, the corresponding agent is considered "cold". No temperature equilibrium protocol is activated.

(7) When $M >= V_i$ is valid, the active task is allocated to the corresponding t_j agent. M-1 is applied, representing the workload increment in that agent.

(8) On the other side, when $M < V_i$ is valid, the allocation process is considered null. The simulator is restarted to generate a new set of agents.

For this case, three jobs are generated using the numeric format specified on Table 1. In this case, the values are 1.4,0.8 y 8.3 corresponding to the 1st, 2nd and 3rd job tasks. In the same manner, the heat values for the agents are generated, obtaining 7.5 as the highest value, which leads us to consider the agent as "hot in second degree". For this reason, 8 tenths of its value are substracted, distributed uniformly among the rest of the agents. Each job value is compared against the most capable agent. In the example shown in the figure, 2 tasks are successfully allocated and 1 remains unassigned.

3. Rescheduling

The Rescheduling process can be defined in general terms as a dynamic adjustment that updates the production scheme in response to sudden interruptions susceptible of occurrence in the manufacturing shop floor. There are diverse strategies that specify how and when to apply rescheduling to properly confront those occurrences. The integration of multi-agents in different aspects of manufacturing has increased in the recent years. Kornienko et al [24] have studied flexible manufacturing, on which rescheduling is a core concept applying multi-agents, even proposing a MAS repairer of damaged plans in a manufacturing environment [25]. Within the scope of this work, the next rescheduling strategies will be considered: High Priority and High Utilization [16]. For the proper application of this test, the software components developed are shown in Figure 3. In the above figure, it can be noted that the WinTempAg class defines the equilibrium protocol into the system, which applies the principles of the temperature model [17], described briefly on the following section.

3.1 Temperature model

This model allows determining the amount of workload ("heat") inherent for each agent into the MAS and, in consequence, it´s useful for achieving a better distribution of the generated tasks based on three main concepts:

Relative heat, temperature and latent heat. Relative heat Ht (Aij) measured during a t time lapse represents the active workload corresponding to agent Aij. This concept is defined as follows:

$$H_t(A_i^j)=D_b+D_e-D_p$$

Where:

$$D_b=\sum_{T_x=0}^{b} m(T_x)/qT_x$$

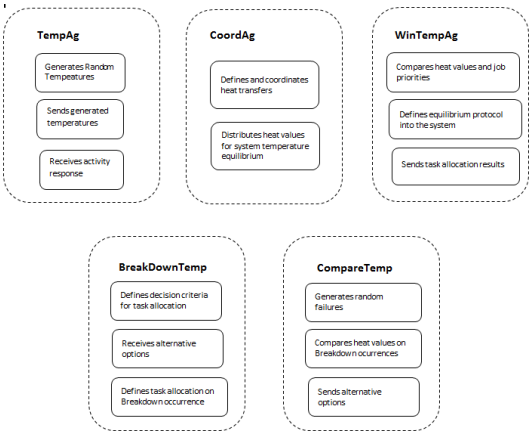


Fig. 3 Software classes for Rescheduling tests

The b sub index represents the entire set of waiting activities. Tx=0 is the reference for the task under execution. Every job is associated to a quality service attribute qTx with a value range from 0 to 1. On the other side, De represents the total duration of the new job orders during the t lapse. It is assumed that the activity generation follows a Poisson distribution with an arrival rate represented by λ and an average defined by μ. From here:

$$D_e=\sum_{T_x=1}^{\lambda t} m(T_x)/qT_x$$

Where Dp represents the total duration of the processed activities on time t:

$$D_p=t / \mu$$

From here, the temperature for an agent on a t interval is defined as follows:

$$T_t(A_i^j)= H_t(A_i^j)/t$$

The concept of latent heat is related to the price involved into the activity transference between two agents with no incidence on the temperature increasing. For that reason, is not considered in the developed simulations of this study.

3.2 Rescheduling strategies

The first applied test to the MAS under rescheduling using the temperature equilibrium algorithm is the High Priority test. The change in job priority is considered one of the rescheduling factors [29] that can change the system status and affect performance. For practical purposes, a flexible manufacturing environment will be considered; on this, jobs continue to arrive over an infinite time horizon, conforming a dynamic rescheduling working strategy. Once the above defined concepts were considered, the application scheme for this test was developed, just as described in table 2.

Table 2 High Priority algorithm under Rescheduling conditions

(1) Agent $\mathcal{T}_i$ assigns the priority value V_i from the corresponding resource to each generated job order. V_i is a numeric floating point value rounded up to one decimal place, into the $0.0 \leq V_i \leq 9.9$ interval

(2) The coordinator agent A_j, gathers the temperature values t_j, specific to every agent, constituting a set defined by: $\mathcal{T}_j = \{t_j, t_{j+1}, .. t_{j+n}\}$ for assigning each job order. Every $\mathcal{T}_j$ value is a double precision random numeric value rounded up one decimal place, such that $0.0 \leq t_j \leq 9.9$ for all the elements in the set**.**

(3) Each t_j, value represents the workload for every individual agent. For practical purposes b= t_j is considered.

(4) λ is calculated as the probability that the randomly generated job orders value falls between the 5 to 7 interval (high priority). The obtained average is μ=5 from the dynamic allocation tests.

(5) An average time value of *t*=10ms is considered from the previously performed dynamic allocation tests

(6) H_t is calculated using the previous data for every agent in the MAS

(7) Once $H_{t,,}$ is obtained, the equilibrium protocol is initiated, distributing the "hot" agent workload to the "cold" agents proportionally to the temperature difference previously calculated. If this value is low, the general temperature of the system remains unaltered..

(8) A_i verifies each temperature value obtained from the previous stage and calculates the maximum from the generated set: $M=\max(\mathcal{T}_i)$.

(9) A failure condition is generated randomly in the agent with the best job acceptance conditions. The second best agent is then selected as replacement. Operation time increment due to this occurrence is defined by: $t_{ij} = t_{ij} + \Delta t_{ij}$

(10) If, for each T_{ij} **element** $t_j = t_{j+1} = \ldots t_{j+n}$ is true, t_j is then designated as the maximum value for resource allocation, whether a failure occurrence is present or not.

(11) If $M >= V_i$, is true, the resource is then allocated to the corresponding t_j agent. M -1 is then applied representing the workload increment in the agent.

(12) On the other side, when $M < V_i$ is valid, the allocation process is considered null. The simulator is restarted to generate a new set of agents.

3.3 High utilization algorithm

This algorithm is basically applied under the same working conditions considered for the High Priority algorithm. The main difference is that the latter, once activated the failure occurrence in the most capable agent, selects as an alternative the agent with the least allocation capacity from the MAS. The general assumptions for all the simulations are: (a) the job arrivals, processing and departures are deterministic, (b) the machine failures are generated randomly and (c) every machine can only process one job at a time. This algorithm is described in Table 3.

Table 3 High Utilization Algorithm under Rescheduling

(1) The agent T_i assigns the priority value V_i for every resource corresponding to a generated job order. V_i is a numeric floating point value rounded up one decimal place, such that $0.0 \leq V_i \leq 9.9$.

(2) The coordinator agent A_j, reveals each t_j temperature value, specific for every agent, conforming a set defined by: $T_j = \{t_j, t_{j+1}, .. t_{j+n}\}$ to allocate all the job orders. Every T_j value is a random double precision numeric value rounded up one decimal place, such that $0.0 \leq t_j \leq 9.9$ is valid for every element included in the set.

(3) Every t_j value represents the active workload in the individual agent. It is assumed that $b = t_j$,

(4) The value λ is calculated as the probability that the high priority job orders falls in the range between 5 and 7. The average among them is μ=5, obtained from the dynamic allocation tests..

(5) A mean time value of t=10ms is considered, from the average measurements obtained from the dynamic allocation test.

(6) H_t is calculated using the previusly measured data for every agent inside the MAS

(7) Once obtained $H_{t,,}$ sthe equilibrium protocol is initiated. The workload from the "hot" agent is proportionally distributed to the "cold" agents, considering the calculated temperature values. If the measured value doesn´t imply a high value, the system remains unaltered..

(8) A_i verifies every resultant temperature value and obtains the maximum from the generated set, defined by: $M = \max(T_i)$.

(9) A failure condition is randomly generated in the most capable agent. The least apt agent (the one who accomplishes $m = \min(T_i)$) is selected as the main element for job allocation. The operation time increases due to the failure in the form $t_{ij} = t_{ij} + \Delta t_{ij}$

(10) If $t_j = t_{j+1} = \ldots t_{j+n}$ is valid for every element in T_{ij}, then t_j is selected as the maximum value for resource allocation purposes with or without failure ocurrence.

(11) If $m >= V_i$ is true, the resource is allocated for the corresponding t_j agent. $m - 1$ is applied, representing the workload increasing for the agent.

(12) Otherwise, if $m < V$, is true, the allocation is considered deserted. The simulator is restarted in order to generate a new set of agents.

3.4 Priority change algorithm

For the proper application of this test, two priority levels (random values) are generated for the active job order. Once generated, both are compared against an agent subsystem integrated with the most and least capable agent in order to evaluate their behavior. The algorithm is described on Table 4.

Table 4 Priority Change Algorithm

(1) The agent T_i assigns the initial and final priority value sets V_{ij} and H_{ij}, respectively , corresponding to

every generated job order. $\mathcal{H}_i > V_i$ is valid for all cases.

(2) The coordinator agent $\mathcal{A}_j$ presents the temperature values t_j, specific for each agent, confirming a set defined by: $T_j = \{t_j, t_{j+1}, .. t_{j+n}\}$ to properly allocate every job order. All T_j are randomly generated, double precision numeric values, rounded up one decimal place, such that $0.0 \leq t_j \leq 9.9$ is true for all the elements in the set.

(3) $\mathcal{A}_j$ verifies every resultant temperature value, obtaining the maximum and minimum values of the generated set. These are defined by: $M = \max(T_i)$ y $m >= V_i$, respectively.

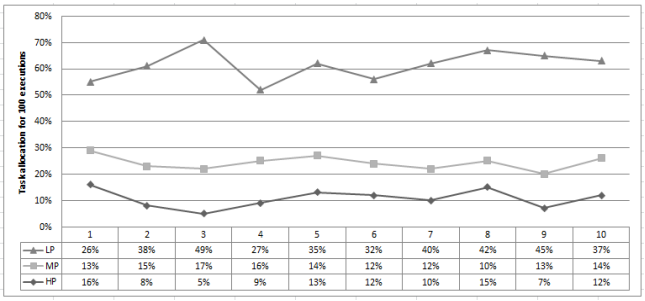
(4) $K = M + m$, is applied, defined as the total system capacity for processing the job order..

(5) If $K < V_i$, is true, the allocation process is considered deserted, and a new execution process is initiated. This is interpreted as system inability for properly processing orders due to saturation.

(6) If $K >= V_i$ is true, the system is evaluated before an aventual priority change for all probable increment scenarios.

4. Results and Discussion

4.1 Dynamic environment test



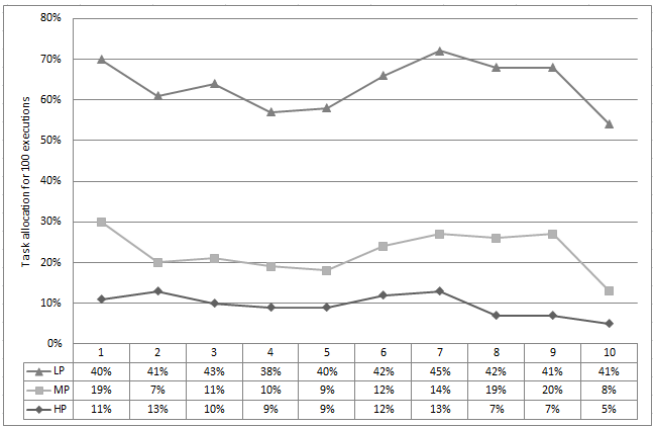
	1	2	3	4	5	6	7	8	9	10
LP	26%	38%	49%	27%	35%	32%	40%	42%	45%	37%
MP	13%	15%	17%	16%	14%	12%	12%	10%	13%	14%
HP	16%	8%	5%	9%	13%	12%	10%	15%	7%	12%

Fig. 4 Job allocation for Dynamic environment test

The dynamic environment test was simulated with 100 iterations for every 1 to 10 new job orders generated with random priorities. The priority range values were set as follows: 8-9 for high priority, 4-7 for medium priority, and 0-3 for low priority. The results shown in Figure 4 reveal that the high priority job allocation average decreases when the number of generated jobs increases in the system. When one job order is generated, the average result for 100 iterations reaches 16%. On the other hand, the allocation average corresponding to medium priority jobs remains almost equal for 1 to 10 jobs generated.

Similarly, it can be noted that a very high amount of allocated jobs correspond to the low priority level. Also, the unallocated jobs index with one job is very high (39%), while it remains almost stable (10,5% on average) when the number of jobs increases.

4.2 High priority test



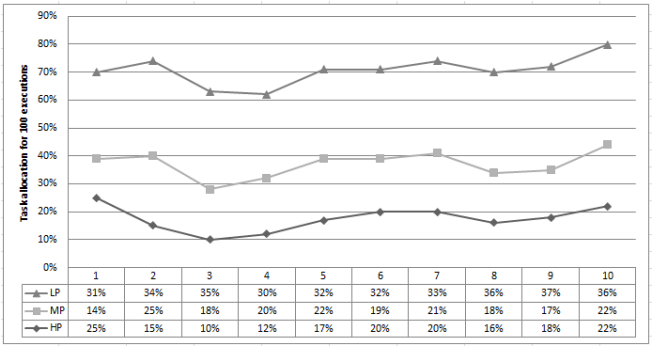
	1	2	3	4	5	6	7	8	9	10
LP	40%	41%	43%	38%	40%	42%	45%	42%	41%	41%
MP	19%	7%	11%	10%	9%	12%	14%	19%	20%	8%
HP	11%	13%	10%	9%	9%	12%	13%	7%	7%	5%

Fig. 5 Job allocation for High Priority test

The above figure shows the distribution of completed job orders including the random failure generation (Breakdown) in the best agent of the system, and applying the temperature equilibrium protocol when required. The average distribution of completed orders is: 40%, 14% and 10% for Low, Medium and High priorities, respectively.

In order to develop a more realistic simulation of the working environment while incorporating the intrinsic characterisitics of the proposed algorithm, some modifications were added to the software model; among them, the development of Java classes allowing both the random flow generation based on the Poisson distribution and the generation of a random set of numerical values rounded up one digit for better precision. This model was tested and analyzed including the necessary rescheduling conditions.

4.3 High utilization test



	1	2	3	4	5	6	7	8	9	10
LP	31%	34%	35%	30%	32%	32%	33%	36%	37%	36%
MP	14%	25%	18%	20%	22%	19%	21%	18%	17%	22%
HP	25%	15%	10%	12%	17%	20%	20%	16%	18%	22%

Fig. 6 Job allocation for High Utilization test

For this test, the High Priority job tasks allocation performs well, reaching an average 25%, a better result than the obtained with the previous tests. On the other side, the unallocated job tasks index remains almost at the same level than before with an average of 30. In this specific environment, the simulator defines the least apt agent as an alternative once the failure (Breakdown) has occurred in the main agent. One of the main characteristics of software agents is autonomy. From the code perspective, three static functions were developed representing this ability to manage the required agent change decision properly.

In general, under High Utilization conditions, the medium priority job orders allocation is the best obtained of the all the rescheduling simulations.

4.4 Priority change

The next figure shows the results of applying the Priority Change test in the temperature equilibrium environment. The average increment inside the system was evaluated when a random priority change in the job orders was applied. Similarly, the table indicates the generated values.

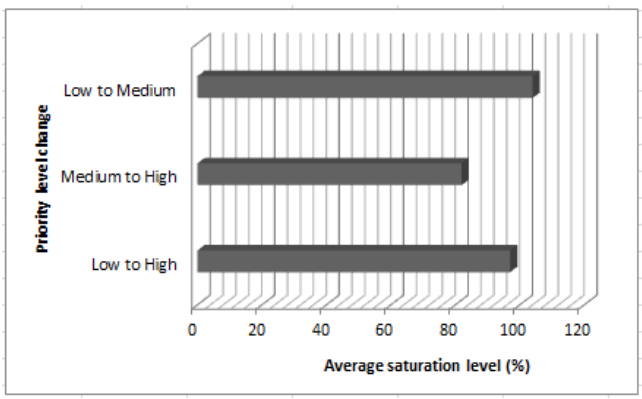


Fig. 7 Average saturation level. Priority Change test results

Under the above conditions, the simulator generates randomly the priority values from 0 to 9. To determine the joint priority, since two job tasks of different value arrive simultaneously, the following conditions are considered:

- Low: the sum is between the 0 to 6 range.
- Medium: the sum is between the range of 7 to 12.
- High: The sum is beween the range of 13 to 18.

For practical purposes, if the system is capable of receiving the job order with an initial priority level but is not able to allocate it when the priority changes to a higher level, an oversaturation is declared inside the system.

Table 5. Average saturation levels inside the system when allocating the job orders with random priority change.

Experiments	Low to Medium priority	Medium to High priority	Low to High priority
1-20	97%	79%	108%
21-40	95%	81%	98%
41-50	99%	85%	105%

From the data in the above table and figure, the obtained average saturation level reaches 94% in the transition to a higher priority. Also, only for the Medium to High priority transition level an adequate performance is observed.

5. General results and conclusions

On first instance, and in order to obtain the final results for the dynamic test, a summatory for the completed job orders was conducted in the three priority levels. In the same way, a concentrated result corresponding to the unallocated orders was obtained. Once obtained these values, it can be noted that the system presents some problems for completing job tasks of both medium and high priority, obtaining just a 25%. In comparison, the percentage level obtained for low priority job tasks is relatively high, reaching 38%; however, the unallocated orders level is equiparable to the latter, with 37% average, representing three times value corresponding to the high priority completed orders.

The High Priority test results are very similar to the previously obtained for the Dynamic Allocation test: the total amount of unallocated orders is higher than the corresponding value obtained for the high and medium priority jobs; however, a significant increment in low priority jobs allocation is observed , representing almost 50% of the total generated orders.

For the High Utilization test, despite the failure occurrence in the agent with the best allocation capacity, the total amount of successfully completed job orders increments significantly in reference to the results previously obtained (figure 10). Furthermore, the final distribution results arranged by priority level, reveals an increased amount of high priority job orders succesfully completed, reaching more than 30% on average, and in consequence, the low priority level job orders successfully allocated diminishes for this test, representing a 17% on average.

Given that the maximum saturation level must correspond to 100%, it can be concluded that the system is highly saturated under this working conditions.

From all of the above, it can be concluded that the Temperature Equilibrium algorithm is a robust and feasible alternative for working conditions with high probability of failure occurrence and/or high saturation levels. The test with the highest rejection index – Priority Change – is relatively good, considering the low average saturation level observed in the transition from medium to high priorities. Under dynamic working conditions with minor failure occurrence probability, the results shown reveal a lesser performance, evidencing that the strength of this algorithm resides on collaborative and equillibrated work, and in the first tests applied, only the main agent is considered for allocation purposes.

References

[1] Journal of Intelligent Manufacturing (2000) 11, 403-419.M.G. Mehrabi, A.G. ULSOY and Y. Koren.
[2] J. Blazewicz, J.K. Lenstra, A.H.G. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity, Discrete Appl. Math. 5 (1983). 11–24.
[3] Y. Yan, T. Kuphal, J. Bode, Application of multiagent systems in project management, Int. J. Prod. Econ. 68 (2000) 185–197.
[4] Koestler, A., 1967, The Ghost in the Machine, PAN Book, London.
[5] Valckenaers, P., Van Brussel, H., 2005, Holonic Manufacturing Execution Systems, Annals of the CIRP, 54/1: 427-432.
[6] Fu-Shiung Hsieh. "Deadlock Free Task Distribution and Resource Allocation for Holonic Manufacturing Sytems Based on Multi-Agent Framework", 2001.
[7] Verderame, P.M. & Christdodoulos, A.F. (2008). Integrated operational planning and medium-term scheduling for large-scale industrial batch plants. Industrial & Engineering Chemistry Research, 47(14),4845-4869.
[8] W. Shen and D.H Norrie. Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and In- formation Systems, an International Journal, 1(2):129–156,1999.
[9] R.L. Axtell, Why agents? on the varied motivations for agents in the social sciences, in: C.M. Macah, D. Sallach, (Eds.), Proceedings of the Workshop onAgent Simulation: Applications, Models, and Tools, Argonne, Illinois, Argonne National Laboratory, 2000.
[10] N. Gilbert, Agent-Based Models, Sage Publications Inc., 2007.
[11] J. Wang, P.B. Luh, X. Zhao, Wang Jinlin, An optimization-based algorithm for job shop scheduling, Sadhana, J. Indian Acad. Sci. 22 (Part 2) (1997) 241–256.
[12] J. Wang, P.B. Luh, X. Zhao, Wang Jinlin, An optimization-based algorithm for job shop scheduling, Sadhana, J. Indian Acad. Sci. 22 (Part 2) (1997) 241–256.
[13] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review , 10(2):115–152, 1995.
[14] Effective Resource Management in Manufacturing Sytems. Springer Series in Advanced Manufacturing.2006. p. 5.
[15] T.N. Wong et al. / Experts Systems with Applications 31 (2006) 486-494.
[16] Jain, A. K. and Elmaraghy, H. A., 1997. Production schedul-ing/rescheduling in flexible manufacturing. International Journal of Production Research, 35(1), 281-289.
[17] Martyn Fletcher and S.Misbah Deen. "Temperature equilibrium in Multi-Agent Manufacturing Systems". DEXA Proceedings of the 11th International Workshop on Database and Expert Systems Applications. IEEE Computer Society Washington, DC, USA 2000 ISBN:0-7695-0680-1.
[18] R. Zimmermann, S. Winkler, F. Bodendorf, Supply chain event management with software agents, in: S. Kirn, O. Herzog, P. Lockemann, O. Spaniol (Eds.), Multiagent Engineering – Theory and Applications in Enterprises, Springer, Berlin-Heidelberg, 2006, pp. 157–175.
[19] M. Calisti, P. Funk, S. Biellman, T. Bugnon, A multi-agent system for organ transplant management, in: A. Moreno, J. Nealon (Eds.),Applications of Software Agent Technology in the HealthCare Domain, Birkha¨user, Basel, 2004, pp. 199–212.
[20] H. Lee, P. Mihailescu, J. Shepherdson, Realising team-working in the field: an agent-based approach, IEEE Pervasive Computing, June 2007, pp. 85–92.
[21] Bellifemine, F., Caire, G., Poggi, A., & Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications. Lessons learned. Information and Software Technology, 50(1), 10-21.
[22] FIPA Standard website. www.fipa.org
[23] McFarlane, D., Sarma, S., Chirn, J. L., Wong, C. Y., & Ashton, K. (2002, July). The intelligent product in manufacturing control and management. In 15th Triennial World Congress, Barcelona, Spain.
[24] S. Kornienko, O. Kornienko, P. Levi, Flexible manufacturing process planning based on the multi-agent technology, in: proceedings of the 21st International Conference AIA'03, Innsbruck, Austria, 2003 pp. 156–161.
[25] S. Kornienko, O. Kornienko, P. Levi, Multi-agent repairer of damaged process plans in manufacturing environment, in: Proceedings of Eighth International Conference on Intelligent Autonomous Systems, Amsterdam, March 2004.

[26] G.M.P. O'Hare, N.R. Jennings (eds.): Foundations of Distributed Artificial Intelligence. John Wiley & Sons, New York, NY, USA, 1996.
[27] [9] H.V.D. Parunak: Industrial and Practical Applications of DAI. In G. Weiss (ed.), Multi-Agent Systems, pp. 377–421. MIT Press, Cambridge, MA, USA, 1999.
[28] Bussmann, S., & Schild, K. (2001, October). An agent-based approach to the control of flexible production systems. In Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on (Vol. 2, pp. 481-488). IEEE.
[29] Dutta, A., "Reacting to scheduling exceptions in FMS environments." *HE Trans,* 22(4),300-314(1990).