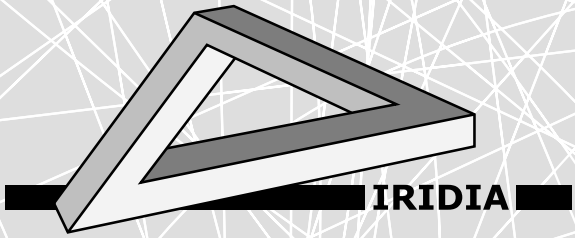


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

An Ant-Based Algorithm for the Dynamic Task Allocation Problem

Roberto Ghizzioli, Shervin Nouyan,
Mauro Birattari, and Marco Dorigo

Technical Report No.

TR/IRIDIA/2004-16

September 2004

An Ant-Based Algorithm for the Dynamic Task Allocation Problem

Roberto Ghizzioli, Shervin Nouyan, Mauro Birattari and Marco Dorigo
IRIDIA
Avenue Franklin Roosevelt 50,
CP 194/6 - 1050 Bruxelles - Belgium
rghizzioli@iridia.ulb.ac.be
{snouyan,mbiro,mdorigo}@ulb.ac.be

Abstract

Different multi-agent algorithms, based on a model of division of labor of social insects, are applied to an online scheduling problem and compared. A painting facility, consisting of trucks and painting booths, can be taken as example of the problem. Trucks roll of an assembly line to get painted by booths. Crucial to this problem is to minimize the makespan that is the time until all tasks in the system have been processed. Additionally, in this scenario booths may be identical or heterogeneous in the meaning that different booths require different process times for the same task. In particular, based on previous work of Cicirello *et al.* [8], we propose an insect-based algorithm that increases the plasticity of the agents to the environmental changes and their specialization level based on their characteristics. By an empirical analysis we show that, for the classes of instances considered, our algorithm performs better than the others considered. Additionally, we emphasize the contribution to the makespan of each introduced rule.

1 Introduction

The use of Multi-Agent Systems (MAS) is rapidly increasing in a variety of fields of computer science, engineering and artificial intelligence in the last years. A multi-agent system can be described as a loosely coupled network of agents interacting to collectively solve problems which are beyond the capabilities of each individual agent [10]. Agents are autonomous systems capable of adapting to changing environments and able to exhibit goal-directed behaviors [26].

In this paper our interest is in using multi-agent algorithms to solve the Dynamic Task Allocation (DTA) problem. The DTA problem is an online, non-deterministic scheduling problem, that is, a decision-making process for assigning tasks to agents working in parallel. Each task belongs to a particular type and if a working agent changes the type of task it performs, a setup is required and a given cost is incurred. In the DTA problem different agents can have different processing times for different task types. If this is the case, the DTA problem is said heterogeneous; otherwise, if all the agents are identical, the problem is said homogeneous (DTA_{Hom}). The objective for this problem is to minimize the makespan, that is, the time until all tasks in the system have been processed.

Different work has been done using multi-agent algorithms on the homogeneous case of the problem. Most of the proposed algorithms use paradigms based on the *specialization* concept, where agents tend to specialize for one type of task, to avoid unnecessary reconfiguration and in this way increase the efficiency of the system. Morley [14] is the first one that solved a painting problem similar to the homogeneous case of the DTA problem. His algorithm was adopted in the GM facility and was found that his system performs 10% better than the previously used centralized

scheduler. Later, Campos *et al.* [7] proposed to solve the same problem with an insect-based approach. Subsequently, Cicirello *et al.* [8] proposed another insect-based algorithm introducing concepts not considered by Campos *et al.*

In this paper, we present *Ant Task Allocation* (*ATA*), an algorithm for the homogeneous case of the problem based on the work of Cicirello *et al.* The algorithm of Cicirello *et al.*, especially for problems in which demands change dynamically, often takes much time to re-adapt or does not succeed to re-adapt at all. In order to overcome this problem we propose some modifications and additional rules to speed up the adaptation process.

Applying these algorithms to the heterogeneous case of the *DTA* problem, we found that they perform poorly because the agents do not consider their process speed to perform a task. Therefore, we propose a *Different Process Speed* (*DPS*) rule applicable to all the considered insect-based algorithms. This rule is inspired by division of labor as observed in different castes of *Pheidole* ants.

We compare all presented algorithms on the *DTA* problem. Additionally, we introduce a non-adaptive algorithm in the experimental analysis in order to have a performance reference. The comparison considers two possible real-world situations: a big painting factory with identical agents and a medium dimension factory with two heterogeneous subsets of agents. Particular attention has been payed to the experimental conditions; in fact, we have used two instance generators in order to obtain the two classes of instances of the problem and we have done a rigorous parameters tuning for each used algorithm. Furthermore, a statistical analysis has been done to study the data obtained. We will show that our algorithm achieves the best results for both the considered class of homogeneous and heterogeneous problems. Moreover, we will show that all adaptive algorithms in all situations achieve better results than the non-adaptive algorithm. Again, Campos' *et al.* algorithm does not obtain good results in both the experiments but we are grateful to him because his work has the merit of having introduced the insect-based approach to solve this problem. Moreover, the performance of the insect-based algorithms are ranked according to the time they was made. In fact the algorithm of Campos' *et al.* does not obtains good results, the algorithm of Cicirello *et al.* gives a significant improvement to the performance obtained by Campos' *et al.* as they show in paper [8] and our algorithm improves again the results of the solution of Cicirello *et al.*

Another set of experiments deals with the analysis of the proposed rules. First of all we show that *ATA* with all the introduced rules performs better than the original algorithm of Cicirello *et al.* Moreover, we observe that one of the proposed rules gives a large contribution to reduction in the makespan, two others a significant contribution to it and the last one does not give any apparent contribution to the results. Finally, we show that the *DPS* rule improves the performances of all the insect-based algorithms on the heterogeneous case of the *DTA* problem.

In Section 2 we give a formal description of the *DTA* problem and use the metaphor of a painting facility to describe it in a simple way. Section 3 presents the related work for the homogeneous case of the *DTA* problem. We present two approaches: the market-based approach with the algorithm of Morley and the insect-based approach with the algorithms of Campos *et al.* and Cicirello *et al.*. Afterwards, Section 4 introduces *ATA*, our algorithm for the DTA_{Hom} problem. In Section 5 we describe the *DPS* rule which is applicable to the insect-based algorithms in the heterogeneous case of the problem. Section 6 details the experimental setup and the instance generators. Furthermore, a summary of the algorithms and of the parameter is given. In the second part of this section, we analyze the obtained results. Finally, we conclude in Section 7.

2 Problem Definition

The problem considered here is a particular non-deterministic scheduling problem that we refer to as the Dynamic Task Allocation (*DTA*) problem. This problem is a decision-making process for assigning tasks to parallel working machines. In the *DTA* the machines differ in their process speed. Moreover, tasks information is only known at execution time. A very similar problem,

with identical machines, was presented and solved by Morley [14] for painting trucks at General Motors.

The *DTA* problem can be described by considering a painting factory environment.

Painting facility. Trucks roll off an assembly line at the end of which there is a storage where each truck waits to be assigned to a painting booth. The number of available colors is fixed and a truck's color is predetermined by a customer order. The distribution of colors of trucks and the truck release dates are not specified.

A painting booth is an agent able to paint with all the available colors. Booths may be identical or may have different process times for the same type of tasks. This situation may happen for example in a factory with old and new machines, where the new ones require less time to accomplish a same task. Another possibility is that a set of agents do some work faster than another and vice versa. Moreover, each booth has a fixed queue length which can be filled by trucks. If the color of a painting booth must be changed, a setup time is necessary. For example, if a booth is applying red and the next truck to be processed by that painting booth requires white, a fixed flush time is taken. If no setup is necessary, the booth starts immediately to paint the next truck in its queue. A setup may also be related to a monetary cost that for example is the amount of paint lost during a swap.

The objective of this problem is to assign trucks to painting booths minimizing the makespan, that is, the completion time of the last truck in the system. We also analyze the number of setups and the storage dimension that should be kept as low as possible.

2.1 Dynamic Task Allocation (DTA) Problem

According to Pinedo [16] we formalize the scheduling problem introduced by the following notation

$$< Qm|r_j, s_{j,h}|C_{MAX} >, \tag{1}$$

where each variable has the following meaning:

- **Machines in parallel** Qm: in the system there are m machines in parallel that are in charge of m booths in the paint facility. Machines may have different process times for different types of tasks. Groups of machines can be grouped in different subsets S_i. The process times of a subset S_i for the types of tasks c_j is given by $t_{proc}(S_i, c_j)$. The setup time t_{setup} is identical for all the working machines. Furthermore, each machine k has a queue that can be filled with up to q_k tasks. If all machines have the same processing time t_{proc} for all types of tasks, we refer to the homogeneous case of the problem (DTA_{Hom}).

- **Release date** r_j: this is the time at which a task j is released to the painting facility. The value of r_j is not available at the beginning of the experiment. It is the earliest time at which the task j can start being processed. Each tasks j requires a single operation and may be processed on any one of the m machines. Furthermore, each task j belongs to a particular type c_j which is the color to be painted and also this information is not available at the beginning of the experiment.

- **Sequence dependent setup times** $s_{j,h}$: represents the sequence dependent setup time between two tasks j and h. If $c_j = c_h$, no setup is required and $s_{j,h} = 0$. This means that the respective machine can immediately start processing task h after having finished to process task j. If $c_j \neq c_h$, a configuration time t_{setup} is needed. The setup time between task j and task h is independent of the machine.

- **Makespan** C_{MAX}: the objective of the DTA problem is to minimize the makespan that is the time until all tasks in the system have been processed.

In the DTA problem, the task generation process is not specified. For instance, the release dates and the types of the tasks may be distributed exponentially or normally. This distributions may

vary dynamically so that at a random time the probability mix changes and the machines need to adapt to the new environment. It also may happen the extreme case of all tasks being create before the scheduling process start. This last example would make the *DTA* instance a very simple scheduling problem requiring only a simple algorithm to optimize a schedule because everything in known a priori.

3 Related Work

In the previous section we used a painting facility as a real world example of the DTA_{Hom} problem. This example originates from Morley [14], who was given the task of optimizing a scheduler for a General Motors truck painting facility. Morley used a decentralized market-based approach that, compared to the previously used centralized scheduler, improved the performance in terms of decreasing the amount of paint lost. In the following section we give a short overview of market-based algorithms for problems that are related to the DTA_{Hom} problem. In particular, we detail Morley's approach.

Independently of each other, Campos *et al.* [7] and Cicirello *et al.* [8] used similar insect-based approaches to solve the same problem. They were inspired by a threshold model proposed by Bonabeau *et al.* [2, 3] and Theraulaz *et al.* [22]. In Section 3.2 we first introduce the threshold model and then explain the two approaches.

3.1 Market-Based Approach

Market-based approaches are often used for coordinating asynchronous scheduling operations in the face of imperfect knowledge [9, 13, 12]. The decision process is based on a decentralized bidding mechanism where autonomous agents bid for a task or a resource, which is then assigned to the agent with the highest bid. The agents dynamically adjust their bids according to their capability to solve a task or according to the availability of a resource.

For example, Waldspurger *et al.* [24] presented a computational system called *Spawn*, in which each task bids for the use of machines on a network. According to its priority, each task is given a certain budget, which it has to use such that it will be assigned to the resources it requires. Prices are adjusted dynamically and are based on the demands of other tasks.

Another example is given by Schwartz *et al.* [20] that use the bidding mechanism for data allocation of self-motivated data servers with no common preferences and no central controller. The location of each data unit is determined using a bidding mechanism where the server bidding the highest price for obtaining the data will actually obtain it. This market-based approach yields an efficient and fair solution, a simple implementation and the bidders are motivated to offer efficient prices.

Clearwater *et al.* [11] also use a market-based approach for a thermal resource distribution problem: computational agents that represent individual temperature controllers bid to buy or sell cool or warm water air.

All these examples show that market mechanisms provide methods to deal with coordinating asynchronous operations in the face of imperfect knowledge. A computational system set up along market rules can allow the system as a whole to adapt to changes in the environment or disturbances to individual members. Additionally, the price mechanism behind this approach simplifies the design of adaptive algorithms.

3.1.1 Morley's Algorithm

The following algorithm definition originates from the paper of Campos *et al.* [7]. In fact, the algorithm developed by Morley is a manufacturing application and many details are protected.

Morley [14] proposed a market-based algorithm to solve an example of the homogeneous case of the *DTA* problem which was applied to a General Motors factory. We refer to his algorithm as *Market-Based Approach* (*MBA*). In his market-based approach booths bid for trucks to paint

them. For each booth the intensity of the bid for a particular truck is based on the amount of time and the additional costs that would be related to an assignment. If the queue of a booth is full, it does not participate to the bidding process. Morley proposed a bid function applied to each booth that considers the importance of the tasks, the color of the presented task, and the state of the booth. More precisely, if we consider a task j of color c_j that is in the storage, a booth k that does not have a full queue participates in the bidding process with a range of values given by:

$$B_k(j) = \frac{P * w_j * (1 + C * e(k, j))}{\Delta T_k(j)^L}, \tag{2}$$

where w_j gives the importance of the task, $e(k, j)$ is a function that is 1 if the color of the last truck in the queue is equal to c_j, and 0 otherwise. P, C, and L are parameters that weight each component. $\Delta T_k(j)$ is the time until task j starts to be painted, and is determined by the following equation

$$\Delta T_k(j) = q * t_{proc} + n * t_{setup} + t_{working}, \tag{3}$$

where q is the number of trucks in the queue of booth k, t_{proc} is the time required to paint one truck, n is the number of setups between trucks in the booth's queue, t_{setup} is the time required for a setup, and $t_{working}$ is the time necessary to finish the truck currently being painted.

The assignment process is very simple. For all trucks each booth with space in its queue bids for them according to Equation 2 and the respective truck is assigned to the end of the queue of the highest bidder. If all the queues are full no booth can bid and the truck stays in the storage. If more than one booth submits the same highest bid, the respective truck is assigned to the booth that requires no setup to paint it. In the case that all competing booths require a setup, the winner is chosen randomly. If none of the booths requires a setup, the truck is assigned to the booth with the shortest queue size or, in case all the booths have the same queue length, it is assigned randomly.

3.2 Insect-Based Approach

Campos *et al.* [7] and Cicirello *et al.* [8] used an insect-based approaches to solve the DTA_{Hom} problem which are inspired by the methodology of division of labor in social insect. Insect societies perform different activities simultaneously exploiting specialized individuals. This *parallelism* is undoubtedly more efficient than sequential task performance by unspecialized workers because *individual specialization* leads to a higher grade of colony efficiency as specialized individuals don't switch between one type of task and another, a process that often requires time. Moreover, a specialized individual can perform a type of tasks more efficiently.

A key feature of division of labor is *plasticity*. The ratios of workers performing the different tasks can vary in response to internal and external challenges. Factors such as food availability, climatic conditions or phase of colony development influence the specialization of the colony's workers. In the *Pheidole* ant species, for example, there are minor workers that are smaller and morphologically distinct from major workers. Minor and major workers tend to perform different tasks: while majors cut large preys and defend the nest, minors feed the brood or clean the nest. Wilson [25] showed experimentally that when removing minors, majors get engaged in the tasks usually performed by minors to replace them.

The fixed threshold model In order to explain Wilson's [25] observations, Bonabeau *et al.* [4] have developed a simple model that relies on response threshold [18, 17]. The idea behind the fixed threshold model is that each individual has a *threshold* value for every kind of *task* available in an environment. A threshold represents the level of specialization of an agent for a particular task. A task emits a *stimulus* to attract the individuals' attention. Based on the stimulus, an individual will or will not accept to start performing the respective task. The higher the intensity of a stimulus the higher the attraction toward workers to accept that task. This model is said to be with *fixed* thresholds because these thresholds do not change over time.

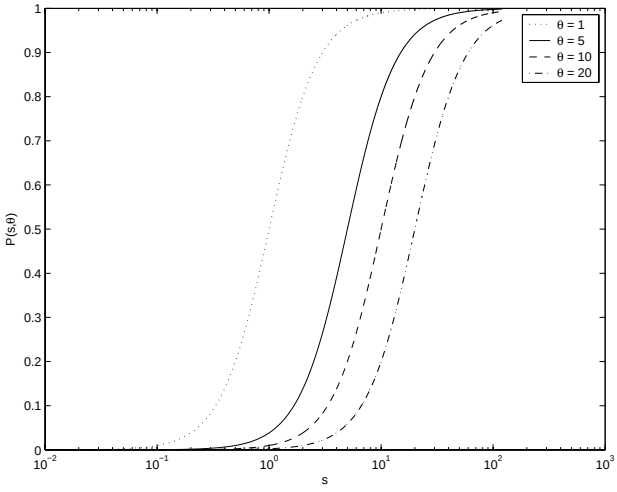


Figure 1: Response curve $P(s, \theta)$ with different thresholds.

The first question that we have to answer is: how do we formally define a response threshold? Assume that m tasks need to be performed. Each task j is associated with a stimulus s_j, the level of which increases if it is not satisfied. Let us assume that there are N agents, denoted by i, with *response threshold* $\theta_{i,j}$ $(i = 1, ..., N)$ for each task j. In the threshold model, individual i engages in task j with probability

$$P(s_j, \theta_{i,j}) = \frac{s_j^2}{s_j^2 + \theta_{i,j}^2}. \tag{4}$$

This equation shows that for $s_j \ll \theta_{i,j}$ the probability of an individual i to be engaged in task j is close to 0 and for $s_j \gg \theta_{i,j}$ the probability is close to 1. Therefore, agents i with a lower threshold $\theta_{i,j}$ are more likely to respond at a lower level of stimulus s_j.

The fixed threshold model assumes that agent's thresholds are fixed over time and so their level of specialization for a given task does not change in time. As a consequence, this model does not take into account the plasticity characteristic of the colony because it assumes that individuals are differentiated by their preassigned threshold values. Therefore, it is valid only over sufficiently short time scales, where thresholds can be considered constant. Finally, the model is not consistent with experiments with honey bees [19, 6], showing that aging and/or learning play a role in task allocation.

The dynamic threshold model In order to take these limitations into account, Theraulaz *et al.* [22] extended the threshold model by allowing thresholds to *vary in time* following a simple *reinforcement* process: the threshold associated to a task decreases when the corresponding task is performed, and increases otherwise.

Let ξ be the coefficient that describes *learning* (specialization) and φ be the coefficient that describes *forgetting* characteristics of an insect colony. If we consider a time-incremental model, individual i specializes when performing task j during a time period of duration Δt by changing its threshold as follows:

$$\theta_{i,j} \to \theta_{i,j} - \xi \Delta t, \tag{5}$$

and it forgets if it does not perform task j during a time period of duration Δt by by changing its threshold as follows:

$$\theta_{i,j} \to \theta_{i,j} + \varphi \Delta t. \tag{6}$$

Now, let $x_{i,j}$ be the fraction of time spent by individual i in task j: within Δt, individual i performs task j during time $x_{i,j}\Delta t$, and other tasks during $(1 - x_{i,j})\Delta t$. Let ξ and φ be identical

for all tasks, and the dynamics of $\theta_{i,j}$ be restricted to an interval $[\theta_{min}, \theta_{max}]$. Therefore, threshold values vary continuously in time according to:

$$\frac{\partial \theta_{i,j}}{\partial t} = [(1 - x_{i,j})\varphi - x_{i,j}\xi]\Theta(\theta_{i,j} - \theta_{min})\Theta(\theta_{max} - \theta_{i,j}), \tag{7}$$

where $\Theta(\cdot)$ is the Heaviside function with $\Theta(y) = 0$ if $y < 0$, $\Theta(y) = 1$ if $y \geq 0$.

The probability of an individual i to be engaged in task j is still described by equation 4, but now $\theta_{i,j}$ is dynamic in time and therefore:

$$P(s_j(t), \theta_{i,j}(t)) = \frac{s_j^2(t)}{s_j^2(t) + \theta_{i,j}^2(t)}. \tag{8}$$

As explained above, $x_{i,j}$ is the fraction of time spent by individual i in performing task j. Its value increases according to the threshold $\theta_{i,j}$ of agent i for task j. $x_{i,j}$ varies in time according to:

$$\frac{\partial x_{i,j}}{\partial t} = P(s_j, \theta_{i,j})(1 - \sum_{k=1}^{m} x_{i,k}) - p x_{i,j} + \Psi(i, j, t), \ \forall k \neq j, \tag{9}$$

where $1 - \sum_{k=1}^{m} x_{i,k}$ is the fraction of time potentially available for performing tasks. The term $p x_{i,j}$ expresses that an active individual gives up task performance and becomes inactive with probability p per unit time (identical for all workers and all tasks). The average time spent by an individual in task j performance before giving up this task is $1/p$. $\Psi(i, j, t)$ is a centered Gaussian stochastic process with variance σ^2, which is uncorrelated in time, among individuals and among tasks.

For simplicity, the stimulus dynamics for a task j increases at a fixed rate per unit time if no agent bids for it. The differential equation that describes the variation of s_j in time is

$$\frac{\partial s_j}{\partial t} = \delta - \frac{\alpha}{N}(\sum_{i=1}^{N} x_{i,j}), \tag{10}$$

where δ is the increase in stimulus intensity per unit time, and α is a scale factor measuring the efficiency of task performance. It is assumed that both factors are identical for all tasks, and that α is fixed and identical for all individuals.

In reality, however, α could vary as a result of specialization. The amount of work performed by active individuals is again scaled by the number of individuals N.

3.2.1 Campos' Algorithm

Campos' *et al.* [7] solved the DTA_{Hom} problem with an algorithm based on the dynamic threshold model as shown in the previous section. They developed this algorithm to show similarities between the *insect-based* approach and the *market-based* approach. We call their algorithm *Ant-Based Approach* (*ABA*). In this approach agents are in charge of booths and autonomously bid to paint trucks. In Campos' model, a stimulus s_{c_j} is associated to the color c_j of a truck j. Each agent k has a threshold value θ_{k,c_j} for each available color. The stimulus s_{c_j} for each color is given by the sum of the stimuli of the unassigned tasks for each particular color. The equation that describes the stimulus demand is:

$$s_{c_j} = \sum_i s_i \delta(c_i - c_j), \tag{11}$$

where c_i is the color of truck i waiting in the storage and c_j is the color of the considered truck j. $\delta(\cdot)$ is the Dirac function and the sum is taken over all trucks.

The probability of the booth k to get engaged in task j in given by:

$$P(s_{c_j}, \theta_{k,c_j}) = \frac{s_{c_j}^2}{s_{c_j}^2 + \alpha\theta_{k,c_j}^2 + \Delta T^{2\beta}}, \tag{12}$$

where c_j is the color of truck j and θ_{k,c_j} is the threshold of agent (booth) k for color c_j. α and β are parameters and ΔT is the waiting time necessary before the truck starts to be painted by that booth. ΔT is computed like in equation 3. If there are no bidders because all the queues are full the truck stays in the storage. For this algorithm, values of $P(s_{c_j}, \theta_{k,c_j})$ for different agents are compared and the tasks is assigned to the booth with the highest value.

Only when a truck j is assigned to booth k, the threshold values are updated for all booths. θ_{k,c_j} decreases by an amount ξ:

$$\theta_{k,c_j} \rightarrow \theta_{k,c_j} - \xi, \tag{13}$$

and the thresholds θ_{m,c_j} of all other paint booths for color c_j increase by an amount φ

$$\theta_{m,c_j} \rightarrow \theta_{m,c_j} + \varphi, \ \forall m \neq k. \tag{14}$$

Variations of θ_{*,c_j} take place within the bounds θ_{min} and θ_{max}. Equation 13 expresses the fact that booth k tends to specialize on color c_j because it increases its probability of responding to a truck with color c_j by decreasing its response threshold θ_{k,c_j}.

3.2.2 Cicirello's Algorithm

Cicirello *et al.* [8] proposed a similar algorithm to solve the DTA_{Hom} problem called *R-WASP* that is also inspired by the dynamic threshold model.

The algorithm incorporates aspects which have been ignored by Campos. In particular, in Cicirello *et al.*'s algorithm, each task j in the system sends to all agents a stimulus s_j that is equal to the length of time the task is already waiting to be assigned to an agent and does not depend on its color.

Campos' algorithm uses the probability function to determine the booth to assign the truck. Cicirello *et al.*'s model instead uses $P(s_j, \theta_{k,c_j})$ as a *pre-disposition* to respond to a stimulus. Therefore, $P(s_j, \theta_{k,c_j})$ represents the probability for a booth to bid for a task. In the algorithm of Cicirello *et al.*, the pre-disposition to respond to a stimulus is given by:

$$P(s_j, \theta_{k,c_j}) = \frac{s_j^2}{s_j^2 + \theta_{k,c_j}^2}, \tag{15}$$

where c_j represents the color or the type of task j.

Furthermore, in Campos' approach, *thresholds* are updated only when a truck is assigned to a paint booth and that rule involves only thresholds of the respective color. In Cicirello *et al.*'s approach, each agent k, at each time step, updates its own thresholds $\theta_{k,*}$ according to the following rules. At each time step, if the agent k is processing or setting up for a task j of color c_j the agents' thresholds are updated according to:

$$\theta_{k,c_j} \rightarrow \theta_{k,c_j} - \xi, \tag{16}$$

$$\theta_{k,c_i} \rightarrow \theta_{k,c_i} + \varphi, \ \forall i \neq j. \tag{17}$$

Otherwise, if the agent k is currently not processing any task, a third update rule is used for each threshold:

$$\theta_{k,c_i} \rightarrow \theta_{k,c_i} - \delta^t, \ \forall i, \tag{18}$$

where t represents the number of time steps in which the agent is already idle. The rule given in Equation 18 increases the value of $P(\theta_{k,c_j}, s_j)$ to encourage an idle agent k to take whatever tasks it can get rather than remaining idle.

When two or more agents of the colony want the same task, they interact with each other in a *dominance contest*. If this interaction takes place, the agent with the higher social rank has a higher probability of dominating in the challenge. Through such interactions, insects within the colony self-organize into a *dominance hierarchy*. The dominance contest rules are applied if two or more agents respond positively to the same stimulus. In this case there is a kind of challenge

between bidders and the winner is assigned the task. The idea is that each participant k has a force value F_k:

$$F_k = 1 + T_{proc} + T_{setup}, \tag{19}$$

where T_{proc} is the sum of the processing times t_{proc} for all the tasks in the queue of booth k and T_{setup} is the sum of their setup times t_{setup}. For a booth k, a lower force value F_k corresponds to a shorter queue and leads to a higher probability to win in a dominance contest. The rule to determine the probability for agent k to win against all the others competitors, is:

$$P(F_1, ..., F_n) = \frac{\sum_{i \neq k}^{n} F_i^2}{(n-1)\sum_{i=1}^{n} F_i^2}. \tag{20}$$

If more than one agent competes for a given task, a single dominance contest is used to determine the winner.

4 *ATA*: An improved algorithm for the homogeneous case

The agent-based algorithm proposed by Cicirello *et al.* shows a good level of specialization of the paint booths. This characteristic is desirable because it avoids unnecessary reconfigurations. The way the solution is achieved is completely distributed and no global information is required. We used the algorithm of Cicirello *et al.* initializing all working agents with the same threshold for all types of tasks.

In general, we observed that during the experiment the agents become specialized in one type of task and the performances of this algorithm are better than these obtained by one of the non-adaptive algorithms. However, we have observed that *R-WASP* requires some amount of time to initially adapt to the product mix as well as to dynamically re-adapt to a changing product demand. For example, Cicirello *et al.* consider in their experiments a case with two types of task, an asymmetric probability distribution and a product mix that changes at the half of the simulation. They show that the adaptation process to the new probability mix takes more than one third of the experimental time.

In order to overcome this problem we propose three modifications on the original algorithm and one additional rule to speed up the adaptation process. We call this improved algorithm *Ant Task Allocation* (*ATA*) and specify it in the following two sections. Afterwards, in Section 4.3 we show the plasticity of *ATA* in a particularly dynamic environment by discussing an example.

4.1 Modifications of Existing Rules

Threshold Update Rules (TUR): The update rules proposed by Cicirello *et al.* depend on the type of the task (the color c_j) which is currently used by a booth. We remark that an agent may have a queue of several tasks behind itself. These tasks in the queue are not necessarily of a same type. For example, an agent might be processing a task of type 1 and have only one task in its queue that is of type 2 like in Figure 2. This means that, as long as the task of type 1 is not finished, the corresponding threshold for 1 is decreased and the threshold value for 2 is increased. If the agent is offered two tasks, one of type 1 and one of type 2, the probability to take the task of type 1 is higher than the probability to pick task 2. This is not desirable because it can cause additional setups. We have chosen to modify the update rules by letting the last task in an agent's queue determine which threshold values are updated in order to reduce the number of necessary setup and consequently, the makespan.

Calculation of the Force Variable (CFV): The dominance contest introduced by Cicirello *et al.* tries to find a good solution to choose among several booths competing for a same task. The force value is defined as the sum of the process times and setup times of the tasks waiting in the queue of a booth. Figure 3 shows a situation where two agents with the same queue length bid for the same task. In the algorithm of Cicirello *et al* the probability to win in the dominance contest is

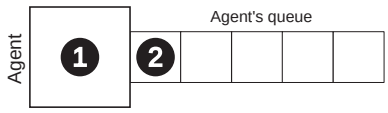


Figure 2: This agent is currently processing a task of type 1 and in its queue there is a waiting task of type 2.

the same. This does not take into account a possibly required setup for the task they are bidding for. We modify the force value according

$$F_k(j) = 1 + T_{proc} + T_{setup} + t_{setup,j}, \tag{21}$$

where $t_{setup,j}$ represents the setup time between the last task in the booth's queue and the current truck j.

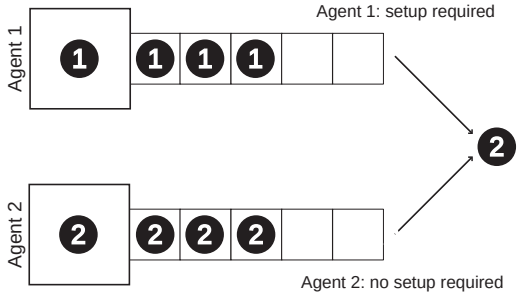


Figure 3: Two agents with the same queue length that bid for the same task.

DOminance Contest (DOC): Using the dominance contest rule as specified in Equation 20, the more machines compete with each other in a dominance contest, the smaller are the differences between the probabilities to win. In general the probability for one competitor to win a dominance contest with n competitors is never higher than $\frac{1}{n-1}$. In order to avoid this, we introduce this new rule:

$$P_k(F_1, ..., F_n) = \frac{\frac{1}{F_k^2}}{\sum_{i \neq k}^{n} \frac{1}{F_i^2}}. \tag{22}$$

4.2 Additional Rule to Optimize the Threshold Values

Idle Machine does not Bid (IMB): Equation 18 offers a threshold update rule for idle agents in order to encourage them to bid for tasks of any type. This update rule decreases all threshold values by a value that exponentially increases in time. The idea is that a machine may stay idle for same time rather than being forced to take any task immediately. We observed that this can cause agents to stay idle for very long and therefore has a negative effect on several performance measures. We propose a new update rule, which is employed in case an idle agent refuses to bid for a task it is offered:

$$\theta_{k,c_j} \rightarrow \theta_{k,c_j} - \gamma. \tag{23}$$

In this case, the corresponding threshold θ_{k,c_j} of the refused task of type c_j is decreased by the fixed value γ. Variations of θ_{k,c_j} takes place within the bounds θ_{min} and θ_{max}.

4.3 Example

In this section we present an example that shows the plasticity of *ATA* to the dynamic changes on the DTA_{Hom} problem environment. In the paper of Cicirello *et al.* a very similar situation is presented where the adaptation process to the environmental changes takes more than one third of the experimental time. We will show using *ATA* that this process is very quick. We think that the proposed improvement rules are essential to achieve this behavior.

In order to show this, we consider a simple problem instance with 4 identical agents. Each agent has a queue size of 5 trucks. To paint a truck a booth requires 5 time steps. Another 10 time steps are needed for a setup. Trucks exit the assembly line during 420 time steps. If one step is equal to a minute, trucks enter in the system during seven hours. The number of trucks to be painted is in average equal to 336. This number is chosen because it is in the given time the maximum number of trucks that 4 booths can paint considering the process time, no setups and no idle states. The number of trucks that exit the assembly line is equally distributed during the 420 time steps. In this example trucks can be painted in two colors only: type 1 and type 2. In order to analyze the plasticity capability of the system, we chose a particular color probability mix: for the first 210 time steps the tasks of type 1 appear three times more frequently than those of type 2. Afterwards, these probabilities swap so that the type of task that appeared more frequently appears more rarely. We refer to this probability mix as CHANGE. The parameter set used for *ATA* is the same as those used in Section 6.1.3 for the DTA_{Hom} problem instances.

For this example we expect that for the first half of the experiment a higher number of agents becomes specialized in task 1 and that this swaps for the second half. Figure 4 shows a typical *ATA* behavior on this configuration. At the beginning agents that are in the same starting conditions become specialized and start performing the tasks. From the time step 210 the demand of type 2 tasks becomes high and the relative stimulus increases (Figure 4(a)). At the same time, agents working on type 1 tasks wait for more tasks of the same type but none becomes available. Initially these agents do not bid for type 2 tasks. Until here the behavior of the agents of *ATA* is similar to the agents of *R-WASP*. Then, in *ATA* the *IMB* rule takes effect and decreases the threshold of each agent that prefers to rest idle and not bid for type 2 tasks. The effect of the stimulus increment and of the *IMB* rule is that two agents become specialized for type 2 tasks (Figure 4(d) and (e)). The others agents do not change their specialization level. This behavior do not change until the end of the experiment.

5 Extension of the Algorithms to the heterogeneous case

The *DTA* problem definition considers agents that can have different processing times for different task types. The insect-based algorithms presented give satisfying results on the homogeneous case of the *DTA* problem but if they are applied on the heterogeneous case the performances are not so acceptable. The problem is that agents bid for a task considering only their specialization level, without taking into account their own characteristics, that is, their capacity to process a task quickly or slowly. In order to have better results, we extend the algorithms so that the faster an agent can perform a task, the higher is the probability to bid for it. In Section 5.1 we present this extension that is applied to the presented insect-based algorithms and is inspired by the caste subdivision of labor observed in social insects. Afterwards, in Section 5.2 we show its impact by discussing an example.

5.1 Different Process Speed (DPS) Rule

In an insect colony different castes of agents have a different response to a considered task. Wilson [25] observed a division of labor between insect casts studying ant species *Pheidole*. In normal conditions some castes have higher probability of obtaining a task than others. If in the environment there is a high demand to perform a particular task or a subset of castes is eliminated from the environment, the behavior of the ants in the system changes so that they start to perform tasks for which they are not predisposed. To reproduce this behavior in the considered insect-based

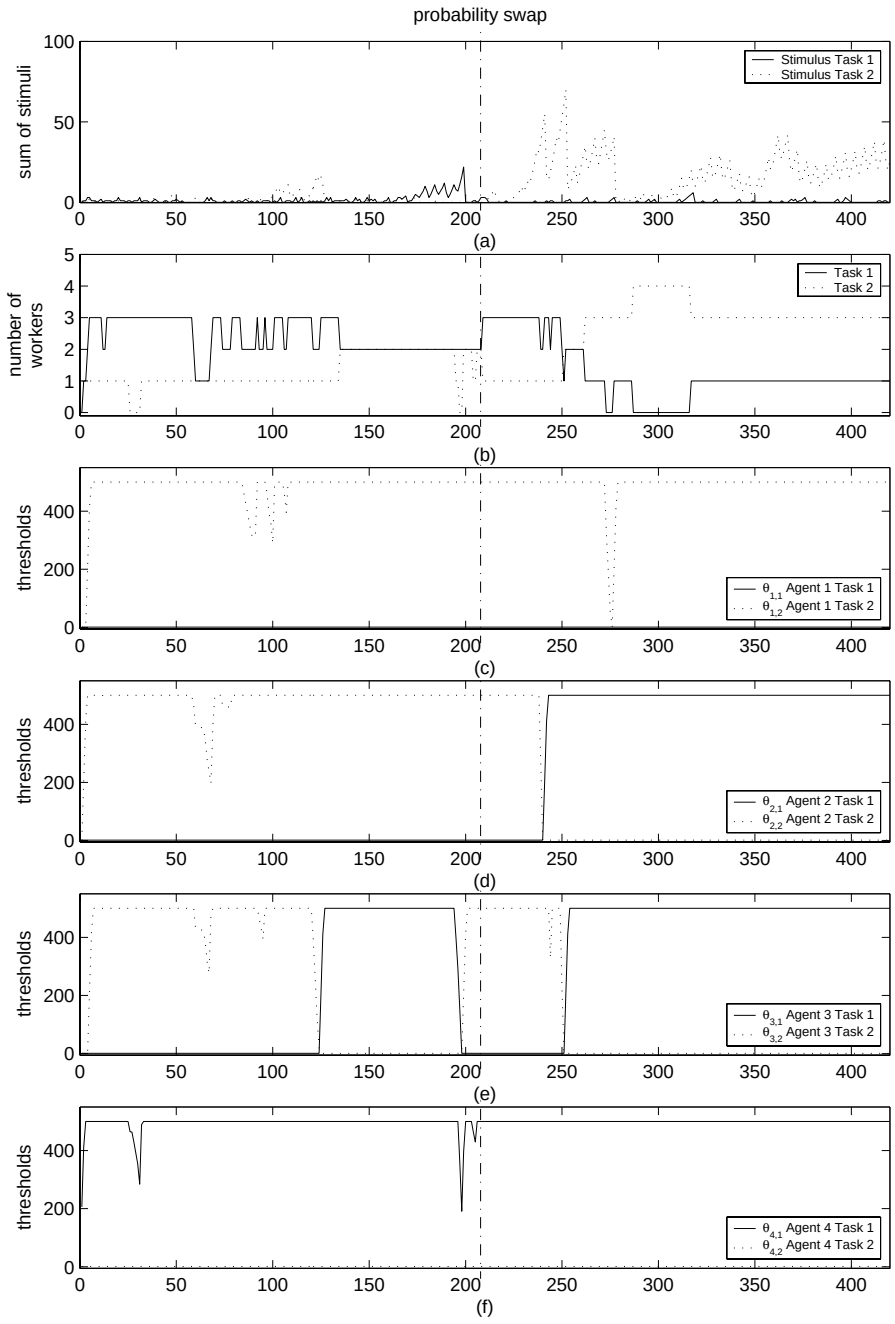


Figure 4: The graphs show the sums of the tasks stimuli, the number of working agents for each type of task and the dynamic of thresholds of each agent. Graphs (c), (d) and (e) show that for the first half of the experiment agents 1, 2 and partially 3 are specialized on tasks of type 1 (low thresholds). This behavior is confirmed by graph (b). Graph (f) shows that agent 4 becomes specialized on tasks of type 2. At time 210 the probability mix reverse. Agents still remain specialized to the previous mix. The stimulus of type 2 tasks increases (a) and the agents remain idle. At time 250 (graph (d) and (e)) agents 2 and 3 adapt themselves to the new environment. From this moment to the end of the experiment there are three agents working on tasks of type 2 and one on tasks of type 1 (b).

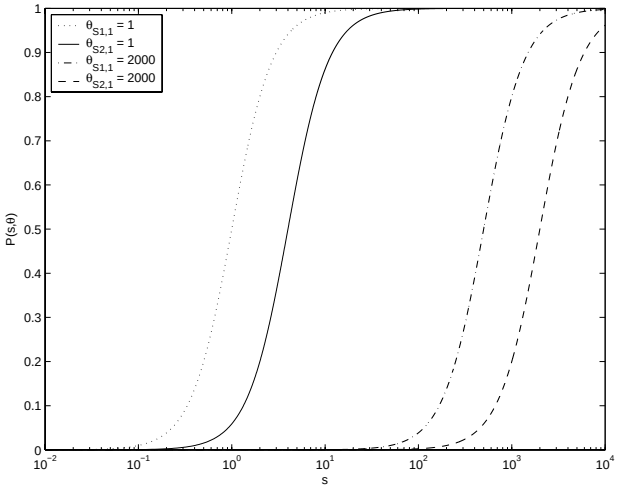


Figure 5: The threshold response function that one agent of subset S1 and one agent of subset S2 have for a task of type 1 with different specialization levels.

algorithms, we introduce the process speed that an agent k has for a task j of type c_j in the probability function $P(s_j, \theta_{k,c_j})$. The most promising probability function is obtained modifying Equation 15 as follow:

$$P(s_j, \theta_{k,c_j}) = \frac{s_j^2}{s_j^2 + \theta_{k,c_j}^2 * (t_{proc,k,c_j} - tmin_{proc,c_j} + 1)}, \tag{24}$$

where $t_{proc,k,j}$ is the processing time that agent k has for task j and $tmin_{proc,c_j}$ is the minimum processing time in the system for the type of task c_j. The added term performs like a weight to the specialization level during the bidding process.

We study the probability function in a simple instance of the *DTA* problem where there are two sets of working agents with two booths for each set and two types of task. The process time is given in Table 1(a) whereas the relative probability function $P(s_j, \theta_{k,c_j})$ is presented in Figure 5. The figure shows that agents of subset S1 always have a higher probability to bid for tasks of type 1 than agents of subset S2. An important characteristic of this function is that the distance between the response to a stimulus of an agent k of the first subset and the same response by an agent of the second set is constant and independent from the specialization value θ_{k,c_j}. This improvement allows a correct distribution of work between agents with a behavior that still maintains a high plasticity level to the environmental changes. When an insect-based algorithm uses the *DPS* rule, we refer to it with a letter c at the end of the respective name (*ABAc*, *R-WASPc*, *ATAc*). Finally, we observe that, if we use these extended algorithms on a DTA_{Hom} problem instance, they perform as the original ones.

5.2 Example

The aim of this example is to analyze the division of labor among agents of a *DTA* instance using *ATAc* algorithm. We consider a situation where there is an asymmetric demand of types of tasks. The considered instance has 2 subsets of working agents (S1 and S2) with 2 booths within each subset. Trucks can be painted in two colors: type 1 and type 2. Each agent has a queue size of 5 trucks. 10 time steps are needed for a reconfiguration. Trucks exit the assembly line during 420 minutes. Process times are described in Table 1(a). The number of trucks to be painted is on average equal to 446. This number is the maximum number of trucks paintable in 420 minutes by the system considering that three booths work for one type of task, the fourth one works on

Table 1: Values for a *DTA* problem instance with two subsets of working agents, two agents for each subset and two different types of tasks.

(a) Process time matrix.

	Type 1	Type 2
subset S1	3	9
subset S2	9	3

(b) Number of worked tasks.

	Type 1	Type 2
subset S1	278	0
subset S2	30	138

the other type of task and no setups and idle times are necessary. The color probability mix is asymmetric with a higher number of type 1 tasks: a truck of type 1 has probability 0.77619 to appear at each time step and a truck of type 2 has probability 0.332. We refer to this probability mix as DIFF. The parameter set used by *ATAc* is the same as that used in Section 6.1.3 for the *DTA* problem instances.

Before running the example we expect this behavior:

- In a normal condition tasks are executed by agents that need short time to process them.
- If there is a high demand of a type of tasks and there are no more agents that perform it quickly, agents that perform it slowly start working for it.
- If an agent that performs a type of task quickly and an other agent that performs the same type slowly are both specialized for that type, the quicker agent should have a higher probability to process it.

Figure 6 shows a typical *ATAc* behavior on this particular *DTA* problem instance. At the beginning of the experiment agent 1 and 2 within subset S1 start to perform type 1 tasks and become specialized on it (Figure 6(b), times 0-150). Instead, agent 3 and 4 within subset S2 that are able to perform type 2 tasks quickly, become specialized on that type (Figure 6(c)).

The asymmetric probability mix at time 100 increases the demand of type 1 tasks (Figure 6(a)) but in the system all the agents of subset S1 are busy so at time 150 the high stimulus of type 1 leads agent 4 of the subset S2 to work on type 1 tasks (Figure 6(g), time 150). From time step 150 agent 4 becomes specialized in tasks of type 1. In fact, its threshold for that type of task is close to 0 (Figure 6(g), time 150-end). Agent 3 still works on tasks of type 2. In fact, its threshold value for that type of task is close to 0 (Figure 6(f), time 0-end). Table 1(b) represents the number of worked tasks at the end of the experiment and confirms that type 1 tasks are performed principally by subset S1 and type 2 tasks by subset S2, that is, agents that are able to perform tasks quickly, using the DPS rule, really process them.

6 Empirical Analysis

Previous works on the homogeneous case of the *DTA* problem [7, 8, 15] analyze few and sometimes simple instances of the problem where it is difficult to show that an algorithm performs better than another one. It is possible that a simple modification of the instance leads to a drastic performance decrease. Moreover, little or no analysis has be done to show the contribution of the proposed rules.

The aim of this section is to compare all the presented algorithms on the *DTA* problem and analyze the influence of each presented rule on the makespan.

We have introduced two type of rules with the objectives to:

1. improve the algorithm of Cicirello *et al.* to particularly dynamic DTA_{Hom} problem instances.
2. extend all the insect-based algorithms with the DPS rule to improve the performance on *DTA* problem instances.

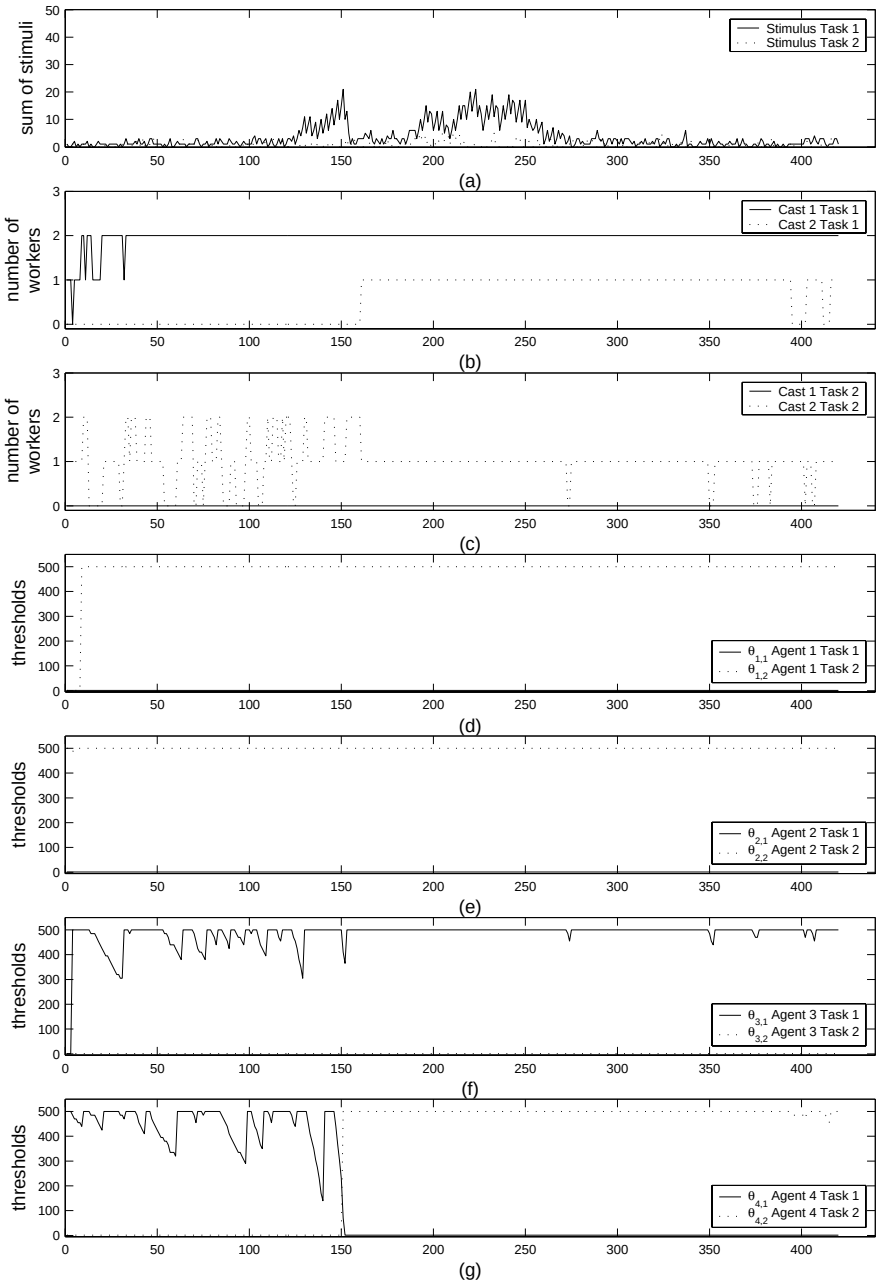


Figure 6: The graphs show the tasks stimuli, the number of working agents for each type of task and the dynamics of the threshold values of each agent. At the beginning of the experiment only agents 1 and 2 work on type 1 tasks (Figure (b)), and only agents 3 and 4 on type 2 (Figure (c)). Type 1 stimuli increases from time 100 to 150 (Figure (a)) until when the agent 4 (Figure (g)) starts to work on this type. From time 150 to the end of the experiment type 1 stimuli become low (Figure (a)). From this time on, all the agents of the subset S1 (Figure (b)) and one agent of the subset S2 work for tasks of type 1. Agent 3 is enough to satisfy the low demand of type 2 tasks.

Table 2: Summary of the classes of instances of the DTA and DTA_{Hom} problem considered in the experiments.

	DTA_{Hom} Class	DTA Class
Subsets of agents	1	2
Max. number of agents	24	12
Broken probability	0.02	0.02
Queue length	10	5
Process Time	5	3-9
Setup Time	10	10
Simulation Time	420	420
Number of Tasks	2016	840
Avg. types of tasks	12	10
Types mixes	1)$P(1..n/4) = 3P(n + 1/4..n)$	1)$P(1..n/2) = 3P(n + 1/2..n)$
	2)like 1, switch after 210 steps	2) like 1, switch after 210 steps
Probability between mixes	0.5	0.5

According to this, we present two experiments: one that uses a class of instances of the homogeneous problem and another one that uses a class of instances of the heterogeneous problem.

In the following section we describe the instance generators, a summary of the algorithms we use in the experiments and the methodology to tune their parameters. Afterwards, Section 6.2 describes the obtained results.

6.1 Setup

A lot work has been done to make a strict configuration and evaluation of the experiments. First of all in our experiments we consider *classes of instances* of the problem. A class contains problem instances that differ by variables such as the number of working agents, the task types and the color probability mix. In this section we define two problem classes in order to describe two real-world painting environments. Afterwards, for each considered algorithm we present the best combination of the parameter values tuned by an Evolutionary Algorithm (EA). Moreover, we summarize all the algorithms we use in the experiments and we introduce a simple, non-adaptive algorithm in order to emphasize during the results analysis the differences between adaptive and non-adaptive systems.

6.1.1 Instance Generators

An instance generator is a tool able to generate various instances from a class of the problem. We consider two instance generators able to obtain two classes of instances of the DTA problem. While the first one creates instances of the homogeneous problem, the second one creates instances of the heterogeneous problem. The first class of instances contains all instances with identical agents. On this class we are able to compare all the presented algorithms and to study the performance of each rule introduced in ATA. The second class of instances contains two subsets of agents, the first one able to perform quickly half of the types of tasks, while the second one has the opposite behavior. Applying on this class the original algorithms for the DTA_{Hom} problem and the insect-based algorithms with the DPS rule, we are able to compare the performance of each algorithm and to understand the contribution of the introduced rule. Table 2 summarizes the configurations of the two considered classes of instances. Instances within a same class may differ by the number of agents, the number of task types and the type of the probability mix of the tasks. In the following we describe each class separately.

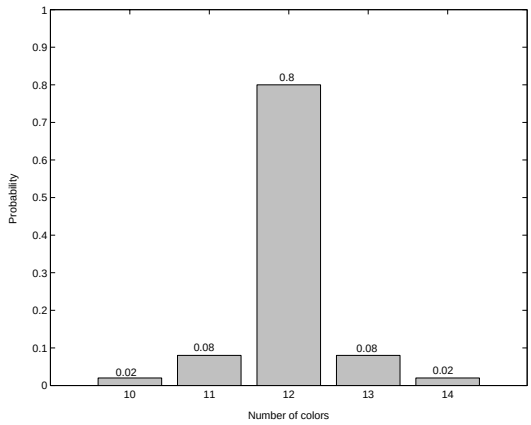


Figure 7: Probability distribution of the number of colors for the DTA_{Hom} experiment.

DTA_{Hom} **Class** This class contains all instances of the homogeneous problem where the agents are identical. In this class we try to simulate a typical working day of a big painting factory. According to this, the experimental time is equal to 420 minutes, that is, 7 working hours. The agents are 24 identical painting booths. In order to simulate the possibility to have some booths broken we introduce a variability on the number of maximum booths each instance has. We refer this variability to as broken probability. The broken probability for this class of the problem is equal to 0.02. Here the agents have a queue size equal to 10. The process time of each booth is equal to 5 minutes, 10 minutes are needed for a reconfiguration.

The number of trucks that exit the assembly line is equal to 2016 and is independent of the number of agents. This number is chosen because it is the maximum number of trucks that 24 booths can paint considering the process time, no setups and no idle time.

For this class we extract the number of colors from the probability distribution presented in Figure 7. Color types are assigned to the trucks according to the two mixes:

1. A part of the n colors have an higher probability than the others:

$$\begin{cases} \sum_{i=1}^{\lfloor n/4 \rfloor} P(i) = 3 \sum_{j=\lfloor n/4 \rfloor + 1}^{n} P(j) \\ P(1) = \cdots = P(\lfloor n/4 \rfloor) \\ P(\lfloor n/4 \rfloor + 1) = \cdots = P(n) \end{cases}$$

2. like the previous one, but after 210 minutes types that appear more frequently appear more rarely and vice versa.

Each instance uses with the same probability one mix or the other.

DTA **Class** This class contains instances of the heterogeneous problem where the agents differ by their processing speed. In this class we try to simulate a typical working day of a medium painting factory. Trucks exit from the assembly line for 420 minutes. We consider 12 painting booths with a probability of being broken that is equal to 0.02. Each agent has a queue size equal to 5. In this case there are two subsets of agents: one needs 3 minutes to process the first half of the available types of tasks and 9 minutes for the other half instead the other subset has the opposite behavior. For both the subsets the setup time is equal to 10 minutes.

The number of trucks exiting the assembly line is always equal to 840, which is the maximum number of trucks that 12 booths can paint considering an average process time of 6 minutes, no setups and no idle time.

For this class we extract the number of colors from the probability distribution presented in Figure 8. Color types are assigned to the trucks according to the two mixes:

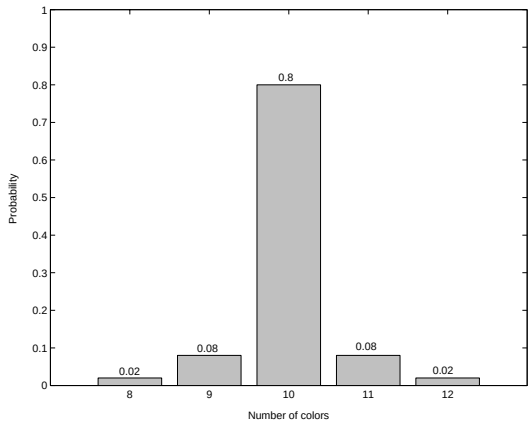


Figure 8: Probability distribution of the number of colors for the *DTA* experiment.

Table 3: Summary of the algorithms applied in both the experiments.

DTA_{Hom} Class	*DTA* Class	
MBA	*MBA*	-
ABA	*ABA*	*ABAc*
LOCUST	*LOCUST*	-
R-WASP	*R-WASP*	*R-WASPc*
ATA	*ATA*	*ATAc*

1. A part of the n colors have an higher probability than the others:

$$\begin{cases} \sum_{i=1}^{\lfloor n/2 \rfloor} P(i) = 3 \sum_{j=\lfloor n/2 \rfloor + 1}^{n} P(j) \\ P(1) = \cdots = P(\lfloor n/2 \rfloor) \\ P(\lfloor n/2 \rfloor + 1) = \cdots = P(n) \end{cases}$$

2. like the previous one, but after 210 minutes types that appear more frequently appear more rarely and vice versa.

Each instance uses with the same probability one mix or the other.

6.1.2 Algorithms Summary

Table 3 summarize all the algorithms we use during the empirical analysis. We consider a class of instances of the homogeneous (DTA_{Hom}) problem and a class of the heterogeneous (*DTA*) problem. We run on the DTA_{Hom} problem instances the market-based algorithm *MBA* of Morley described in Section 3.1.1, the insect-based algorithm *ABA* of Campos *et al.* described in Section 3.2.1, the insect-based algorithm *R-WASP* of Cicirello *et al.* described in Section 3.2.2 and our proposed algorithm *ATA* described in Section 4. For the *DTA* problem we run all the original algorithms and the three insect-based algorithms with the DPS rule.

In the following we present *LOCUST*, a dummy, non-adaptive algorithm used as a base-line for evaluationg the performances of the presented algorithms. In both the experiments we apply also this algorithm in order to have a comparison between adaptive and non-adaptive systems. *LOCUST* has functionalities inspired by the presented algorithms but it does not consider the specialization of each agent. It is described in the following.

The *LOCUST* algorithm: In this algorithm agents are independent units that autonomously bid to obtain tasks using a force value which is obtained by the sum of the time the agent needs to finish the working task plus the sum of processing times of each task in its queue and the current task plus the possible setup times between each queued task. At each time step agents bid to obtain tasks if their queue are not full. The agent with the lower force is assigned the task at the end of its queue. If two or more agents have the same force, the winner is chosen randomly. If no agents bid for a task, the task remains in the storage. For this simple algorithm no parameter tuning is necessary.

6.1.3 Parameters Tuning

The aim of the tuning is to find a good set of parameter values for each algorithm. We apply the tuning for all the algorithms considered in this paper because each one has some parameters to determine. In the *MBA* algorithm we tune the three constants in Equation 2. In the *ABA* algorithm we tune the constants α and β of Equation 12 and the two constants used in the update rules. In the *R-WASP* algorithm we tune the constants presented in the three update rules and in the *ATA* algorithm the same constants of *R-WASP* plus the one for the new *IMB* rule. For the insect-based algorithms we fix the range of the thresholds to $\theta_{min} = 1$ and $\theta_{max} = 500$. The initial threshold θ_{init} is fixed to 1.

Table 4 describes for each algorithm used in this experiment which parameters are tuned, the relative range and the values found. To tune the parameters of the algorithms we use an Evolutionary Algorithm (EA). The evaluation has been done as follows: we define a *cost function*, that in our case is the makespan to be minimized. Each population of the EA is composed by 20 individuals. An individual represents a set of values of the parameters to be tuned. Each individual is evaluated on 40 instances of the considered problem class and the average of the makespan values is its cost value. For each instance we execute a single run of the algorithm to be tuned. Individuals of a population are evaluated on the same 40 instances of the considered class. In this tuning we stop after 40 generations because normally after 15 generation the cost function becomes stable. Between generations the instances are changed to avoid using not representative instances of the class. Additionally, the tuning considers the elitism between generations so that the best 4 individuals of the previous generation are used in the new one and the other 16 individuals are generated using a self-adaptive $(\mu + \lambda)$ evolutionary strategy [21, 1]. If an algorithm is used in both the experiments we apply the parameters tuning twice.

Data show a kind of symmetry in the insect-based algorithms between the increment and the decrement of threshold update factors. It means that the agents become specialized during the experiment. Further information concerning the EA, the tuning logs, the generations values, the best individual of each generation and all the cost function values, are available at the paper web page with address `http://iridia.ulb.ac.be/~rghizzioli/dta/`.

6.2 Results

In this section we present and compare the results obtained by the algorithms in the two sets of experiments: the homogeneous case is presented in Section 6.2.1 and the heterogeneous case is presented in Section 6.2.2. The results are achieved running the algorithms of Table 3 on the classes of instances summarized in Table 2 using the parameters of Table 4 on 1000 problem instances. Our main performance measure is the makespan. For completeness and for a better understanding we also present the number of setups per agent and the storage dimension. For each experiment we show the contribution to the makespan given by each introduced rule.

6.2.1 DTA_{Hom} Class

The homogeneous case of the *DTA* problem considers a single set of identical agents. In this case agents should specialize on one type of task in order to minimize the required number of setups. A good distribution of the task types between the agents allows a low makespan. Results about

Table 4: Summary of the parameter values for all algorithms used in the empirical analysis for both experiments. In the first column there is the names of the algorithm and the names of the tuned parameters. The second column contains the range for each parameter. The third and the fourth column contain the values tuned by the Evolutionary Algorithm. If a value is equal to 0, this means that for that contest the respective rule is not important.

	Range	DTA_{Hom} Class	DTA Class
MBA			
P	(0-100)	46	11
C	(0-10000)	7200	5710
L	(0-5)	2.78	2.25
ABA			
ξ	(0-500)	475	215
φ	(0-500)	67.5	30
α	(0-100)	44.6	78
β	(0-50)	2	2.5
R-WASP			
ξ	(0-500)	345	310
φ	(0-500)	480	26
δ	(0-500)	490	0
ATA			
ξ	(0-500)	165	224
φ	(0-500)	205	115
δ	(0-500)	1.2	0
γ	(0-500)	34	0
ABAc			
ξ	(0-500)	-	430
φ	(0-500)	-	450
α	(0-100)	-	98
β	(0-50)	-	2.95
R-WASPc			
ξ	(0-500)	-	395
φ	(0-500)	-	6
δ	(0-500)	-	0.5
ATAc			
ξ	(0-500)	-	95
φ	(0-500)	-	65
δ	(0-500)	-	0
γ	(0-500)	-	25

Table 5: Summary of the makespan values for all the algorithms run on the class of instances of the DTA_{Hom} problem. The table shows the quantile values, the mean, the minimum and the maximum values achieved by each algorithm. The median values show that *ATA* performs better than the other algorithms. *ATA*, compared with the other algorithms, is also a stable algorithm because has smaller difference between the 1st and the 3rd quantile. For a graphical view see Figure 9. For a statistical analysis see Table 6.

	ATA	*R-WASP*	*ABA*	*LOCUST*	*MBA*
Min.	472.0	588.0	717.0	803.0	464.0
1st Qu.	497.0	749.0	843.8	944.0	484.0
Median	509.0	812.5	887.0	980.0	531.5
Mean	513.4	808.0	895.9	984.1	599.2
3rd Qu.	525.0	864.0	940.3	1017.0	706.0
Max.	699.0	1066.0	1156.0	1217.0	1042.0

Table 6: Paired Wilcoxon Rank Sum Test applied on the makespan values of the algorithms on the class of instances of the homogeneous problem. Values are obtained with a confidence level of 95%. All results given by this table assure that differences between algorithms on the median values of the makespan are significant.

	ABA	*R-WASP*	*LOCUST*	*MBA*
R-WASP	$< 2.2e^{-16}$	-	-	-
LOCUST	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-	-
MBA	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-
ATA	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$

the makespan are shown in Table 5 and in Figure 9. *ATA* achieves the best results because, as we showed, it has a high level of plasticity to the environmental changes. Furthermore, the 1000 makespan values produced by *ATA* are very similar and compared with the results obtained by the other algorithms it seems that *ATA* is not influenced by the variability between instances in the class. In the next paragraph we show the significance of each rule introduced in *ATA*. As can be observed, the algorithm of Campos *et al.* does not obtain good results for this class of instances of the DTA_{Hom} problem. It is also interesting to show a linear performance improvement in time of the insect-based algorithms. Moreover, the market-based algorithm *MBA* reaches good performance for this class of instances.

To be sure of the real significance of the obtained results we apply a Wilcoxon Paired Rank Sum Test as presented in Table 6. The values show that different results of the makespan between algorithms are significantly different with a confidence level of 95%.

Figure 10shows the box plot of the number of setups per agent and the storage dimension for each algorithm applied to the DTA_{Hom} class. These graphs have are similar to the makespan box plot of Figure 9. They reconfirm that *ATA* has the best performances also considering the number of setups and storage dimension.

The average number of setups of an agent for the *ATA* algorithm is 3.939 and, on average, 61.47 trucks wait in the storage.

The significance of the introduced rules for the DTA_{Hom} problem *ATA* is the solution we propose to solve the homogeneous case of the *DTA* problem. *ATA* is an extension of *R-WASP* with all the proposed rules *TUR*,*CFV*, *DOC* and *IMB* activated. Here we perform a statistical analysis to understand the contribution that each rule gives to the makespan.

First of all, as shown in Figure 11, we use a binary notation for each possible configuration of

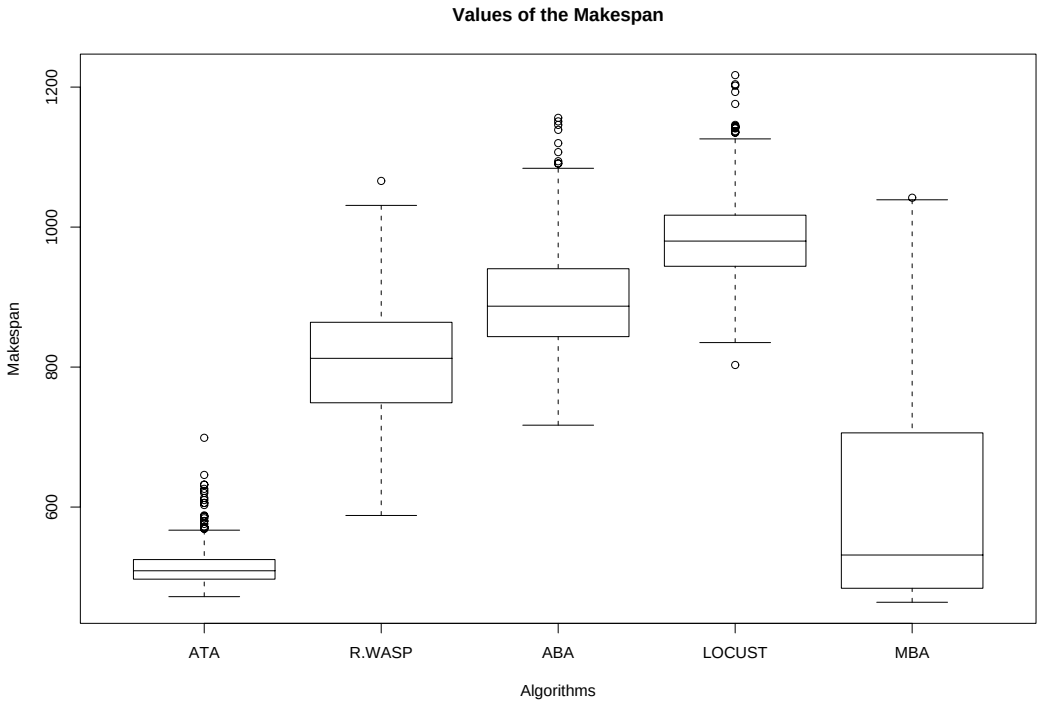
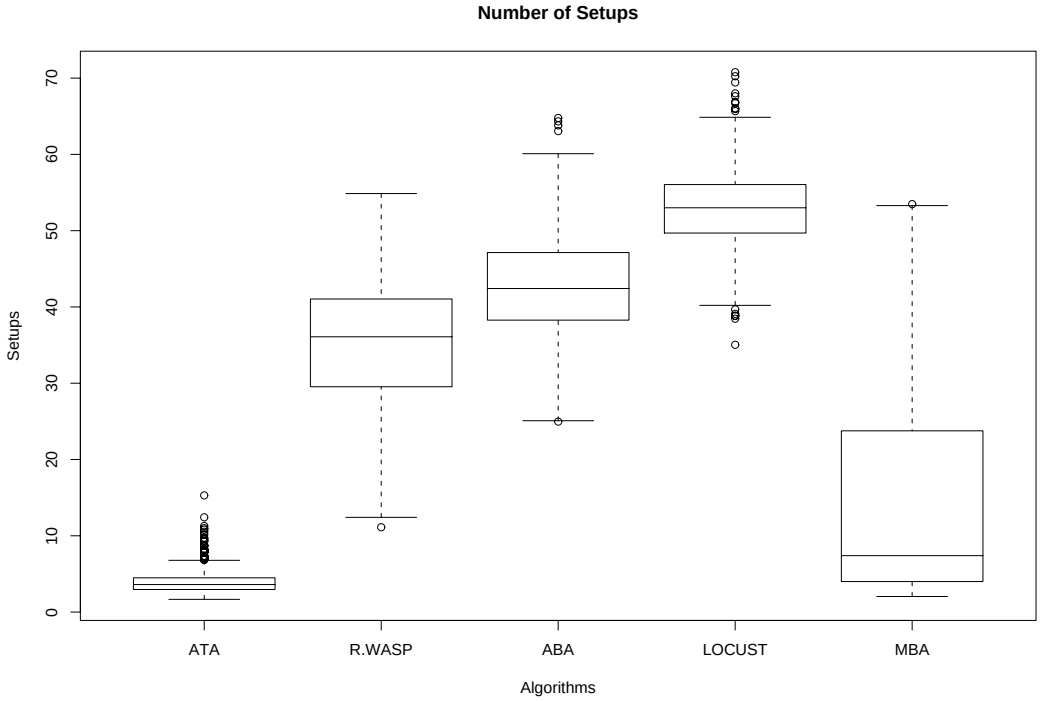


Figure 9: Box plot of the values and of the ranks of the makespan for all the algorithms run on the class of instances of the homogeneous problem. *ATA* is the algorithm that has the lower makespan and the smaller quantile representation area.



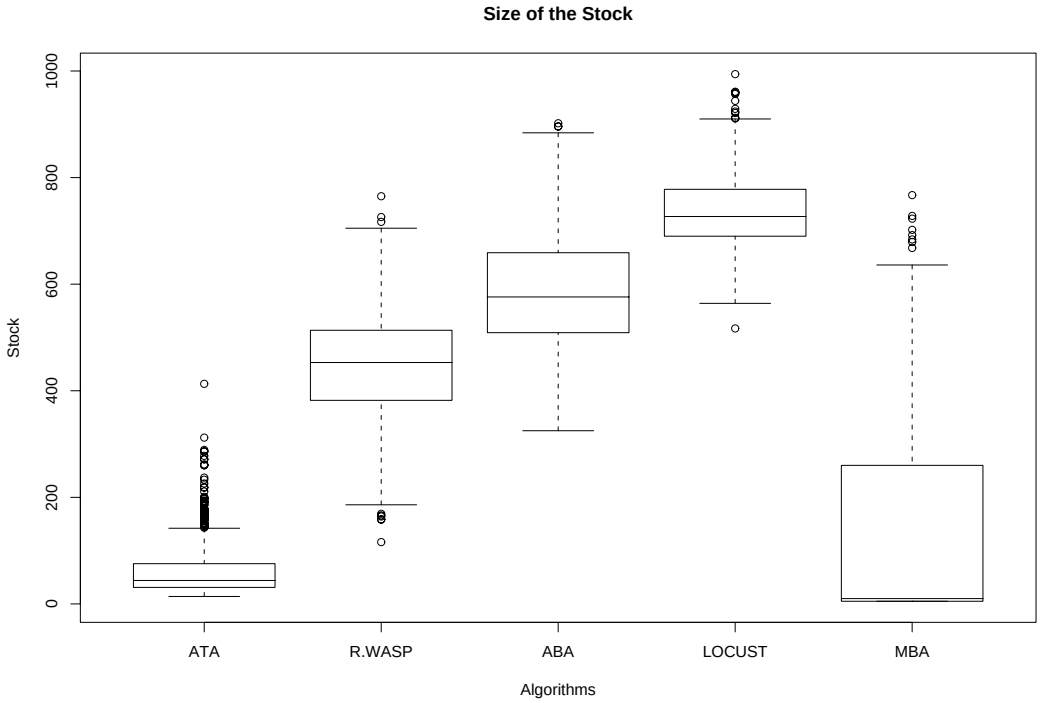


Figure 10: Number of setups per agent and storage dimension for the homogeneous experiment. Both figures have a form similar to Figure 9. Obviously, a low makespan implies a low number of setups and a low storage usage.

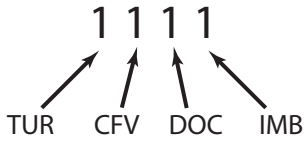


Figure 11: Coding system used to indentify the 16 configurations of the proposed algorithm.

Table 7: Analysis of variance. The asterisks indicates significance.

	Df	Sum Sq	Mean Sq	F-value	$Pr(>F)$	
TUR	1	4258.2	4258.2	60415.9493	¡ 2.2e-16	***
CFV	1	0.1	0.1	1.1285	0.2881	
DOC	1	31.3	31.3	444.1618	¡ 2.2e-16	***
IMB	1	107.5	107.5	1525.5993	¡ 2.2e-16	***
TUR:CFV	1	9.5	9.5	134.4396	¡ 2.2e-16	***
TUR:DOC	1	0.001543	0.001543	0.0219	0.8824	
CFV:DOC	1	3.4	3.4	48.7311	3.051e-12	***
TUR:IMB	1	10.9	10.9	155.3420	¡ 2.2e-16	***
CFV:IMB	1	3.7	3.7	51.7992	6.421e-13	***
DOC:IMB	1	2.4	2.4	34.5107	4.323e-09	***
TUR:CFV:DOC	1	29.5	29.5	418.5021	¡ 2.2e-16	***
TUR:CFV:IMB	1	3.6	3.6	50.5260	1.226e-12	***
TUR:DOC:IMB	1	2.0	2.0	28.4315	9.839e-08	***
CFV:DOC:IMB	1	16.1	16.1	228.8127	¡ 2.2e-16	***
TUR:CFV:DOC:IMB	1	2.1	2.1	30.1007	4.164e-08	***
Residuals	15984	1126.6	0.1			

ATA. For example 0000 indicates that no rules are activated, 1000 only *TUR*, 0100 only *CFV*, 0010 only *DOC*, 0001 only *IMB*, 0110 *CFV* and *DOC*, etc. Makespan data are collected by running all these 16 configurations on 1000 instances on the DTA_{Hom} problem after a parameter tuning of each of them. The evolutionary algorithm and the methodology used are described in Section 6.1.3. Parameters belonging to unused rules are not tuned.

A first analysis of these data is shown in Figure 12(a) and in Figure 12(b). Our results show that, on average, configuration 1011 performs better than all the others and that *ATA* is very close to it. Moreover, all possible configurations have a lower makespan than the solution of Cicirello *et al.* (configuration 0000). Figure 12(b) shows that results in the right part of the graph, with the *TUR* rule activated, are significativelly better than those in the left part of the graph where the rule is deactivated.

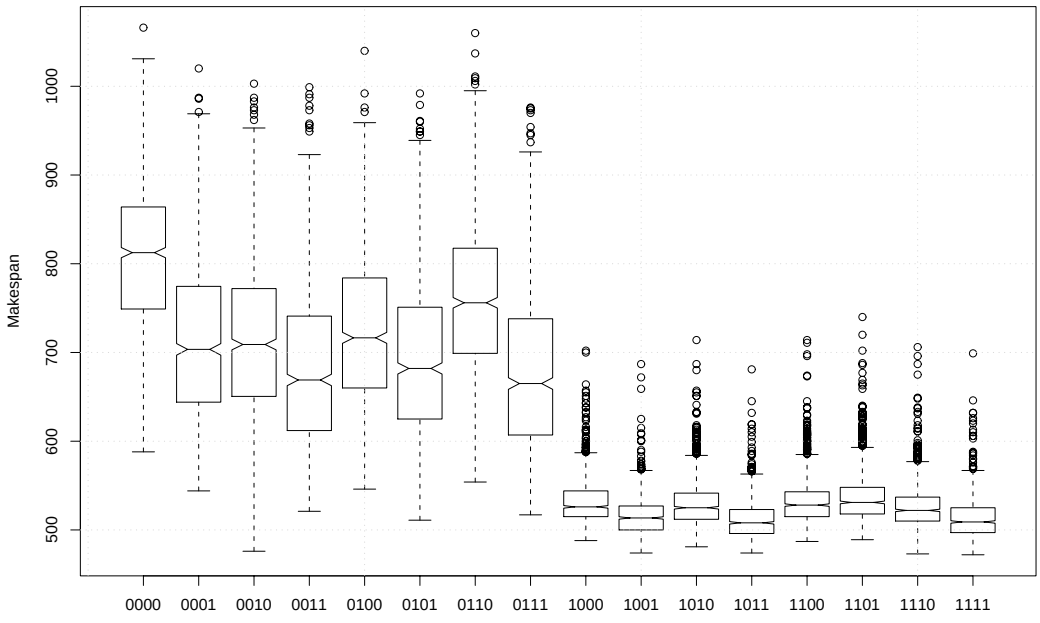
Figure 13 and Figure 14 show the contribution that each rule gives to the makespan averaged on all configurations where a considered rule is activated independently the other rules. We clearly observe that if *TUR* is activated there is a large difference on the makespan values. Instead, the *CFV* rule does not give any apparent contribution to the results. Finally, *DOC* and *IMB* give similar contribution to the makespan.

In order to formally assess the contribution of the different rules and their interactions, according to [23], we have considered an ANOVA analysis. As it can be observed in Figure 15, the residual of the makespan data does not meet the hypothesis of the ANOVA test, that is, linearity and heteroskedasticity. Therefore, the analysis was not conducted on makespan data but on the following transformation: $x' = log(420 - x)$ where x represents a single makespan result. Figure 16 shows that after transforming the makespan values the residuals are reasonably fine.

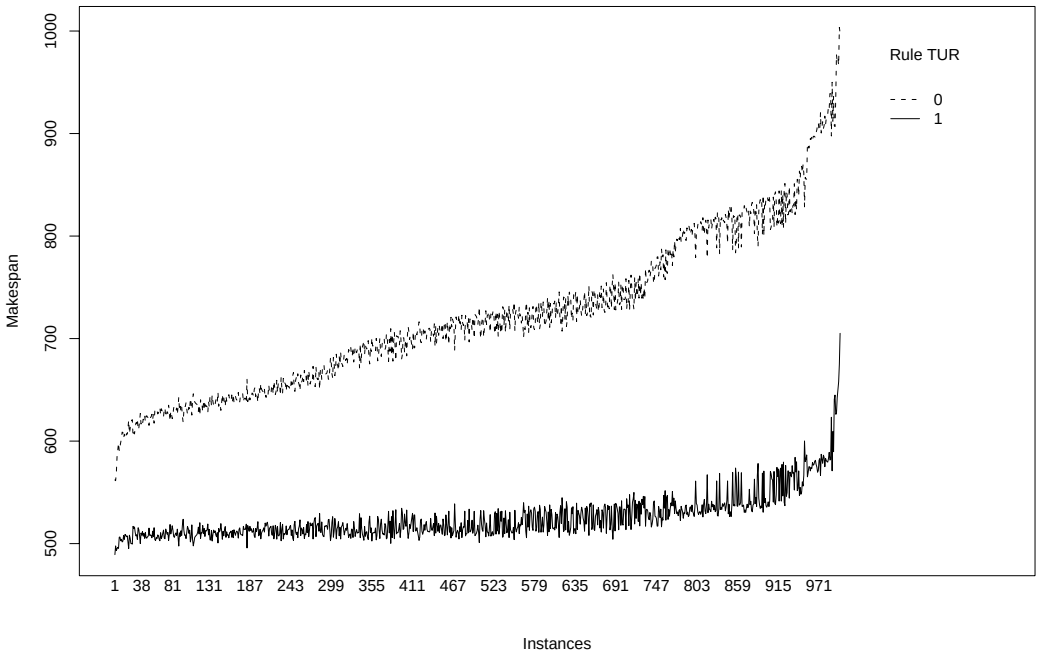
Figure 12: Makespan values obtained by each configuration of *ATA* activating and deactivating the proposed rules on the DTA_{Hom} problem class. For the coding system used to refer to the algorithms see Figure 11.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0000	588	749.0	812.5	808.0	864.0	1066
0001	544	644.0	703.5	714.5	774.3	1020
0010	476	650.8	709.0	717.4	772.0	1003
0011	521	612.0	669.0	684.0	741.0	999
0100	546	660.0	716.5	725.3	784.0	1040
0101	511	625.0	682.0	693.4	751.0	992
0110	554	699.0	756.0	760.8	817.3	1060
0111	517	607.0	665.0	678.8	738.0	976
1000	488	515.0	526.0	533.3	544.0	702
1001	474	500.0	513.5	516.1	527.0	687
1010	481	512.0	525.0	530.5	541.3	714
1011	474	496.0	508.0	512.6	523.0	681
1100	487	515.0	528.0	533.4	543.0	714
1101	489	518.0	531.0	537.6	548.0	740
1110	473	510.0	522.0	526.9	537.0	706
1111	472	497.0	509.0	513.4	525.0	699

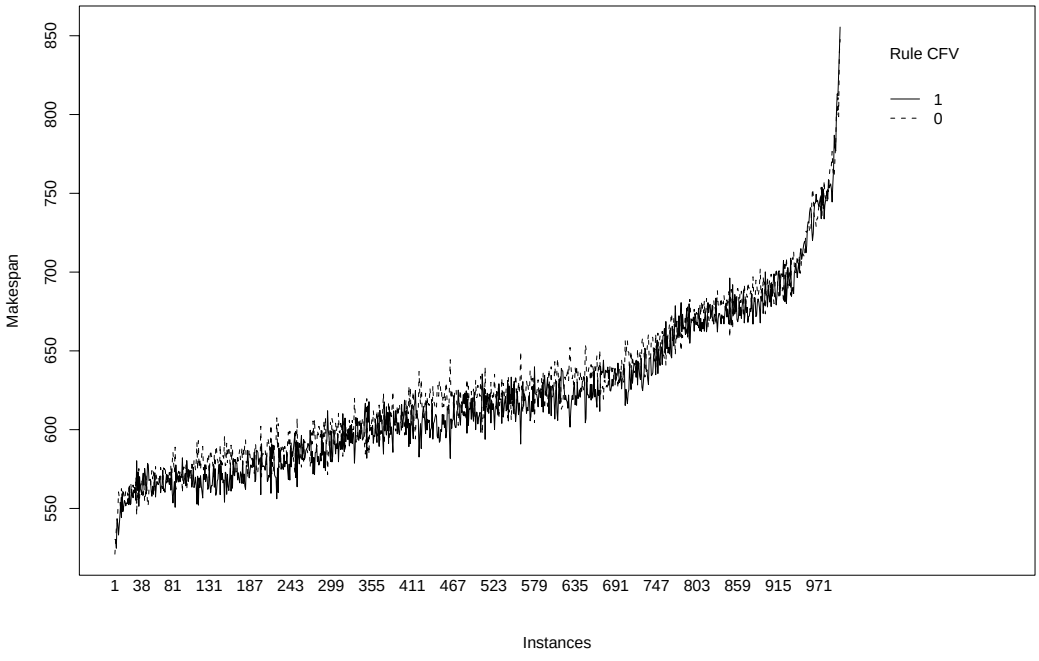
(a)



(b)

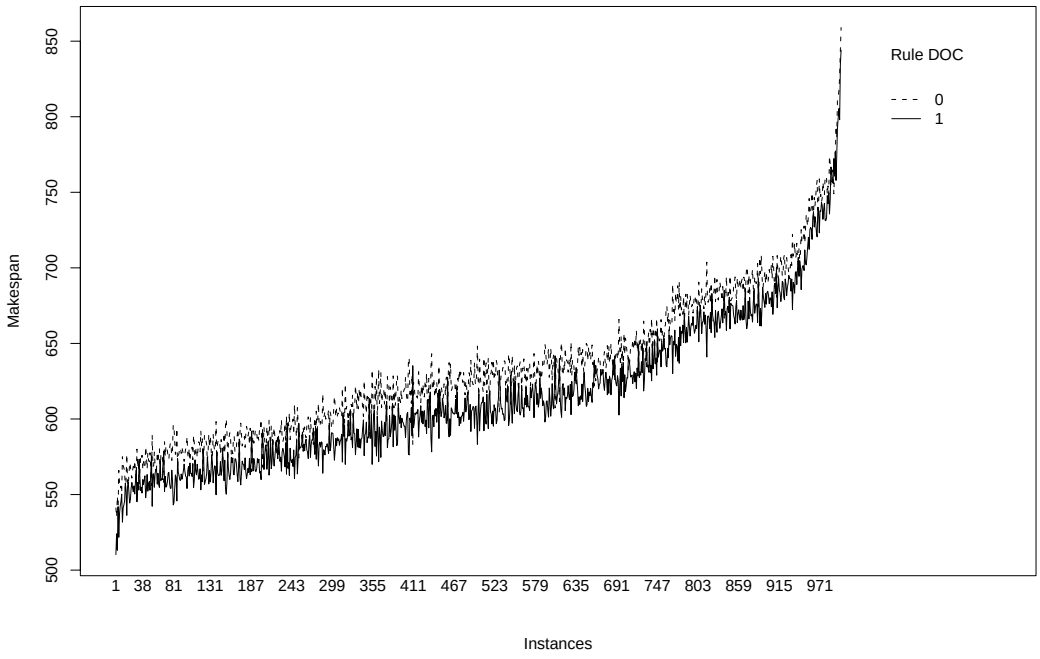


(a)

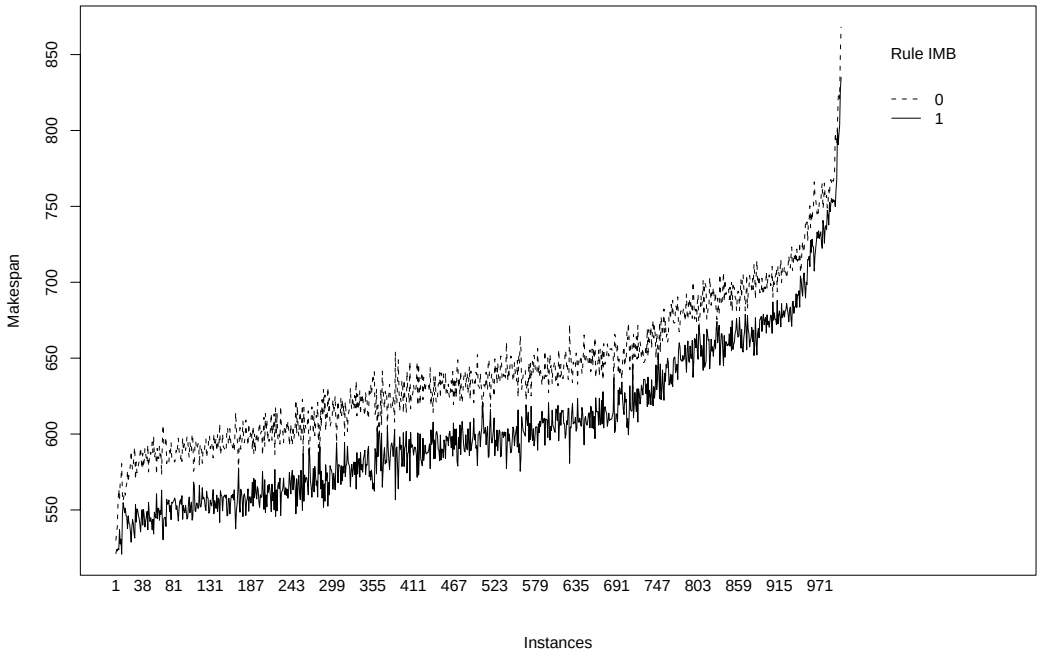


(b)

Figure 13: Makespan values on the ranked instances considering the rules *TUR* and *CFV*.

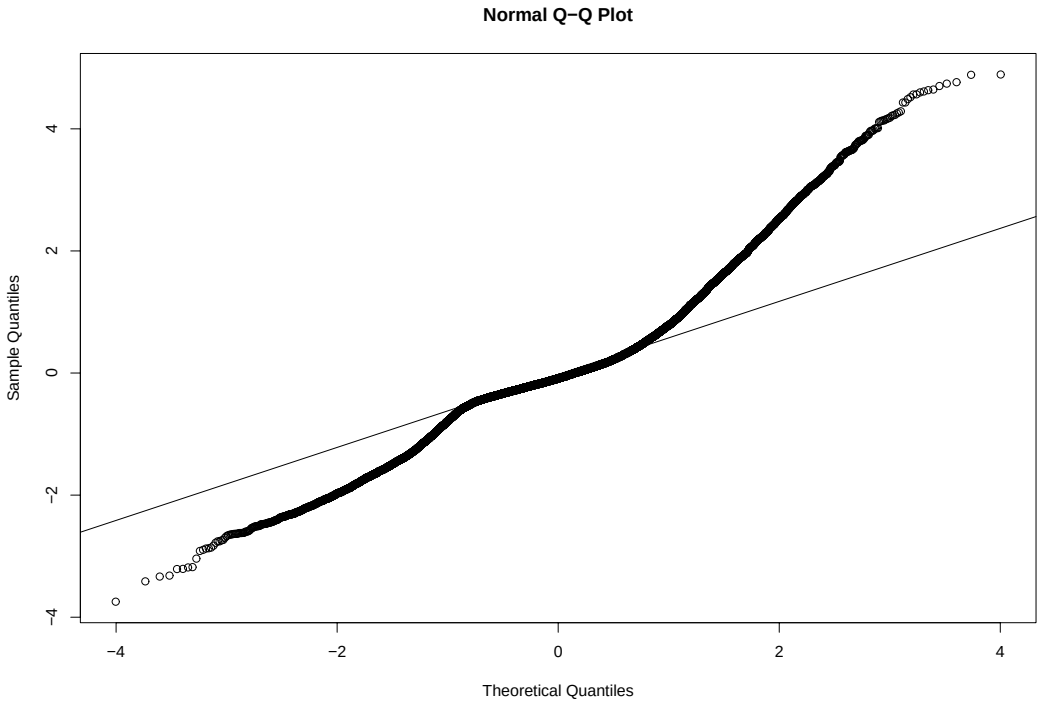


(a)

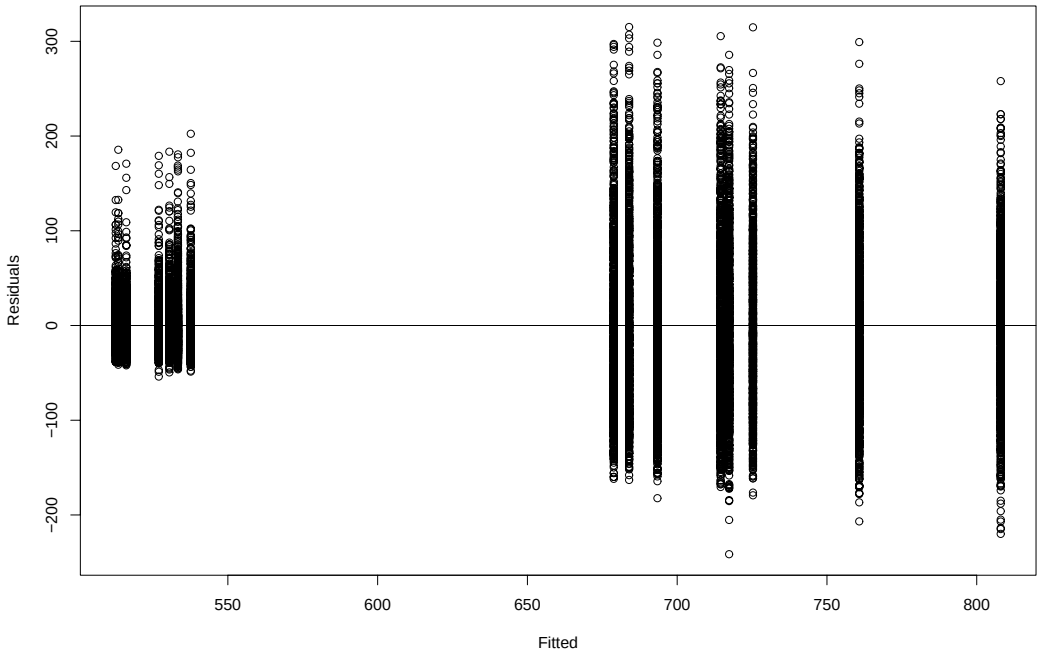


(b)

Figure 14: Makespan values on the ranked instances considering the rules *DOC* and *IMB*.

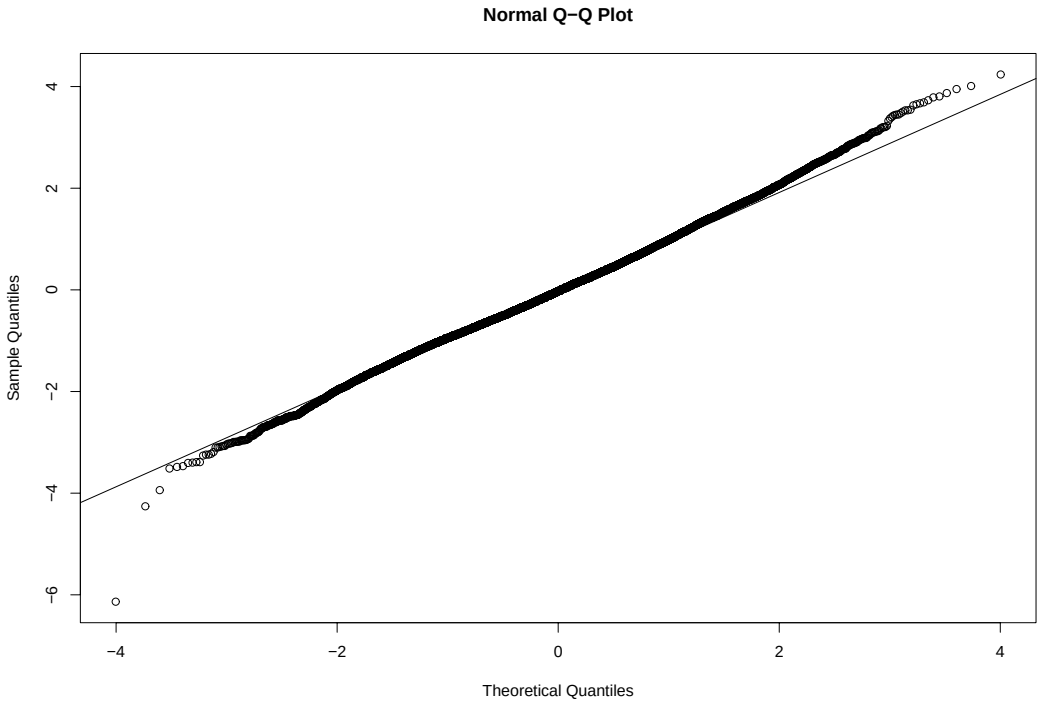


(a)

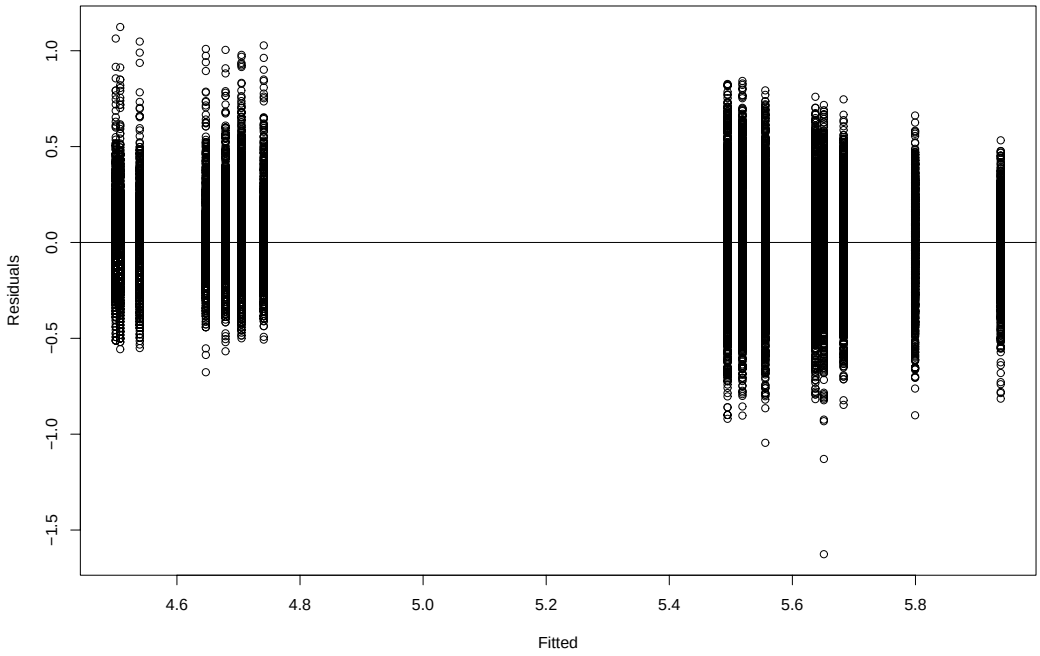


(b)

Figure 15: Residual check of the linear model using makespan data.



(a)



(b)

Figure 16: Residual check of the linear model using a transformation of the makespan data.

Table 8: Summary of the makespan values for all algorithms run on the class of instances of the heterogeneous problem. We compare the original algorithms and the insect-based algorithms with the Different Process Speed (DPS) rule activated. The median values show that algorithms with the DPS rule perform better than the respective algorithms without this rule. Again, *ATA* is the algorithm that has better performance compared with the others original algorithms. For a graphical view of these results see Figure 17. For a statistical test on these data we refer to Table 9.

	ATA	*R-WASP*	*ABA*	*LOCUST*	*MBA*	*ATAc*	*R-WASPc*	*ABAc*
Min.	427.0	487.0	428.0	647.0	429.0	428.0	426.0	426.0
1st Qu.	521.0	795.0	884.0	928.0	861.8	471.0	460.0	678.0
Median	548.0	847.0	941.5	952.0	906.0	486.0	506.0	837.0
Mean	557.5	845.6	917.8	959.5	896.4	485.6	525.6	790.7
3rd Qu.	578.0	897.3	979.0	977.3	944.0	499.0	561.0	913.0
Max.	956.0	1273.0	1341.0	1329.0	1321.0	622.0	1018.0	1311.0

Table 9: Paired Wilcoxon Rank Sum Test applied on the makespan of the algorithms on the class of instances of the heterogeneous problem. Values are obtained considering a confidence level of 95%. The table assures that differences between algorithms on the median values are significant.

	ABA	*R-WASP*	*ABAc*	*R-WASPc*	*ATAc*	*LOCUST*	*MBA*
R-WASP	$< 2.2e^{-16}$	-	-	-	-	-	-
ABAc	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-	-	-	-	-
R-WASPc	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-	-	-	-
ATAc	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-	-	-
LOCUST	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-	-
MBA	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	-
ATA	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$	$< 2.2e^{-16}$

An ANOVA analysis on this model is show in Table 7. The data show that *TUR* gives a large contribution to the makespan while *CFV* does not. Furthermore, also *IMB* and *DOC* bring a contribution less important than *TUR* but still significant.

6.2.2 *DTA* Class

This experiment considers two subsets of agents: one performs fast in half of the available types of tasks and slow in the second half and the other one vice versa. Agents should bid for tasks that they are able to process quickly. Moreover, they take into account the task type in their queue to minimize the number of setups. If an algorithm achieves this behavior it should obtain a low makespan. A low makespan implies a low number of setups and a low storage usage. In this experiment are compared all the original algorithms presented and the insect-based algorithms with the DPS rule activated. The algorithms used in the previous experiments are applicable to the heterogeneous case of the *DTA* problem. In fact they consider the different process times during the dominance contest. Comparing the algorithms with and without the DPS rule we are able to study its contribution.

Table 8 summarizes the makespan values achieved by each algorithm for this class of instances. The respective box plot is presented in Figure 17. For this experiment *ATAc* obtains the best results concerning the makespan and it is very close to the makespan lower bound (420 minutes). Moreover, we observe that the performance order of the original algorithms is the same as observed in Figure 9. Again, we confirm that adaptive algorithms perform better than the non-adaptive

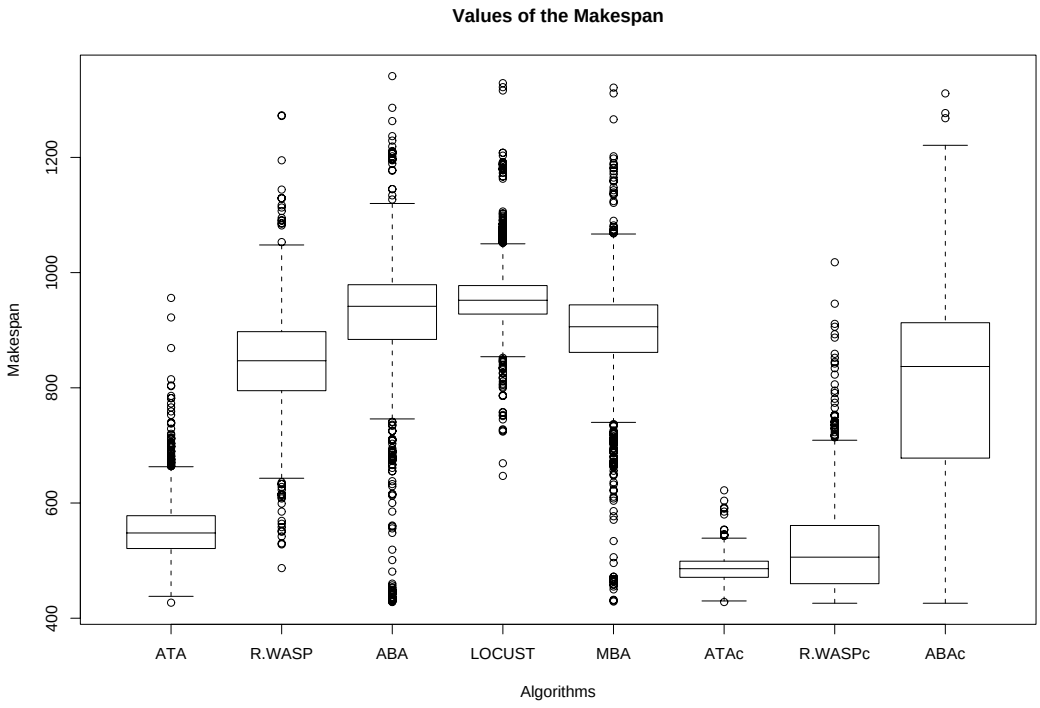


Figure 17: Box plot of the makespan values for all the eight algorithms run on the class of instances of the *DTA* problem. The figure confirms that *ATAc* is the algorithm that achieves the best results and that the extended algorithms perform better than the original ones. The DPS rule gives a high improvement of the performance especially to the *R-WASP* algorithm. Moreover, considering the relative position of the first five box plot with that ones in Figure 9 we can show that the *MBA* algorithm performs worse in this experiment and is near the *ABA* performance.

algorithm *LOCUST*. Only *MBA* has a worsening of performance. Algorithms with the DPS rule applied perform better than the respective algorithms without this rule. A deeper analysis about the makespan contribution of this rule is in the following paragraph.

To be sure of the real significance of the obtained results we applied a Wilcoxon Paired Rank Sum Test. The results presented in Table 9 show that different results of the makespan between algorithms are significantly different with a confidence level of 95%.

Figure 18 shows a box plot of the number of setups per agent and the storage dimension. Again, the trend is similar to the makespan plot of Figure 17. *MBA* has bad performances for setups and storage.

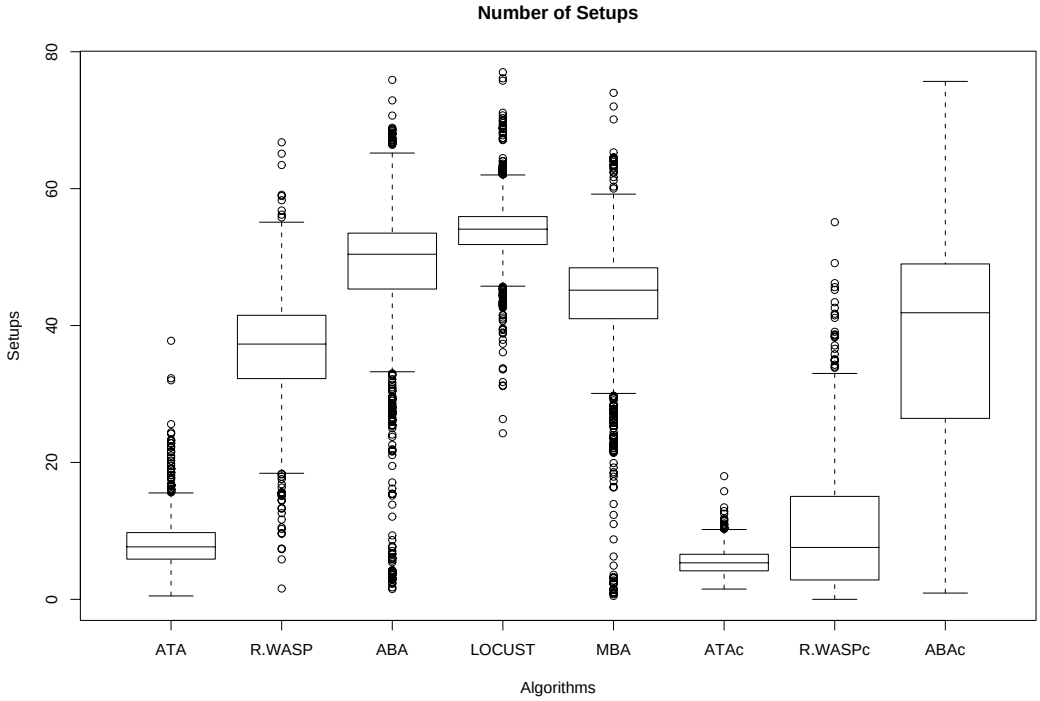
The significance of the DPS rule on the *DTA* problem The introduced rule gives a high bid probability to an agent that has a low process time for the current task. In this experiment we saved for each instance the makespan value that each algorithm achieves. For each pair of the insect-based algorithms and for each of the 1000 instances we evaluated and sorted the makespan differences. The results are presented in Figure 19. In most cases, differences are positive so the makespan of the original algorithm is higher than the extended algorithm. This behavior is similar for all the three algorithms considered. *R-WASPc* is the algorithm that receives the highest advantage from this rule.

7 Conclusions

In this paper we presented *Ant Task Allocation* (*ATA*), an algorithm inspired by the division of labor of social insects and based on the work presented by Cicirello *et al.* [8]. *ATA* introduces four rules in order to speed up the adaptation process for particularly unpredictable instances of the homogeneous case of the *Dynamic Task Allocation* (*DTA*) problem. Moreover, we presented the *Different Process Speed* (*DPS*) rule inspired by the division of labor observed in castes of social insects and applied to all the insect-based algorithms. Its aim is to increase the performance for the heterogeneous case of the *DTA* problem. This rule takes into account the different process time that agents have for different types of tasks during the bidding process. To understand the impact of the proposed algorithm we conducted a comparison with other solutions available in the literature for the *DTA* problem. We considered two possible real-world situations: a big painting factory with identical working agents and medium factory with two heterogeneous subsets of working agents. Particular attention was put on the experimental conditions and on the statistical analysis.

We have shown that *ATA* achieves the best results for the considered class of homogeneous problems and that *ATA* with the *DPS* rule activated (*ATAc*) achieves the best results for the considered class of heterogeneous problems. We have also observed that all adaptive algorithms always achieve better results than the non-adaptive *LOCUST* algorithm. Furthermore, we have shown that the *Market Based Approach* (*MBA*) achieves interesting results for the homogeneous case of the problem but not for the heterogeneous case. Moreover, the performance of the insect-based algorithms are ranked according to the time they was made. In fact the algorithm of Campos' *et al.* does not obtains good results, the algorithm of Cicirello *et al.* gives a significant improvement to the performance obtained by Campos' *et al.* as they show in paper [8] and our algorithm improves again the results of the solution of Cicirello *et al.*

Another important contribution of this paper has been the understanding the contribution of each introduced rule to the makespan. We have shown that *ATA* with the four active rules performs better than the original algorithm proposed by Cicirello *et al.* [8]. Moreover, our results show that the introduced rule *Threshold Update Rule* (*TUR*) gives a large contribution to the makespan while the rule *Calculation of the Force Variable* (*CFV*) does not. Furthermore, also the rule *Idle Machine does not Bid* (*IMB*) and the rule *DOminance Contest* (*DOC*) bring a contribution less important than *TUR*, but still significant. Finally, we showed that the *DPS* rule significantly improves the performances of all insect-based algorithms on the heterogeneous case of the *DTA* problem.



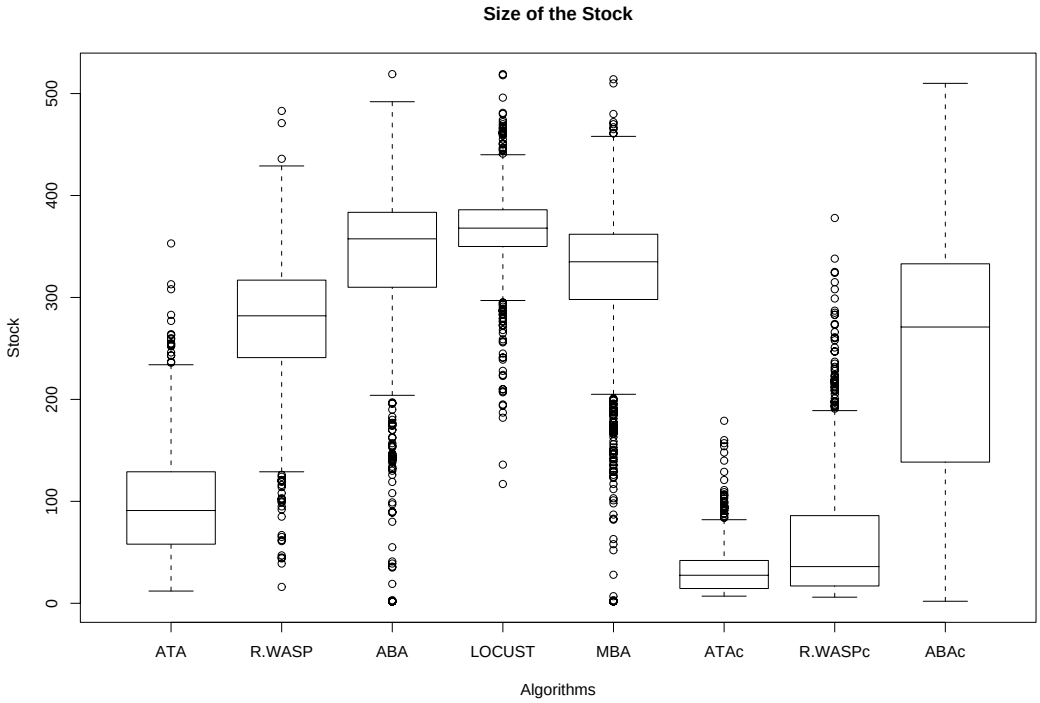


Figure 18: Number of setups per working agent and storage dimension for the experiment with heterogeneous working agents. Both the graphs show a trend similar to that one in Figure 17. Again, values of *ATAc* algorithm are the best ones. The DPS rule gives a significant increment also on these performances.

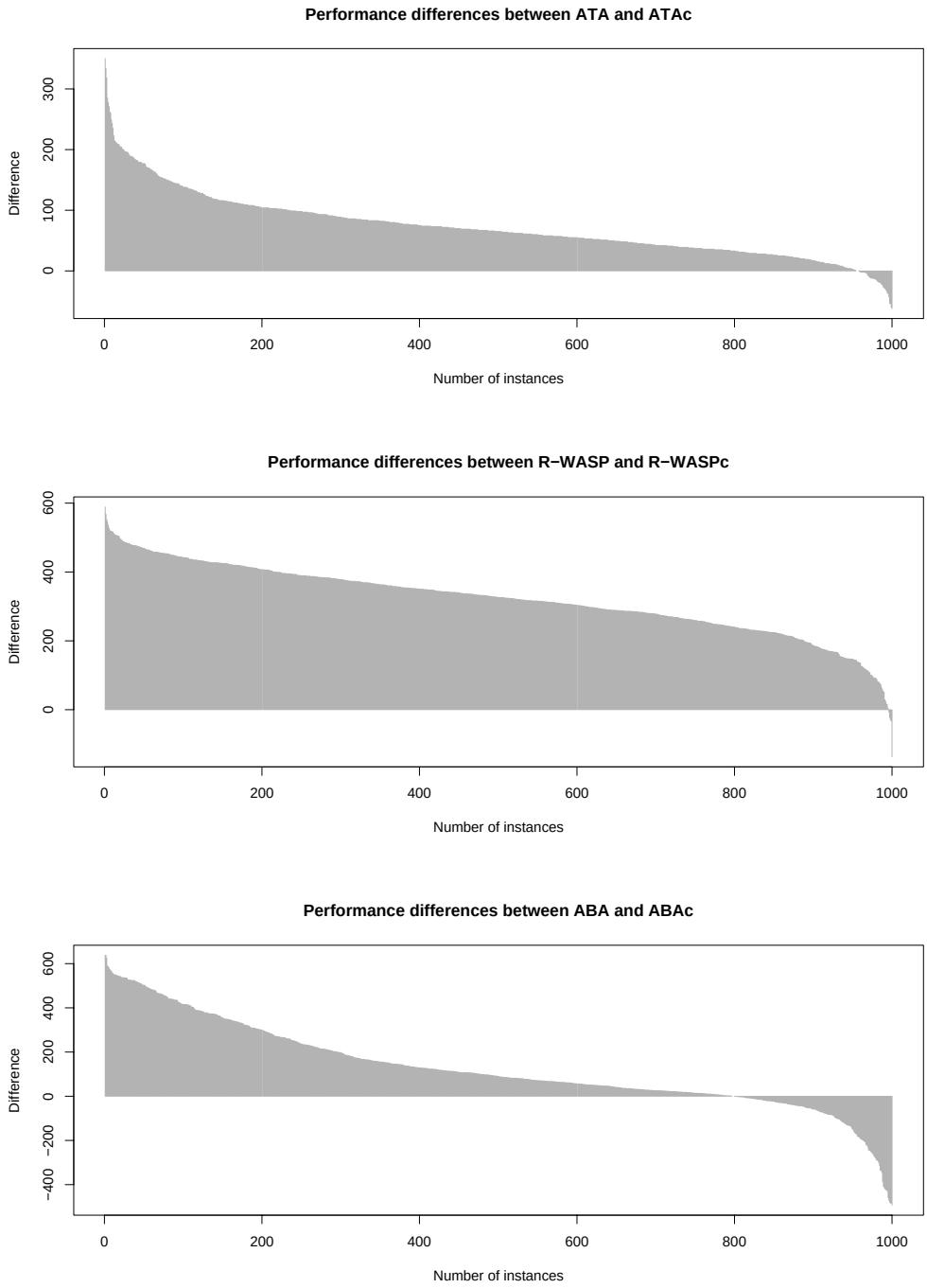


Figure 19: Makespan differences for the three insect-based algorithms without and with the DPS rule. The graphs show that for all the algorithms and for the major number of the 1000 instances considered the difference between the algorithm without and with the DPS rule is positive. This means that the DPS rule gives clearly a benefit to the makespan values.

We believe that this paper, beside introducing some original algorithms, also gives a clear and simple methodology to perform the experiments. Furthermore, it summarizes the work made until now on the *DTA* problem.

To complete this paper we have built a web site `http://iridia.ulb.ac.be/~rghizzioli/dta/` where it is possible to find the results, the source code of the instance generator, the algorithms and the EA tuner with interesting literature links. We hope other researchers will use this work as a starting point for new *DTA* problem extensions or to propose new algorithms to solve it.

References

[1] H.-G. Beyer. *The Theory of Evolution Strategies*. Springer, Berlin, Germany, 2001.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[3] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J.-L. Deneubourg. Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects. In D. Lundh, B. Olsson, and A. Narayanan, editors, *Biocomputing and Emergent Computation*, pages 36–45, Singapore, 1997. World Scientific.

[4] E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg. Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies. In *Proceedings Roy. Soc. London B*, volume 263, pages 1565–1569, 1996.

[5] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. Wiley-Interscience, 1978.

[6] N. W. Calderone and R. E. Page. Temporal polyethism and behavioural canalization in the honey bee. *Apis mellifera. Anim. Behav.*, 51:631–643, 1996.

[7] M. Campos, E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg. Dynamic Scheduling and Division of Labor in Social Insects. *Adaptive Behavior*, 8:83–96, 2000.

[8] V. A. Cicirello and S. F. Smith. Wasp-like Agents for Distributed Factory Coordination. Technical Report CMU-RI-TR-01-39, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Pittsburgh, PA, December 2001.

[9] S.-H. Clearwater. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore, 1996.

[10] E. H. Durfee and V. Lesser. Negotiating Task Decomposition and Allocation Using Partial Global Planning. *Distributed Artificial Intelligence*, 2:229–244, 1989.

[11] B.-A. Huberman and S.-H. Clearwater. Thermal markets for controlling building environments. *Energy Engineering*, 91:26–56, 1994.

[12] B. A. Huberman and T. Hogg. Distributed Computation as an Economic System. *Journal of Economic Perspectives*, pages 141–152, 1995.

[13] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38:705–717, 1989.

[14] R. Morley. Painting Trucks at General Motors: The Effectiveness of a Complexity-Based Approach. *Embracing Complexity: Exploring the Application of Complex Adaptive System to Business*, pages 53–58, 1996.

[15] S. Nouyan. Agent-Based Approach to Dynamic Task Allocation. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – Third International Workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 28–39. Springer Verlag, Berlin, Germany, 2002.

[16] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice-Hall, Englewood Cliffs, Englewoods Cliffs, NJ, 1995.

[17] G. E. Robinson. Modulation of alarm pheromone perception in the honey bee: evidence for division of labor based on hormonally regulated response thresholds. *J. Comp. Physiol.*, A 160:613–619, 1987.

[18] G. E. Robinson. Regulation of division of labor in insect societies. *Annu. Rev. Entomol.*, 37:637–665, 1992.

[19] G. E. Robinson, R. E. Page, and Z.-Y. Huang. Temporal polyethism in social insects is a developmental process. *Anim. Behav.*, 48:467–469, 1994.

[20] R. Schwartz and S. Kraus. Bidding mechanisms for data allocation in multi-agent environments. *Lecture Notes in Computer Science*, 1365:61–76, 1998.

[21] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, Technische Universität Berlin, 1975.

[22] G. Theraulaz, E. Bonabeau, and J.-L. Deneubourg. Response Threshold Reinforcement and Division of Labour in Insect Societies. In *Proceedings of the Royal Society of London B*, volume 265, pages 327–335, 1998.

[23] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS - Third Edition.* Springer, New Your, USA, 1999.

[24] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *Software Engineering*, 18(2):103–117, 1992.

[25] E. O. Wilson. The relation between caste ratios and division of labour in the ant genus phedoile. *Behav. Ecol. Sociobiol.*, 16:89–98, 1984.

[26] M. Wooldridge and N. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10, 1995.