Auction and Swarm Multi-Robot Task Allocation Algorithms in Real Time Scenarios

José Guerrero and Gabriel Oliver
Universitat de les Illes Balears (Dep. of Mathematics and computer Science)
Spain

1. Introduction

A group of several autonomous robots (multi-robot system) can perform tasks that with only one of them would be impossible to carry out or would take much more time, moreover, they are more robust and even can be cheaper, etc than systems with a single robot. In general, the problems that have to be solved to benefit from all these advantages are divided into three main stages: task division/planning, task allocation and motion planning. Task division stage, consists on dividing the general and complex mission into simple tasks that can be carried out by a robot and, if it's necessary, scheduling those simpler goals. The task allocation step will select the best robot or group of robots to execute each goal. Finally, the motion planning issues involve the robots' motions coordination to get the assigned tasks. Although these steps have been explained as independent and sequential stages, they are tightly connected and the decisions made in one level affect the whole system performance. For example, in Zlot & Stentz (2006) the authors proposed a task allocation method that combined planning, task decomposition and task allocation. Although this and similar efforts can be found in the literature. Each one of these steps is still an open problem. This paper will be focused on multi robot task allocation (MRTA) issues, without taking into account the other problems. Task allocation is one of the main problems in multi-robot systems, very especially when the tasks must be executed before deadlines, that is, in real-time scenarios. In most cases this problem is an NP-hard problem, and therefore nowadays there is not any algorithm that in a reasonable computing time gives the optimal tasks allocation. Two main paradigms have been proposed in recent years to try to manage this problem in both real-time and non real-time scenarios: swarm and auction methods. At present, there does not exist any study that compares both strategies when the robots must carry out tasks with soft or hard real-time restrictions. Other works, for example Kalra & Martinoli (2006), compare auction and swarm methods but using tasks without deadlines. Therefore, the first objective of this work is to compare all those methods under different scenarios to identify the weak points of each paradigm when a deadline is assigned to the tasks.
Firstly, our work presents and study three auction-like strategies based on existing ones: Sequential Unordered Auction (SUA), Earliest Deadline First Auction (EDFA) and Sequential Best Pair Auction (SBPA). In all these cases there is a central auctioneer who receives the bids from all the robots and decides the allocation of the tasks. These strategies differ between them on the way the auctioneer announces the tasks to the robots and on the

robots' answers are processed. Besides the auction methods, a classical swarm strategy has been taken into account. The results show that in most cases the most complex algorithm(SPBA) outperforms the other auction based procedures. Besides EDFA is better than the simple SUA strategy. Finally, swarm strategy is better than EDFA or SUA when the number of robots and tasks is not very large.
In Gerkey & Mataric (2004), the MRTA problem was divided using three axes: multi-task robots (MT) vs. single-task robots (ST) depending on whether multiple tasks can be assigned to the same robot or not. The second axes is single-robot tasks (SR) vs. multi-robot tasks (MR) where SR means that only one robot can be assigned to a task, while MR means that several robots (a coalition) can execute concurrently a task. The last axes is, instantaneous assignment (IA) vs. time-extended assignment (TE) where in IA the allocation is made without taking into account the future incoming tasks. In terms of this taxonomy, in this paper we focus on a ST-SR-IA and MT-SR-IA task allocation, our previous work Guerrero & Oliver (2010) was focused on ST-MR-IA approaches in real time scenarios. In MT-SR-IA, each single robot must have the capabilities for deciding how to schedule its assigned tasks. Thus, we extend the original MT category to take into account problems where a robot has the schedule several tasks. For example, in the MT-SR-IA strategy implemented in this work, each robot decides in what order it will visit its assigned goals, that is, the robot has to solve the traveling Salesman Problem (TSP). We have to note that the TSP is also an NP-Hard problem. The results of our experiments show that in a MT-SR-IA system the local planning made by each robot is more important than the algorithm used by the central auctioneer. From these results, this work presents a new MRTA algorithm called Earliest Deadline First Best Pair (EDFBP) which, depending on the robots' scheduling capabilities, uses a more (algorithmic) complex MRTA algorithm or it simplifies the task allocation process.
A classical foraging task has been used to verify our methods. In this mission each object have to be gathered before an specific deadline. The performance measure used to compare the systems is the number of tasks finished before their deadline. A simulator developed by the authors, called RoboSim, has been used to execute most of the experiments, this simulator allows to execute a great number of tests, with a very large number of robots in the colony in a very short time. Some experiments with less robots has also been carried out with the well known Player/Stage simulator Collett et al. (2005).
The rest of this paper is organized as follows: section 2 presents some relevant work in the field of multi-robot task allocation focused very specially on auction and swarm methods; section 3 shows a formal definition of the problem to solve and specifies the details of the real time foraging task; section 4 explains the algorithms implemented when only one task can be assigned to each robot; section 5 extends the algorithms already explained in section 4 to allow multiple tasks assigned to the same robot (MT tasks) ; section 6 shows the results of the experiments using the swarm SUA, EDFA and SBPA methods; section 7 explain the new EDFBP strategy and the experiments carried out to validate it; finally, section 8 exposes some conclusions and the future work of our research.

2. Related work

A lot of research has been done to solve the multi robot task allocation problem, but it is still an open issue. The proposed solutions can be classified into three main groups: centralized, negotiation and self organized system(swarm). In centralized methods there is a central agent who has all the information about the tasks, the robots, environment, etc. and takes all the decisions. The centralized algorithms can use classical optimization methods, like for

example Linear Koes et al. (2005) or Dynamic programming, which get very good results in terms of number of tasks executed per time unit and of energy required by the robots, but they need too much computation time for being used in a dynamic environment. Some other centralized methods, like the best pair selection, use a greedy algorithm that produces a worse allocation but needs less time. This strategy has also been used for non centralized methods, like Broadcast Local Eligibility (BLE)Werger & Mataric (2000). In this work the SBPA is based in this concept, but on an environment where the robots use an auction to get the tasks. In the self organized or swarm systems each robot takes the decision by itself, without any kind of negotiation and executing very simple procedures like in Sahin et al. (2008). Probably, the most used swarm algorithms are the response threshold methods Agassounon & Martinoli (2002); Yang et al. (2009); Ferreira et al. (2010), where each robot has a stimuli associated with each task. When the level of the stimuli exceeds a threshold, the robot starts its execution. The pure threshold based systems, and in general the pure swarm systems, don't require any kind of communication mechanisms. Some studies like Ducatelle et al. (2009) studied how the communication can improve the performance of their swarm method for foraging like tasks. Nonetheless, a disadvantage of these systems is the interference produced when two or more robots decide to execute the same task, when this task can only be executed by a single robot. The swarm method proposed in this work does not use any kind of communication between robots and they will only be able to get information from a central auctioneer. Finally, the negotiation methods are a middle way solutions, among these methods the most used ones are the auction based solutions like for example Gerkey & Mataric (2002); Dias & Stentz (2003); Vig (2006). In this kind of systems, the robots act as self-interest agents and they bid for tasks. The robot with the highest bid wins the auction process and gets the task. The bids are adjusted to the robots' interest (capacity) to carry out the goal. Thus, the best robot for a specific task can be chosen, but they need communication mechanisms between robots.
In this paper we focus on auction and swarm systems. In Zheng et al. (2006) the authors used an auction method with sequential allocation, similar to the SUA method developed in this work, but they improve the sequential results using a prediction of futures assignments. Our improvements of the sequential auctions are based on real time concepts like EDF, or on reducing the algorithmic complexity with EDFBP method. Hybrid methods, which combine both auction and swarm approaches, has also been introduced by several authors like Zhang et al. (2007) or Dasgupta & Hoeing (2008). In all these cases only a single robot can be assigned to the same task (SR scenario). A lot of works allow has been done to solve the multi-robot assignment problem (MR problem) where tightly cooperation between robots is required to execute a task Jones (2009); Vig (2006); Service & Adams (2010); Zheng & Koenig (2008). MR problems are out of the scope of this work and they have been studied by the authors in Guerrero & Oliver (2010; 2004)
Very few work has been done to test swarm systems in real time scenarios, where the tasks must be executed before a deadline. In del Acebo & de-la Rosa (2008) the authors proposed a swarm method to solve conflicts between robots using a local planning and high communication skills to perform a local auction process. Hence, no learning algorithm was proposed to fit the algorithm's parameters that, as the authors pointed, affect dramatically the results. The swarm algorithm proposed in our work is much more simple in order to simplify the comparison with the other methods. Auction strategies have also been used in real time scenarios, for example, in Jones et al. (2006) the robots learnt what to bid to increase the number of tasks that fulfill a deadline. A great effort has

been done to solve the scheduling and allocation problem on multi-processors systems Pinedo (2008). Some of these works use the well known Earliest Deadline First (EDF) Goossens et al. (2002), the same method used in this work to decide in what order the tasks must be auctioned.
Finally, other works, like for example Kalra & Martinoli (2006), compare both auction and swarm without deadlines under different types of communication restrictions. The authors tried to use a classical response threshold algorithm, but in all cases the results with a deterministic selection were better. The same swarm strategy used in Kalra & Martinoli (2006) has been implemented to execute our swarm experiments.

3. General task description

In this section, we will formalize the task allocation problem previously sketched and we will explain the main problems that presents.
We have a set of tasks $T = \{t_1, t_2, ..., t_n\}$ and a set of robots $R = \{r_1, r_2, ..., r_m\}$ where in general $n \neq m$. Each task t_i has associated a set of characteristics $C_i = \{C_{i1}, C_{i2}, ..., C_{il}\}$, these characteristics can be, for example, the location of the task (x and y position), the amount of work to do (object's weight, area to clean, ...), etc. Each task i has a set of restrictions RT_i that must be satisfied by the robot or robots that will be assigned to it. Some examples of restrictions can be: number of robots that can simultaneously execute the task, sensorial skills that the robots must have, time restrictions like a deadline, etc. So, we want to find a task allocation function $A : T \rightarrow R$ that assigns to each task a set of robots in such way that robots' skill meet the task's restrictions taking into account the task's characteristics. We can also define a utility function $U : A \rightarrow \Re$ which given a task allocation returns its utility, that is, a real number that indicates how "good" is an allocation. Thus, our goal will be to find a A function that maximize the total utility U. In this work the utility function will be the number of tasks that fulfill its deadline.

3.1 Foraging task

To validate the algorithms, the classical like foraging task will be used. This task is defined as follows: some randomly placed robots have to pick up some randomly placed objects in the environment. New objects can arrive to the environment following a poisson distribution. There is also a central agent who knows the position of all the objects and will inform (broadcast) this information to all the robots. Each object to gather has a weight and each robot has a work capacity. The robot's work capacity is the amount of object's weight that the robot can process per time unit. The robots stops when it is near an object and starts to process the object for a time equals to *weight/work Capacity*. After this time, the robot goes back to its idle state and tries to get another task.
Each task has associated a utility (U), and a deadline time DL_i. DL_i is the time instant before which the task must be finished. The deadline time for a task will start its countdown just after the task appears in the environment. If the tasks' deadline has been met, then the group receives the utility of the task as a benefit. Otherwise, the task will disappear of the environment and no other robot will be able to execute it anymore. Then, our goal will be maximize the total utility, and thus, minimize the number of tasks executed after its deadline. In the experiments executed all the tasks had the same utility.

4. Implemented approaches for ST-SR-IA

In this section we will explain the four first approaches implemented in this work when only a task can be assigned simultaneously to a robot, that is, when the ST-SR-IA strategy is used. As it has been said, these strategies are: Sequential Best Pair Auction (SBPA), Earliest Deadline First Auction (EDFA), Serial Unordered Auction (SUA) and swarm. In all cases there will be a central auctioneer who will send the tasks information to the robots and decide what robot wins each task when the auction methods are executed.

4.1 Serial best pair auction: SBPA

We have used the classical best pair selection approach very similar to the selection method used in Broadcast of Local Eligibility (BLE) Werger & Mataric (2000) but into an auction process. Each time that a new task appears in the environment or when a robot finishes its execution, a central auctioneer starts a new auction round. The process followed by the auctioneer can be seen in algorithm 1. Firstly, it requests for a bid for each task to all the idle robots (lines 1-3). The idle robots (without any task assigned) bid using its expected time to finish the task, as long as they are able to execute the task before its deadline. Each robot uses its kinematic characteristics to know how much time it will need to finish a task. Then the auctioneer selects in each iteration the pair robot-task with the best execution time and notifies this election to the robot. This algorithm is similar to the studied in Gerkey & Mataric (2004).
The detailed analysis of the algorithmic complexity of our SBPA algorithm (algorithm 1) is as follows: let's n be the number of tasks (objects) and m the number of robots, then the complexity of the loop in lines 1-3 is $O(m)$. To find the best robot-task pair (line 5) it is required to test each robot and each task, but in each iteration a robot and a task are removed, therefore the complexity of the lines 1-8 is $O(\sum_{i=0}^{min(n,m)}(n-i)(m-i)))$. Thus, the total algorithmic complexity of the SBPA algorithm is $O(m+\sum_{i=0}^{min(n,m)}(n-i)(m-i))$. As it can also be seen, in each iteration the auctioneer sends a award message to a robot, therefore the cost of the communication system is equal to $O(m+min(n,m))$

Algorithm 1 SBPA algorithm for the ST approach

```
1: for all unassigned task t do
2:     Ask for a bid to all idle robots
3: end for
4: repeat
5:     Select the best robot-task pair (best bid).
6:     Send an award message to the selected robot
7:     Remove the task and the robot of the list
8: until There is no more unassigned robots or tasks
```

4.2 Earliest Deadline First Auction: EDFA

The earliest deadline first (EDF) is a very well known method in processors scheduling for real time environments. In these cases, the tasks (processes) are sorted by deadline, in such a way that the tasks with the nearest deadline are firstly processed. The same concept has been used in this work to implement the Earliest Deadline First Auction (EDFA) strategy, as it can be seen in algorithm 2. When a new task appears or when a robot becomes idle, the central auctioneer orders all the available tasks by deadline, and sends a request to all the robots for the first task, the task with the earliest deadline. Then, the robots that are able to finish the task before the deadline bid for the task using its expected execution time. Finally, the auctioneer selects the best robot (the robot who is able to finish the task first). If there are more tasks, the task with the next earliest deadline is selected and the process starts again.

Algorithm 2 EDFA algorithm for the ST approach

```
1: sort the list of tasks T by deadline (EDF)
2: for all task t in T do
3:     Ask for a bid to all idle robots
4:     Select the best bid
5:     Send an award message to the assigned robot
6: end for
```

Following the same reasoning as for the SBPA, the analysis of the EDFA complexity is as follows: let's *n* be the number of tasks and *m* the number of robots, then, the complexity for the sorting algorithm (line 1) is $O(nlog(n))$. Then, the auctioneer must check each robot's bid to get the best robot for a task, but we have to take into account that the robots already assigned to earlier objects does not need to be considered. Thus, in each iteration a robot is assigned to a task and the complexity will be equal to: $O(nlog(n)+\sum_{i=0}^{\min(n,m)}(m-i))$. The cost of the communication is similar to the already explained for SBPA algorithm.

4.3 Serial Unordered Auction: SUA

This is the more simple auction strategy implemented in this work. As the tasks arrive to the central auctioneer, it starts a new auction round for each one of them. That is, the task are processed in a sequential way, like in a FIFO cue. When an auction round is started, the robots bids for this task using its expected execution time, as long as the robot is able to finish the task before the deadline. Then, the robots send their bid to the auctioneer who selects as the auction winner the robot with the minor execution time. Finally, if there are more tasks, a new auction process is started again until all objects have been processed. Thus, the central auctioneer does not have to make any decision about the order the task must be offered to the robots.

The computational complexity analysis of this algorithm is as follows: let's *n* be the number of tasks and *m* the number of robots, then the auctioneer for each task has to check all the bids, that in the worst case will be equal to the number of robots. Following the same reasoning as in the EDFA algorithm, we can see that the complexity of this algorithm is $O(\sum_{i=0}^{\min(n,m)}(m-i))$. The communication complexity is the same as in the EDFA strategy.

4.4 Swarm

In the swarm strategy each robot makes the decision about what task it will process by itself, without taking into account any other robot. As new tasks arrive, the central auctioneer sends to all the robots the information about them. Then, each idle robot selects the nearest task to it as its next objective. To select a task the distance between the robot and the task must be lower than a value D, thus, we avoid assigning a robot to a very distant task. Other more complex strategies have been used, like response threshold, but, as was pointed in Kalra & Martinoli (2006) for tasks without deadlines, the nearest task first strategy outperform response threshold in all cases. The complexity of this selection algorithm is obviously $O(n)$ where n is the number of tasks. Here the complexity of the communication system is equals to $O(n)$ because the central auctioneer should send to all the robots the information of the n tasks. Table 5 summarizes the complexity of all the single task algorithms proposed in this work and includes a the new EDFBP method that will be explained later.

5. Multi-task per robot: MT-SR-IA approaches

In this section we will explain the MT-SR-IA strategy implemented for this work. In the MTSR- IA strategies the robots can have more than one assigned task, and therefore, they have to schedule all its objectives. In our case, each robot has an scheduling, P with the objects to gather, thus, this list is the path or plan that the robot will follow. When a robot receives a new offer to execute a task t it will try to add t in P creating a provisional scheduling $P' = P \bigcup t$, in such a way that the total length of P' is as short as possible and all tasks in P' meet the deadline. Then, the robot bids for t using the difference of expected time between P' and P, that is, the bid value $b(t)$ will be:

$$b(t) = \sum tExpected(t_{j-1}, t_j) \qquad (1)$$

where *tExpected*(a, b) returns the expected time that the robots would need to go from task a to task b, the task t_0 is the initial position of the robot. Other bidding strategies has also been tested, like bid the total execution time of the new path, but the equation 1 provided the best results. The central auctioneer will select the best robot (the robot with the lowest b value) using any of the already explained strategies (SUA, EDFA or SBPA). Finally, provisional path P' is assigned to the selected robot.
The minimization of the path P' is a NP-Hard problem called the traveling salesman problem with deadlines in metric spaces, Δ-$D_L TSP$, where the path to minimize can not be a cycle. A lot of work has been done to try to solve the Δ-$D_L TSP$ problem Bckenhauer et al. (2009); Bansal et al. (2004). In this paper the nearest neighbor strategy has been used to get P' from P and t, that is, the new task is inserted between each two tasks in the path P, and the shortest one is selected. If there is no way to create a new path P' that meets all the deadlines, the robot will not bid for the task. Let's n be the number of tasks in P, then to create P' we have to make $n + 1$ tests (places where insert the new task) and $n + 1$ more to verify that all tasks meet the deadline, therefore the complexity of this algorithm is $O(n^2)$. The complexity of this algorithm is quite low but its completeness is not bounded.
This new strategy involves that the algorithm 1 has to be updated. As it can be seen in algorithm 3, now after each assignment the central auctioneer has to ask for a bid to all the robots again (lines 2-4) because the bidding of a robot in one iteration will depend on the decisions made in the last assignment. The SUA and EDFA methods have also been modified, in such a way that now the auctioneer asks for bids before auctioning each task. Thus, the complexity from the communication point of view, has been increased in all cases. Now for EDFA and SUA, for each task the auctioneer has to send a broadcast request to all the robots (1 message), wait for the m bids (one for each robot) and finally send the award to

the winner (1 message). Therefore the complexity of the communication is, in the worst case, $O(n * (m+2)) = O(nm)$. This is very similar for the SBPA method, but in each iteration of algorithm 3 the robot has to send a broadcast message requesting for the remaining tasks (1 message), this message will be much more longer than the sent in a EDFA or SUA strategy. Then, the auctioneer receives the bids from all the robots (*m* messages) and finally it sends the award to the winner (1 message). The complexity from the communication point of view will be $O(n * (m + 2)) = O(nm)$, the same as in EDFA or SUA, but we have to take into account that the length messages will be longer.

Algorithm 3 SBPA algorithm for the multi task approach (MT)

```
1: repeat
2:     for all unassigned unassigned task t do
3:         Ask for a bid to all robots
4:     end for
5:     Select the best robot-task pair (best bid).
6:     Send an award message to the assigned robot
7: until There is no more unassigned robots or tasks
```

As it is shown in table 1, can be easily seen the algorithmic complexity of the algorithms with MT assignment must be recalculated. In all cases, the complexity has increased compared to the single task assignment algorithms.

SUA	$O(nm)$
EDFA	$O(nlog(n) + nm)$
SBPA	$O(n+\sum_{i=0}^{n} m(n-i))$

Table 1. Complexity order of the algorithms introduced in this work in a multi task approach (MT). *n* is the number of tasks, *m* is the number of robots

6. ST-SR-IA and MT-SR-IA experiments

In this section we will show the results of the experiments carried out to study the behavior of the methods explained in sections 4 and 5, that is, the ST-SR-IA and MT-SR-IA approaches (SUA, EDFA, SBPA).

6.1 Experiments design: RobSim and Payer/stage simulator

We used a simulator, called RobSim, developed in our university to execute most of our experiments. RobSim is a simulator that allows to emulate the behavior of a very large population of robots and a huge number of tasks. After each time period the simulator updates the robots' positions and processes all the events happened in that period: new object in the environment, task executed successfully, expired task's deadline, etc . To speed up the simulation time, we assume that the robot can not collide with any other object in the environment and therefore, an obstacle avoidance algorithm is not needed. Thus, RobSim can be considered as a nonrealistic simulator but it is very useful tool to compare the global performance of the different tested strategies. To carry out more accurate experiments with

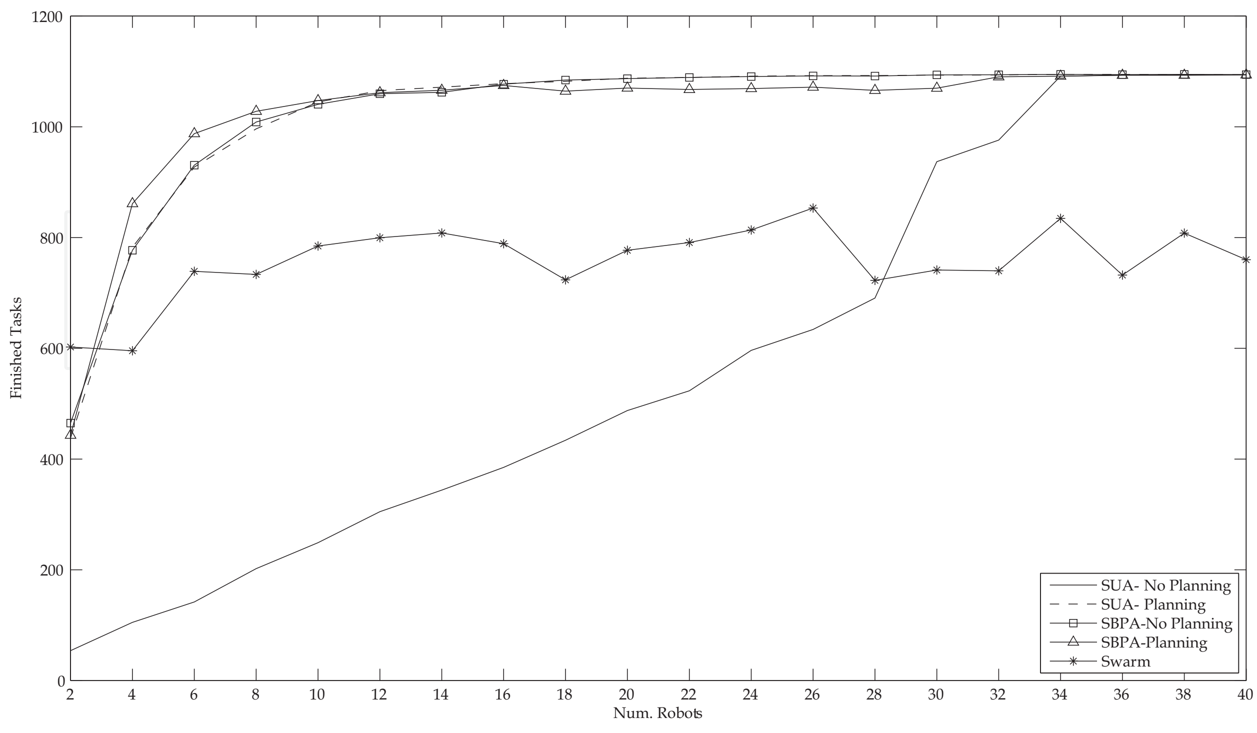
a less number of robots and tasks we have use the very well known and more realistic Player/Stage simulator.
We have executed in the RoboSim experiments with the foraging task already explained in 3.1. New objects to pick up appear in the environment following a poisson process with parameter λ, where λ is the average number of new tasks that will appear in the environment per time unit. Three different values of λ have been used: 0.05, 0.1, and 0.3. Moreover, the followings 5 different kinds of task sets were used:

- 1st Environment (Uniform Tasks): all tasks had the same weight (1 weight unit) and the same deadline time (250 time units)
- 2nd Environment (Weighted Tasks): all tasks had the same deadline time but their weights were generated following a uniform distribution between 1 and 70 weight units.
- 3th Environment (Random Deadline): all tasks had the same weight but the tasks' deadlines were generated according to a uniform distribution between 100 and 500 time units.
- 4th Environment (Hybrid Tasks): similar to the Random Deadline but with a probability of 0.2, the new task has not an associated deadline, that is, it has no time restrictions. Therefore, tasks with and without deadlines coexist in the same environment.
- 5th Environment (Tasks without deadline): there is no deadline assigned to any task, and its weight is equal to 1.

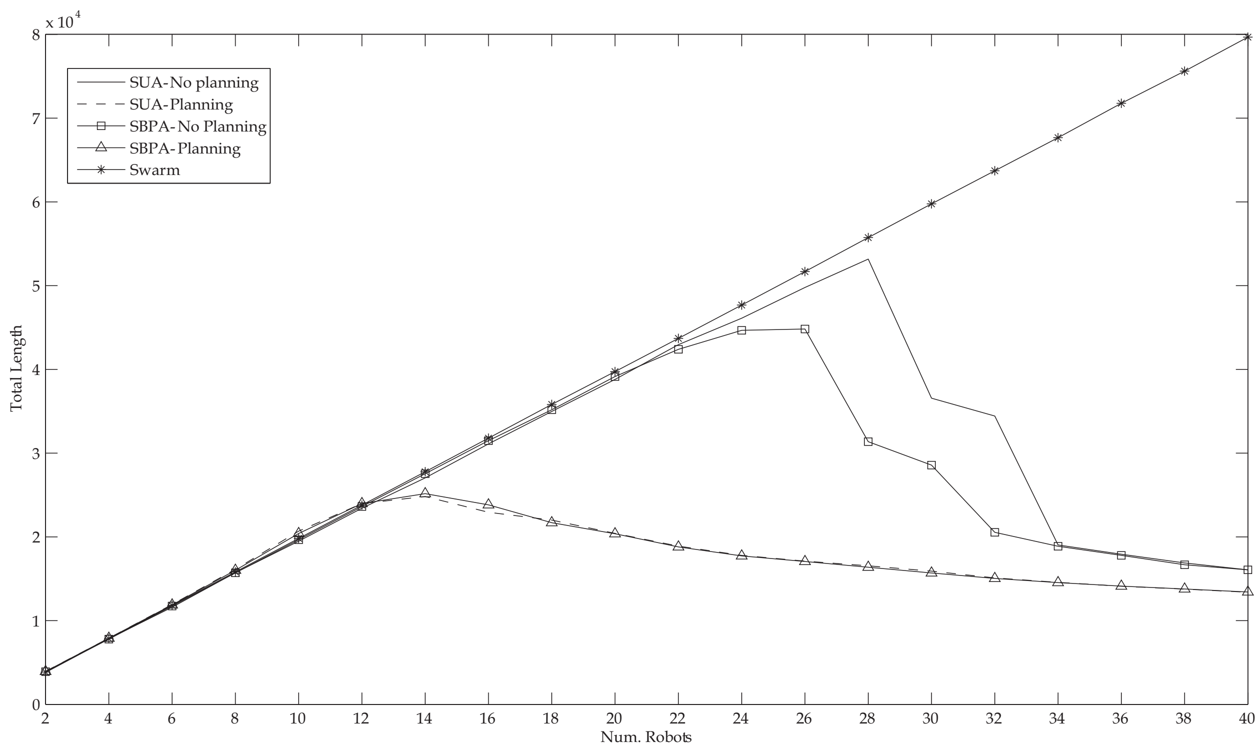
All robots in the colony had the same characteristics they move in a 160m.x160m.environment. The number of robots varied between 2 and 40. 1920 simulations were executed, each one lasting 10000 time units during which 270000 objects were processed.

6.2 Experiments with RoboSim

Here we will explain the main results of the experiments carried out with the RoboSim simulator over a large number of tasks and objects.
Figure 1(a) shows the total number of objects finished when no deadline is assigned to the tasks and with a $\lambda = 0.1$ and a $D = \infty$ for the swarm strategy. The planning methods means (5th. environment) the strategies with a MT-SR-IA strategy, that is, when the local path scheduling, explained in section 5, is performed. The total length covered by all the robots under the same environment is shown in figure 1(b). The EDF strategy has not been tested because without deadlines its results are equal to the SUA method ones. As it can be seen, when the ST-SR-IA (no planning) is used in SUA strategy the number of tasks finished is increased in a quasi lineal way with regards to the number of robots. This number of tasks is greater than swarm strategy if the number of robots is grater than 28. On the one hand, the SBPA method is not affected by the local planning, in both cases MT and ST the number of tasks is very similar. On the other hand, the planning clearly improves the SUA strategy results getting similar results as the SBPA method. For all strategies, except for swarm, there is a number of robots from which the total traveled path decreases. In the swarm system this total length increases lineally regards to the number of robots due to the interference between robots, produced when two or more robots select the same task simultaneously.
Figure 2(a) shows the total number of finished tasks with $\lambda = 0.05$ and figure 2(b) shows the total length traveled by all the robots in this same scenario. In this experiments the SUA

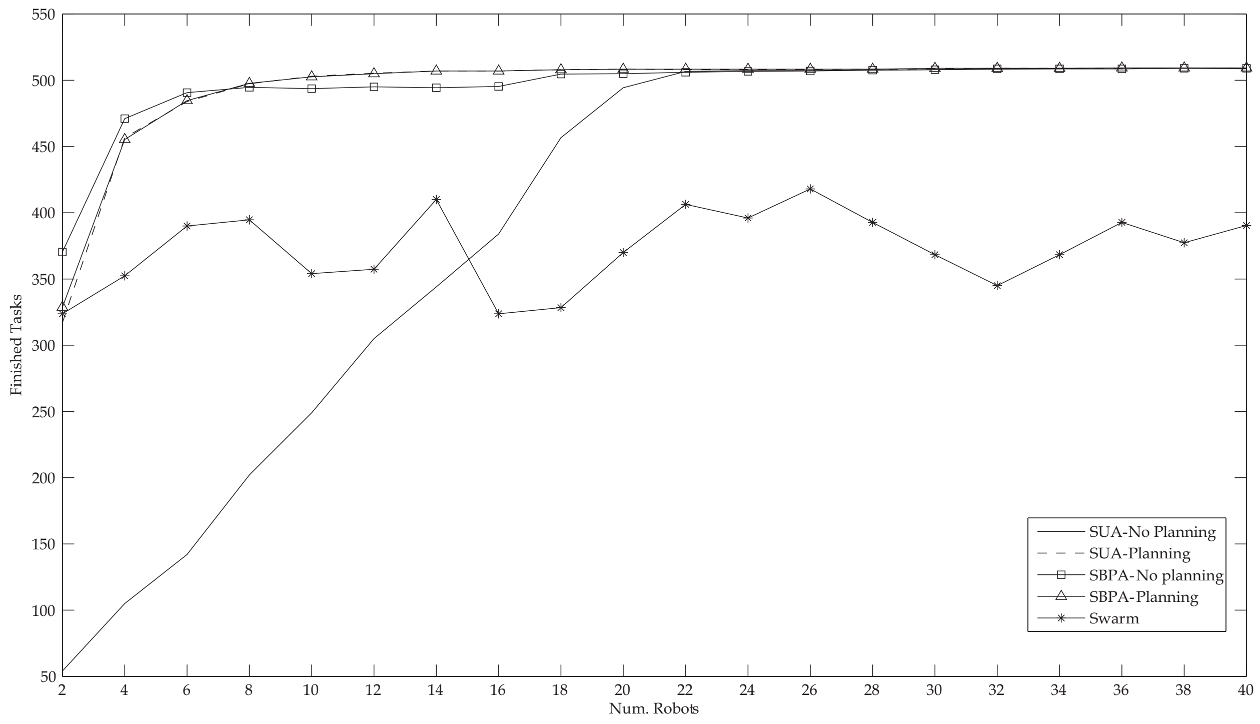


(a) Number of finished tasks.

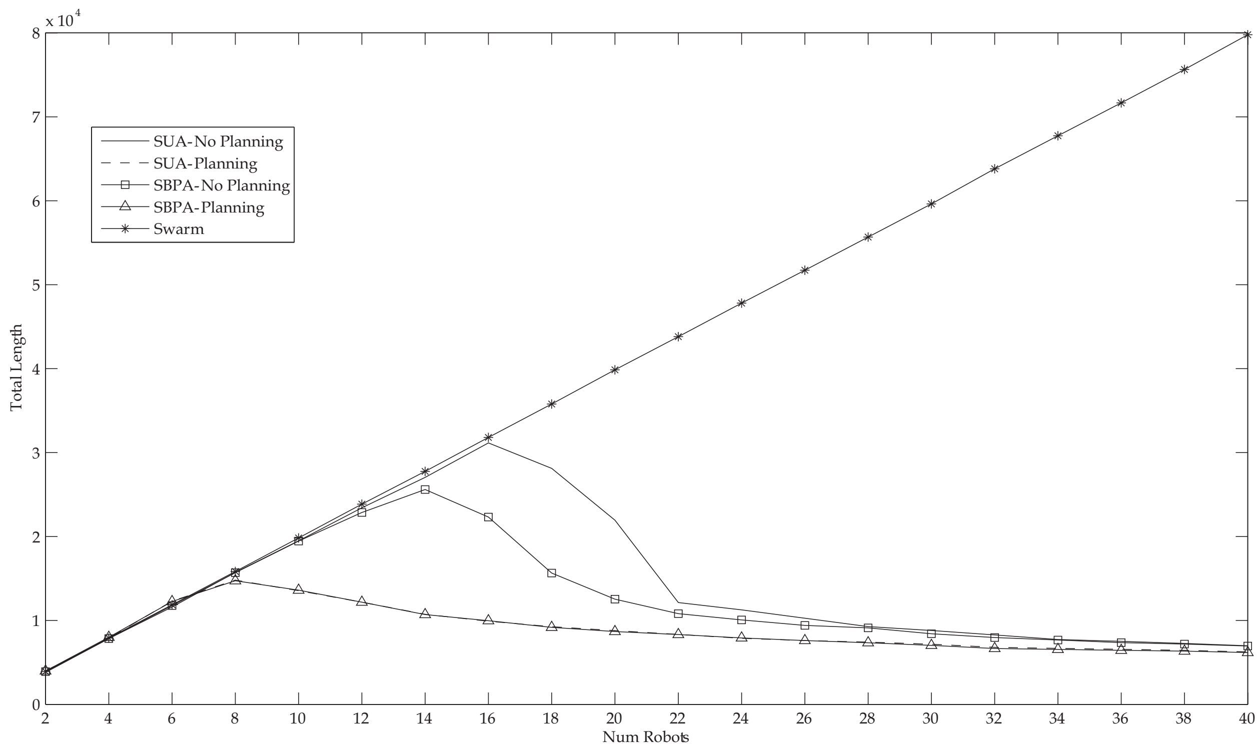


(b) Total length of the robots' path.

Fig. 1. Results without deadlines, with and without using planning. $\lambda = 0.1$



(a) Number of finished tasks.



(b) Total length of the robots' path.

Fig. 2. Total length of the robots' path without deadlines with and without using planning. $\lambda = 0.05$

strategy outperforms swarm with less robot than when λ was equal to 0.1. Besides, the number of robots from which the robots' path length starts to decrease is also lower than in figure 1(b).
The experiments carried out using tasks with deadlines (environments from 1st to 4th), show that all the strategies are very similar when the arrival ratio of tasks is low (low values of λ). Hence, we will focus on the scenarios with a high ratio of arrivals. Figure 3 shows the number of tasks executed before its deadline with $\lambda = 0.3$ and with two different set of tasks: environment 3 (random deadlines) and environment 4 (Hybrid Tasks). In both cases the SBPA without planning outperforms the planning version when the number of robots is low. In the 4th environment the number of robots from which the SBPA-planning is better than SBPA without planning is grater than in the 3th environment. This is because in the 3th environment all the tasks have a deadline, and in the 4th environment there are some tasks without time restriction. In any case, the behavior of the SUA and EDFA strategies is dramatically increased when the planning is used, therefore, in those cases the main key to improve the system is the use of a MT approach (planning).

6.3 Experiments with player/stage

In this section we will analyze in a deeper way the single task strategy (ST) when the number of robots is low. We will use the very realistic simulator Player/Stage, which will allow us to study the physical interference between robots, produced when 2 or more robots want to access to the same point of the space at the same time. The physical interference has already been studied by the authors in ST-MR-IA strategies Guerrero & Oliver (2010).
To execute our experiments we have several pioneer 3DX robots, which have become a standard research platform in robotics. The dimensions of the environment was 18mx18m and the maximum robots' velocity was 0.25m/s. The robots used this information to calculate the expected execution time and to decide if they were able to execute a task before the deadline. We used 200 objects randomly placed in each experiment and with a deadline time uniformly distributed between 12 and 70 seconds. When an object is gathered, it immediately appears another one in a random position, thus, the number of objects in the environment (*M*) is always the same. (*M*) is a parameter and its influence in the performance of the system will be tested in the experiments.
Table 2 shows the number of tasks that do not meet its deadline using the SBPA strategy, where *R* is the number of robots. In all cases, this auction algorithm is better than the swarm system, very specially when the number of robots is increased. For example, with 12 robots this benefit can be around 60%. Furthermore, the maximum distance (*D*) has not a great impact on the results, by contrast, Kalra & Martinoli (2006) showed that without deadline this parameter can be very important.

		R=4	R=8	R=12
D=9	M=5	67 (26%)	38 (38%)	18 (60%)
	M=10	90 (17%)	48 (46%)	48 (44%)
D=12	M=5	69 (39%)	35 (53%)	24 (63%)
	M=10	90 (29%)	56 (47%)	56 (47%)
D= ∞	M=5	78 (32%)	45 (46%)	24 (61%)
	M=10	87 (33%)	59 (46%)	68 (23%)

Table 2. Number of tasks that do not meet its deadline with the SBPA strategy, in brackets the percentage increase of tasks when swarm is used.

Tables 3 and 4 show the number of tasks that do not meet its deadline using SUA and EDFA, respectively. As it can be seen, in all cases the SBPA is better than both SUA and EDFA and, even in most cases, the swarm outperforms the SUA and EDFA results. The performance SUA/EDFA systems get better as the ratio between maximum distance and number of robots (R/D) increases. When there are a lot of robots with regards to the number of tasks, like for example if D = 9 and R = 4, the SUA/EDFA results are 17% worse than swarm, but with low values of $\frac{R}{D}$ the results are SUA/EDFA 14% better than swarm. Moreover, the EDFA seems to improve, in most cases the system performance, very especially for the worst SUA cases. We have to note that these results are partially similar to the already shown in figure 3 where for a low number of robots, the swarm strategy can outperform SUA and EDFA. In the RoboSim experiments the distance D was infinity, and therefore the $\frac{R}{D}$ ratio was the minimum possible.

		R=4	R=8	R=12
D=9	M=5	105 (-17%)	94 (-54%)	98(-118%)
	M=10	128 (-17%)	93 (-4%)	96 (-13%)
D=12	M=5	102 (11%)	85 (-13%)	89 (-39%)
	M=10	128 (-2%)	94 (11%)	89 (15%)
D= ∞	M=5	98 (14%)	84 (0%)	90 (-48%)
	M=10	122 (6%)	100 (9%)	88 (-13%)

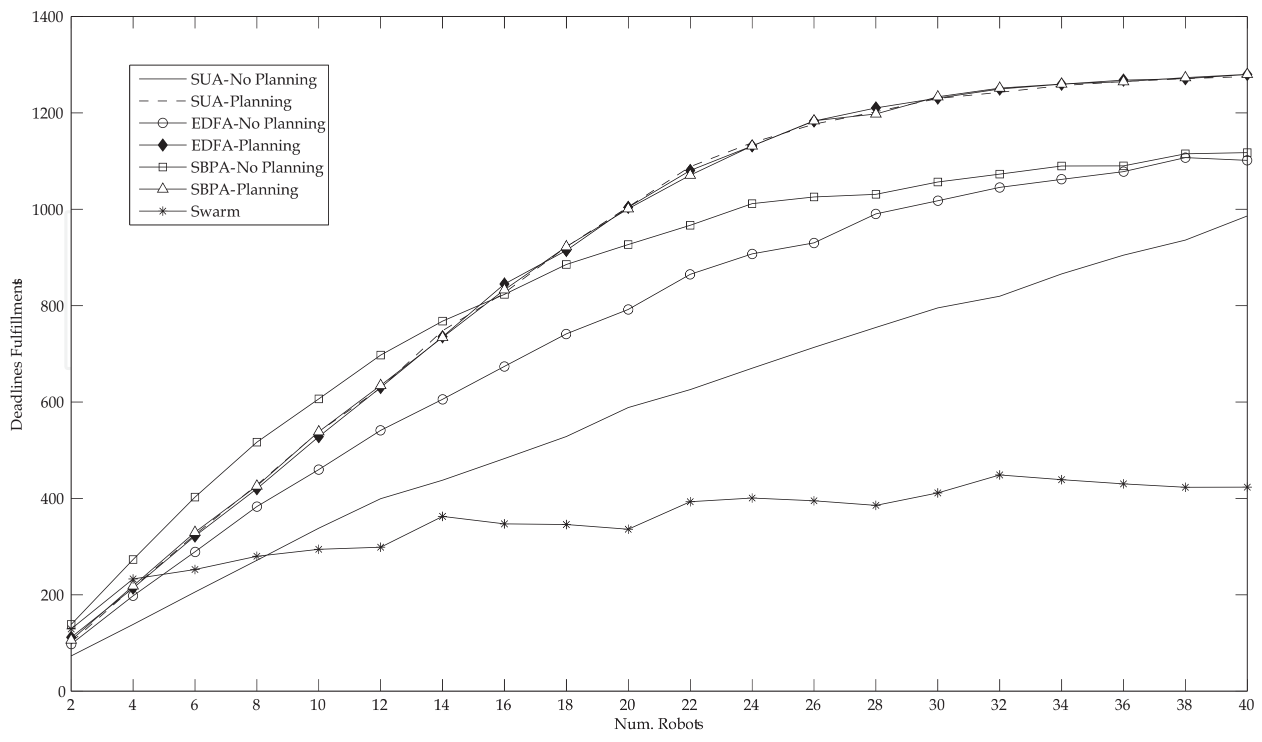
Table 3. Number of tasks that do not meet its deadline with the SUA strategy. The percentage of increase when swarm is used appears in brackets.

		R=4	R=8	R=12
D=9	M=5	99 (-10%)	87 (-43%)	78 (-73%)
	M=10	135 (-24%)	87 (2%)	90 (-6%)
D=12	M=5	97 (15%)	81 (-8%)	91 (-42%)
	M=10	126 (0%)	97 (15%)	100 (-5%)
D= ∞	M=5	96 (16%)	94 (-12%)	85 (-39%)
	M=10	132 (-2%)	94 (8%)	96 (-9%)

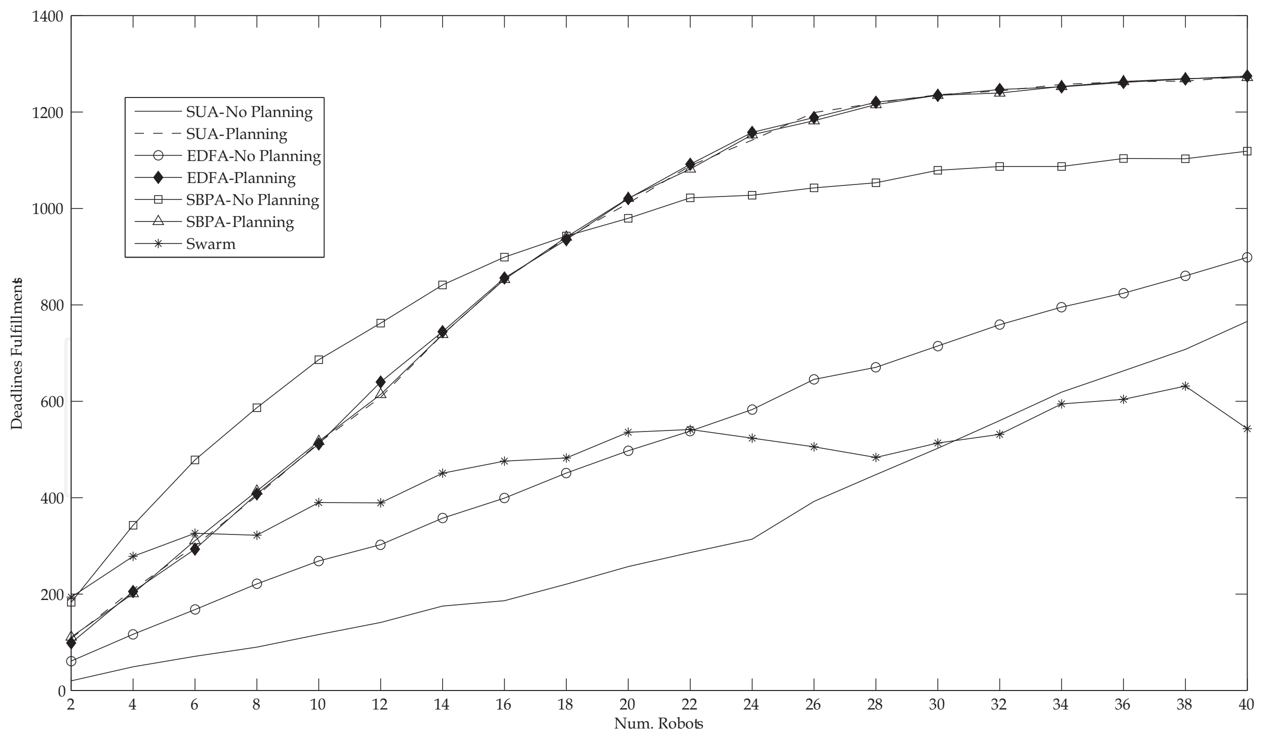
Table 4. Number of tasks that do notmeet its deadlinewith the EDFA strategy. The percentage of increase when swarm is used appears in brackets.

7. Earliest Deadline First Best Pair method (EDFBP)

In this section we will propose a new auction like algorithm called Earliest Deadline First Best Pair (EDFBP), which is in a middle way between the EDFA and the SBPA. The objective of the EDFBP algorithm is to have the SBPA's performance but with a lower algorithmic cost. To achieve this goal, EDFBP follows the algorithm 4, where firstly the central auctioneer orders all the tasks by deadline (line 1). Then, it selects a subset of this tasks (line 2) and apply only over this subset the SBPA procedure (algorithm 1). In our case the total set of tasks (T) is split in several consecutive subsets using the β parameter (see algorithm 5). Thus, β is the percentage of the total number of tasks that will be processed as one unit by SBPA. When β = 1 the EDFBP will behave as a SBPA and when β = 0 it is a EDF algorithm.



(a) 3th Environment (random deadlines)



(b) 4th Environment (hybrid tasks)

Fig. 3. Deadlines fulfillments with $\lambda = 0.3$ and different environments.

The algorithmic complexity of the EDFBP can be easily gotten as follows: let's n be the number of tasks, m the number of robots and $r = \frac{1}{\beta}$. To simplify the process we will assume that all the partitions of T have the same number of tasks, and therefore $|\ TW\ |$ always has the same value. Following the same reasoning as in section 4.2, the ordering process (line 1) has a complexity equal to $O(nlong(n))$. Then, the SBPA algorithm is performed over $\frac{n}{r}$ tasks, and therefore its complexity must be equals to $O(\sum_{i=0}^{min(\frac{n}{r},m)}(\frac{n}{r}-i)(m-i))$. Thus, the total complexity of the EDFBP algorithm is equal to: $O(nlog(n)+r\sum_{i=0}^{min(\frac{n}{r},m)}(\frac{n}{r}-i)(m-i))$. It is easy to see that the complexity of the algorithm is in general lower than the complexity of the SBPA method and the lower value of β the greater algorithm complexity is.

Algorithm 4 EDFBP algorithm

```
Require: T=List of unassigned tasks
 1: sort T by earliest deadline
 2: while T ≠ ∅ do
 3:     $TW \leftarrow tasksOfInterest(T)$
 4:     $SBPA(TW)$
 5:     $T \leftarrow T - TW$
 6: end while
```

Algorithm 5 tasksOfInterest

```
Require: T=List of unassigned tasks
 1: sort T by earliest deadline
 2: $N \leftarrow MAX(1, \lceil \beta * |T| \rceil)$
 3: $TInt \leftarrow T[0..N]$
 4: return TInt
```

Figure 4(a) shows the results of some experiments carried out to validate the EDFBP with the 3th environment, the environment with random deadlines, and with the single task (ST) approach. The β values are expressed in percentage, thus the EDFBP (50%) means the results with $\beta = 0.5$. These results show that with only a $\beta = 0.5$, that is splitting the T table in two, the results are very similar to the the SBPA strategy. This similar performances has been achieved with a much lower algorithmic cost. Figure 4(b) shows the same results but using the 1st. environment, that is the environment with all tasks with the same deadline. In this case, the results of the different values of β are more similar between them than in the last figure, but, like in the 3th environment, when $\beta = 0.5$ SBPA and EDFBP show very similar results.
Figure 5 shows the total path traveled by all the robots in the same scenario as figure 4(b). As it can be seen there are no major differences between the 4 different strategies and can also be noted that the length is practically lineal with the number of robots. If we compare this results with an environment with a low λ value (see figure 1(b) or figure 2(b)), it can be

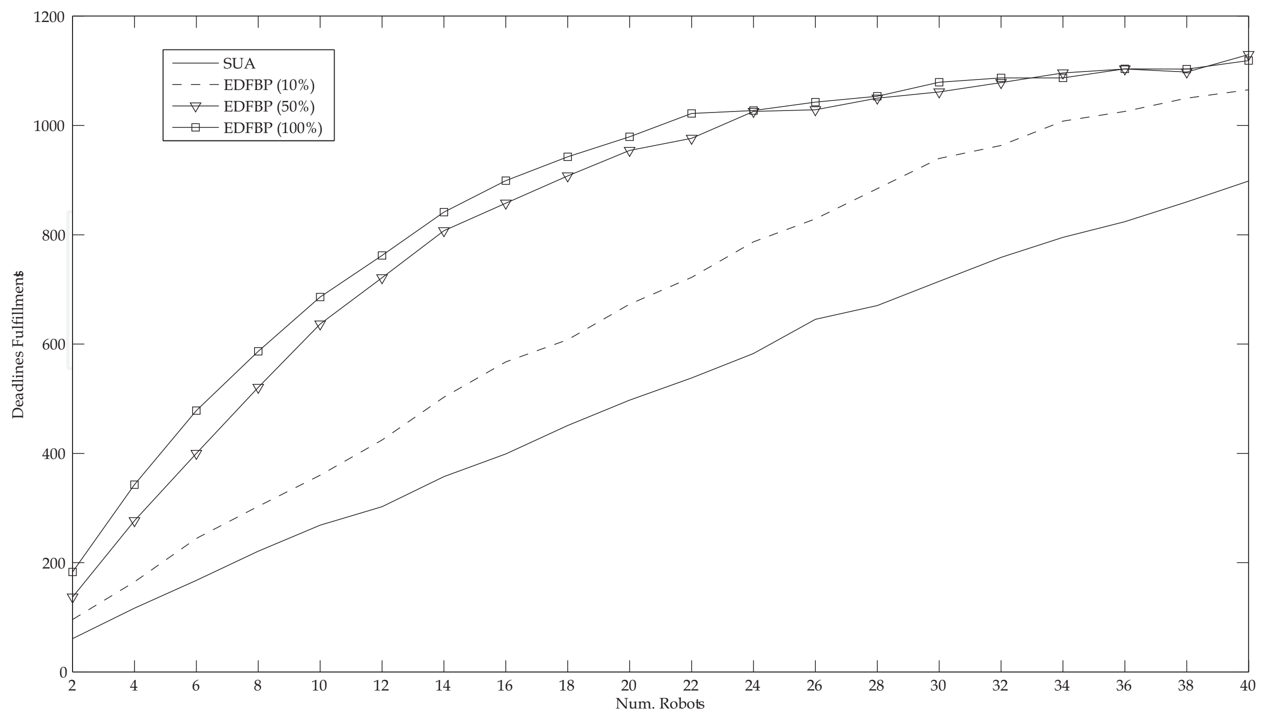
seen that now there isn't any reduction in the length with regards to the number of robots, or at least, this reduction is not relevant with less than 40 robots.

8. Conclusion and future work

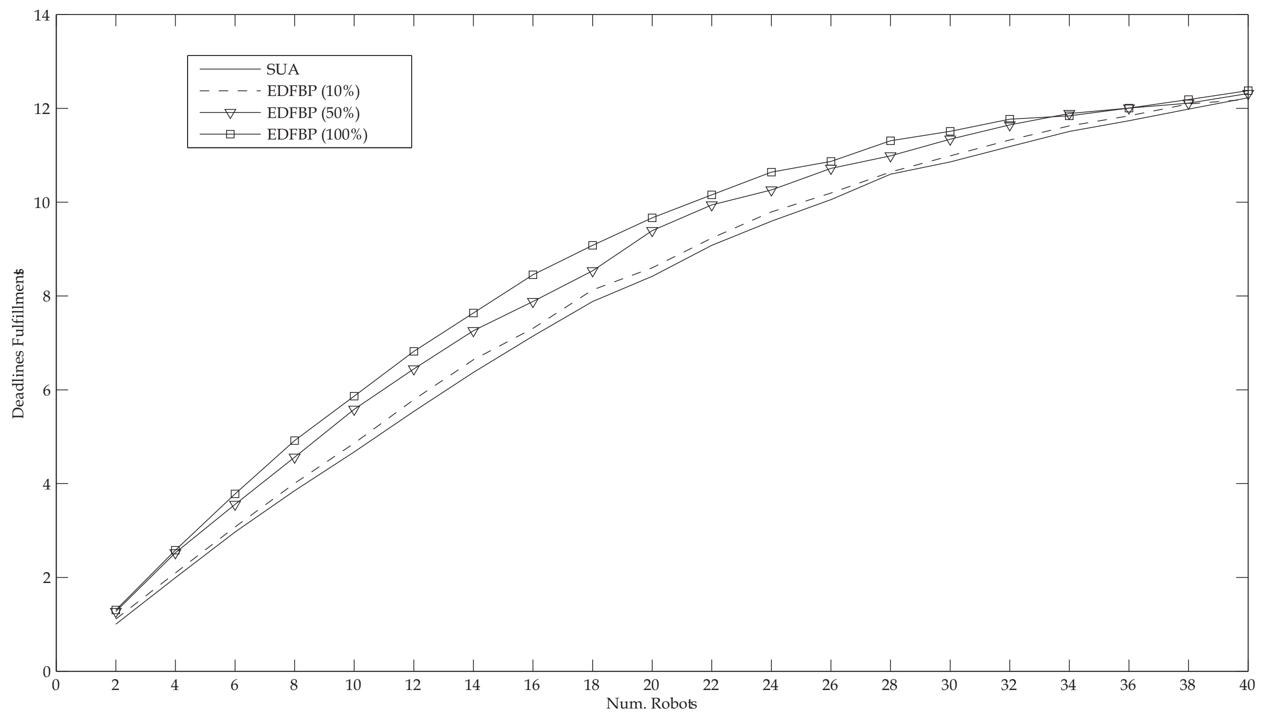
This work has studied how several different auction like algorithms and an swarm approach can carry out a set of tasks with an associated deadline. Three different auction methods has been proposed: SUA, EDFA, SBPA and a deterministic swarm approach. From the results of the experiments carried out, a forth method is also introduced, the EDFBP. The EDFBP is an hybrid between EDFA and SBPA, which, modifying a parameter β, can behave like a SUA or like a SBPA. A study of the algorithmic complexity has also been explained, the tables 5 and 1 show the order of complexity of all the methods studied. As it can be seen, the complexity of EDFBP can be fitted to be between SUA or SBPA.
Moreover, The results of the experiments show that in a single task scenario (ST), when only one task can be assigned to the same robot, the EDFA method outperforms the SUA and shows better results than a swarm approach when there are a lot of robots. In all cases the SBPA has better results than any other strategy. When the robots can have several task (MT approach) all the auction strategies have a similar performance.
The work presented has some challenging aspects to add and to improve. For the time being we are focused on finding a method to learn on-line the β parameter of the EDFBP strategy. New experiments with the Player/stage simulator using much more robots and poisson arrivals for tasks are under consideration too. This new kind of experiments will allow us a more accurate analysis about the effects of our algorithm on real robots and a better comparison with the RoboSim results. The algorithm to schedule tasks explained in section 5 is very simple, more complex and efficient methods should be tested. Some other futures improvements of our work could be: use an heterogeneous set of robots with different sensorial or computational characteristics, develop new methods to learn the expected execution time from the environment's and robot's characteristics or carry out new types of missions, for example cleaning the floor of a room or test our methods on the RoboCup Rescue simulator Kitano et al. (1999).

Swarm	$O(n)$
SUA	$O(\sum_{i=0}^{min(n,m)}(m-i))$
EDFA	$O(nlog(n)+\sum_{i=0}^{min(n,m)}(m-i))$
SBPA	$O(n+\sum_{i=0}^{min(n,m)}(n-i)(m-i))$
EDFBP	$O(nlog(n)+r\sum_{i=0}^{min(n,m)}(\frac{n}{r}-i)(m-i))$

Table 5. Complexity order of the algorithms introduced in this work in a single task environment (ST). n is the number of tasks, m is the number of robots and $r=\frac{1}{\beta}$



(a) 3th Environment (random deadline)



(b) 1th environment (all the tasks has the same deadline)

Fig. 4. Deadlines fulfillments with EDFBP $\lambda = 0.3$, different β values and different environments.

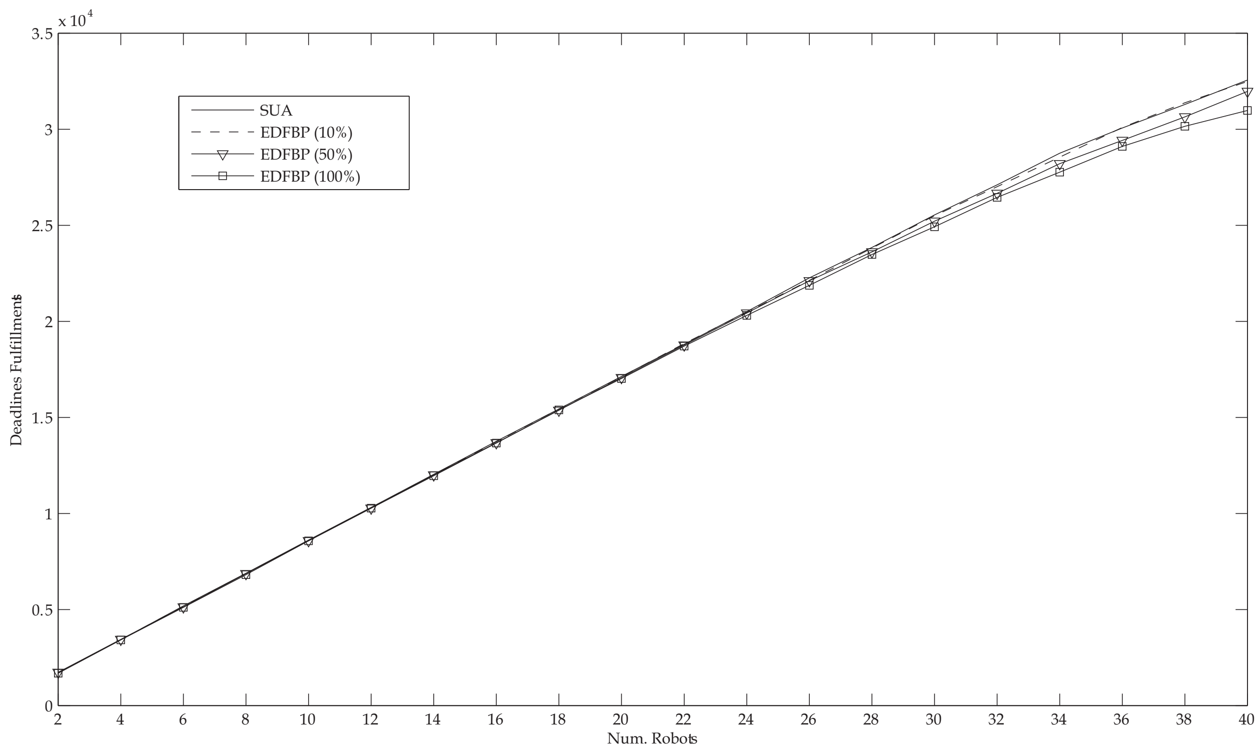


Fig. 5. Total Length with EDFBP λ = 0.3, 1th environment (all the tasks has the same deadline) and different β values.

9. Acknowledgment

This work has been partially supported by project DPI2008-06548-C03-02 and FEDER funding.

10. References

Agassounon, W. & Martinoli, A. (2002). Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems, *1st Int. Joint Conf. on Autonomous Agents and Multi-Agents Systems*, Bologna (Italy).

Bansal, N., Blum, A., Chawla, S. & Meyerson, A. (2004). Approximation algorithms for deadline-tsp and vehicle routing with time-windows, *Proc. of ACM STOC*, pp. 166–174.

Bckenhauer, H.-J., Kneis, J. & Kupke, J. (2009). Approximation hardness of deadline-tsp reoptimization, *Theoretical Computer Science* 410: 2241–2249.

Collett, T.-H., MacDonald, B. A. & Gerkey, B. P. (2005). "player 2.0: Toward a practical robot programming framework", *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, Sydney (Australia).

Dasgupta, P. & Hoeing, M. (2008). *Massively Multi-Agent Technology*, Vol. 5043/2008 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, chapter Dynamic Pricing Algorithms for task Allocation in multi-agent Swarms, pp. 64–79.

del Acebo, E. & de-la Rosa, J. L. (2008). Introducing bar systems: A class of swar intelligence optimization algorithms., *Proceedings of AISB 2008 Convention Communication, Interaction and Social Intelligence*, Aberdeen (Scotland).

Dias,M. & Stentz, A. (2003). Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination, *Technical Report CMU-RI-TR-03-19*, Carnegie Mellon University.

Ducatelle, F., Forster, A., Di-Caro, G. & Gambardella, L. (2009). Task allocation in robotic swarms: new methods and comparisons, *Technical report IDSIA-01-09*, Institute for Artificial Intelligence, Manno, Switzerland.

Ferreira, P. R., dosSantos, F. & et al. (2010). Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies, *Autonomous Agents and Multi-Agent Systems* 20: 421–443.

Gerkey, B.-P. & Mataric,M. (2002). Sold!: Auction methods formulti-robot coordination, *IEEE Transactions on robotics and Automation, Special Issue on Multi-robot Systems* 18(5): 758– 768.

Gerkey, B.-P. & Mataric, M. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems, *Intl. J. of Robotics Research* 23 (9): 939–954.

Goossens, J., Funk, S. & Baruah, S. (2002). Edf scheduling on multiprocessor platforms: some (perhaps) counterintuitive observations, *Proceedings of RealTime Computing Systems and Applications Symposium*.

Guerrero, J. & Oliver, G. (2004). Multi-robot task allocation method for heterogeneous tasks with priorities, *7th. International Symposium on Distributed Autonomous Robotic Systems*, Tolouse (France), pp. 311–317.

Guerrero, J. & Oliver, G. (2010). Amulti-robot auction method to allocate taskswith deadlines, *7th Symposium on Intelligent Autonomous Vehicles (IAV 2010)*, Lecce (Italy).

Jones, E. (2009). *Multi-Robot Coordination in Domains with Intra-path Constraints*, Phd thesis, The Robotics Institute- Carnegie Mellon University, Pittsburg, Pennsylvania (USA).

Jones, E., Dias, M. & Stentz, A. (2006). Learning-enhanced market-based task allocation for disaster response, *Technical Report CMU-RI-TR-06-48*, Carnegie Mellon University.

Kalra, N. & Martinoli, A. (2006). A comparative study of market-based and threshold-based task allocation, *8th Int. Symp. on Distributed Autonomous Robotic Systems*.

Kitano, H., Tadokoro, S., Noda, I. & et al. (1999). Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research, *Proceedings of IEEE international conference on systems, man and cybernetics (SMC)*, Vol. 6, Tokyo (Japan), pp. 739–743.

Koes, M., Nourbakhsh, I. & Sycara, K. (2005). Heterogeneous multirobot coordination with spatial and temporal constraints, *Proceedings of the Twentieth National Conference o Artificial Intelligence (AAAI)*, AAAI Press, pp. 1292–1297.

Pinedo, M.-L. (2008). *Scheduling: Theory, Algorithms, and Systems*, 3th. edition edn, Springer.

Sahin, E., Girgin, S., Bayindir, L. & Turgut, A.-E. (2008). *Swarm Intelligence: Introduction and Applications*, Natural Computing Series, Springer Verlag, Berlin, Germany, chapter Swarm Robotics.

Service, T.-C. & Adams, J.-A. (2010). Coalition formation for task allocation: theory an algorithms, *Autonomous Agents and Multi-Agent Systems* .

Vig, L. (2006). *Multi-Robot Coalition Formation*, Phd thesis, Gratuate School of Vanderbilt University.

Werger, B. B. & Mataric,M. J. (2000). Broadcast of local eligibility for multi-target observation, *Distributed Autonomous Robotic Systems 4*, Knoxville, Tennessee, USA.

Yang, Y., Zhou, C. & Tin, Y. (2009). Swar robots task allocation based on response threshold model, *Proceedings of the 4th Intl. Conference on Autonomous Robots and Agents*, Willengton (New Zeland).

Zhang, D., Xie, G., Yu, J. & Wang, L. (2007). Adaptive task assignment for multiple mobile robots via swarm intelligence approach, *Robotics and Autonomous Systems* 55(7): 572– 588.

Zheng, X. & Koenig, S. (2008). Greedy approaches for solving task-allocation problems with coalitions, *In Proceedings of the AAMAS-08 Workshop on Formal Models and Methods for Multi-Robot Systems*, Estoril (Portugal), pp. 35–40.

Zheng, X., Koenig, S. & Tovey, C. (2006). Improving sequential single-item auctions, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. pages 2238–2244.

Zlot, R. & Stentz, A. (2006). Market-based multirobot coordination for complex tasks, *Intl. J. of Robotics Research. Special Issue on the 4th Intl. Conference on Field and Service Robotics*, Vol. 25 (1).