DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch105

Task Allocation Strategy for Time-Constrained Tasks in Robots Swarms

Yara Khaluf[1] and Franz Rammig[1]

[1]Heinz Nixdorf Institut, University of Paderborn, Germany
{yara, franz}@hni.uni-paderborn.de

Abstract

Task allocation is a key problem, which has a direct influence on the system performance in all kinds of distributed systems. This paper focuses on a specific kind of task allocation in swarm robotic systems, where the tasks are associated with specific time constraints.
The paper presents a self-organized task allocation strategy, which aims to assign robot swarms to time-constrained tasks in a distributed manner. The robots assignment is performed based on particular specifications including task sizes and deadlines in addition to the specification of the single robot performance on the considered tasks. No central control is required to govern the swarm behaviour and no communication is exploited among robots.

1 Introduction

Swarm robotics is a recent field of research that takes inspiration from complex natural systems such as colonies of social insects (ants, honeybees, etc.) or groups of cooperating animals. It is a kind of mobile distributed system with a high density and which can be mainly characterized by its: redundancy, where robots failures do not affect the system functionality, scalability, since the system can be extended by adding robots, and flexibility where it can be used to perform a large spectrum of applications.
In many practical robotics applications the successful execution of a task depends not only on the logical correctness of the operations that robots are performing, but also on the time before which the results are delivered. Such tasks are referred to as real-time tasks and they are generally categorized according to their deadlines into: *hard* real-time tasks, where missing the deadline can lead to catastrophic results and *soft* real-time tasks, where missing the deadline decreases the quality of results. Real-time tasks will be common to encounter as soon as the swarm robotic systems are exported out of the research labs to be involved in real life applications. While dealing with hard-deadlines is beyond the capabilities of a fully stochastic system like swarm robotics, tasks with soft deadlines are the suitable candidates for swarm robotics. This paper discusses real-time tasks with soft deadlines, which need to be performed by a swarm of homogeneous robots. The goal is to assign the robots to the tasks under their time constraints in a fully distributed and autonomous manner. The proposed allocation strategy neglects initially the influence of physical interferences among robots, on the overall performance of the system. However the strategy can be extended later to include this influence.
The rest of the paper is organized as follows: section 2 reviews the literature of task allocation in swarm robotic systems with and without time constraints. In section 3 a formulation of the task allocation problem under time constraints is introduced. The designed allocation strategy is explained with its different stages in section 4. Section 5 presents a numerical example with its Monte-Carlo simulations to illustrate the steps of the allocation strategy and verify it. In section 6 a swarm robotic scenario is introduced where the allocation strategy is applied and simulated. The paper is concluded in section 7.

2 Related Work

Task allocation can be found in natural such as in ant and bee colonies Bonabeau et al. (1998). Mathematical models of task allocation, which focus on simple reactive mechanisms and study the fraction of robots engaged in a particular task as a function of the number of available tasks as perceived by the robots, were performed like Lerman et al. (2006). The task allocation solutions proposed in the literature can be classified in three broad categories: *centralized*, *negotiation-based* and *self-organized*: centralized techniques assume the presence of a central coordinator responsible for the allocation of the agents to the tasks. Self-organized systems, on the contrary, are constituted by peers that take decisions autonomously, with limited negotiations with other peers and without a central point of control. This kind of systems are generally less prone to catastrophic failures and considered a better approach when rapid adaptation to changes in the environment is required. Most of these studies tackle simple problems without task interdependencies, Dahl et al. (2009). Negotiation-based approaches, generally based on auction-based strategies, are the compromise solution be-

tween centralized and self-organized systems, Dias et al. (2005), Gerkey and Mataric (2002), Zheng et al. (2006). In auction-based strategies, the robots bid on the announced task according to specific task characteristics and to their relative capabilities. One of the characteristics that is often criticized in this approach, is that many negotiation-based solutions assume a fully connected network among the robots, which is not the case in many realistic applications. A comparison between the auction-based approaches and the self-organized ones based on threshold can be found in Kalra and Martinoli (2006). A general taxonomy of task allocation strategies in robotic systems has been presented in Gerkey and Mataric (2004) along three main comparisons: *single-task robots (ST) vs. multi-task robots (MT)*, where single and multiple robots are able to perform only one task at time; *single-robot tasks (SR) vs. multi-robot tasks(MR)*, where each task needs one robot or more, and the *instantaneous assignment (IA) vs. time-extended assignment (TA)* where it is assumed that the robots and the environment allow only for an instantaneous task allocation with no future planning.

In swarm robotics, response-threshold mechanisms are relatively common Nouyan et al. (2005), Nouyan et al. (2004), Ducatelle et al. (2009b), Ducatelle et al. (2009a), Krieger and Billeter (2000). In this approach, each robot is programmed to react to stimuli associated to the different tasks. Agassounon and Martinoli (2002) introduce a task allocation for the traditional swarm task "foraging". Another threshold-based algorithm for allocating workers to a given task whose demand evolves dynamically over time is presented in Agassounon et al. (2001). In Liu et al. (2007) a mathematical model for a similar task allocation behaviour is introduced. Some works combine the common swarm response-threshold approach with a kind of communication protocol to avoid the need for a central unit as in Zhang et al. (2007). A very few works to our best knowledge, have assumed a target distribution for the robots over all the available tasks to be reached like we can find in McLurkin and Yamins (2005). Few authors have studied the problem of task allocation in swarm robotic systems with time constraints associated to tasks. Some of the performed studies were based on the auction techniques for the allocation in respect to deadlines like in Guerrero and Oliver (2010) and Guerrero and Oliver (2011). Other works like Acebo and Rosa (2008), have introduced a heuristic based on the so-called Bar-System model, where the key idea is to simulate the way waitresses assign themselves to bar customers in an efficient and distributed way. The approach is then applied to a group of loading robots for a commercial harbour. In Schneider et al. (2005) and Jones et al. (2007) market-based task allocation strategies, where time is the critical constraint, are considered together with a reward mechanism associated to a task being successfully completed.

3 Problem Formulation

The problem of autonomous task allocation in presence of deadlines can be formulated as follows: a swarm of N robots should be allocated to a set of m tasks $\{T_1, \ldots, T_m\}$. The task deadlines $\{D_1, \ldots, D_m\}$ and the task sizes $\{S_1, \ldots, S_m\}$, are assumed to be known a priori. The task is assumed to be built up of individual parts, where the robot can accomplish one part per time. The size of any task represents the discrete number of parts which should be accomplished within the task deadline. Each task T_i is composed of S_i parts and accomplishing T_i is achieved by accomplishing all of its S_i parts. The real-time tasks we consider in our study have soft deadline and require to be executed in parallel. The switching costs between them are negligible in comparison to the task deadlines. The system is designed as a fully autonomous one, where no communication among robots is applied and no central unit is used for the allocation purposes.

The single robot performance on a specific task is expressed in terms of the random time required by the robot to accomplish one part of the considered task. The random variable associated with the single robot performance is modelled in this paper as a normal distributed variable with a task-specific mean and a task-specific standard deviation. The single robot performance is an essential input for the developed allocation strategy. Robots can measure their individual performances by working on each of the considered tasks for a specific period of time, registering the times they require to accomplish individual parts and estimate the mean and standard deviation related to their performance on each task. Tasks are served according to their priorities, which are derived based on the task deadlines. The task with a shorter deadline has a higher priority to be executed. Before starting the execution, a list of the m tasks with their sizes and deadlines is provided to the swarm. These task specifications in addition to the single robot performance on the different tasks are the inputs used later to perform the tasks allocation.

We concentrate in this paper on a kind of *dynamic* task allocation, where the robot is allowed to stay on the same task or to switch to another one. The switching decision could be taken each time the robot finishes working on a part of the current task or at specific time points. Switching at specific time points, during the execution times of the tasks, requires global synchronization among the robots to take the decision2 at the specified time point. Dynamic task allocation is useful to be applied in many applications, where the switching costs among tasks can be considered being negligible. Such cases can be encountered when the tasks occupy a shared physical arena, so robots do not need to travel among them while switching. An example is foraging multiple kinds of objects where each kind represents an individual task and all kinds are scattered on the same arena. Another possibility to omit switching costs, is when they are negligi-

ble in comparison to the task deadlines.
The selection of the next task is performed by means of allocation probability matrices, which are calculated based on the task specifications and the single robot performance. The probability matrices are of the form P_{ij}, which represents the probability to switch to task T_j from task T_i while the switching costs are negligible in comparison to the deadlines.

4 Robot Allocation Strategy

The goal of this study is to develop a feasible task allocation strategy that allows robots to assign themselves autonomously to a set of real-time tasks with respect to the task deadlines. In a fully stochastic system like swarm robotics where even the performance of a single robot on a specific task represents a stochastic variable, it is particularly difficult to develop an allocation strategy which can guarantee the execution of the task within its deadline.
Our strategy attempts to find an optimal number of robots to be assigned to each task based to the size and deadline of the task in addition to the performance of the single robot on that task. The required number of robots on each task is used to derive allocation probability matrices, which are used by the robots to allocate themselves autonomously to the tasks during their execution.
Each task is considered to be *"inactive"* as soon as it is completely accomplished or when its deadline is exceeded, otherwise the task is considered as *"active"*. Robots are not required to be assigned to inactive tasks. Consequently, the number of robots available for allocation changes based on the current number of active tasks. In order for the allocation strategy to exploit the current number of robots available for allocation, the time between the start of tasks execution and the largest deadline, is divided into periods: $\{\pi_1, \ldots, \pi_m\}$ where:

$$\pi_i = D_i - D_{i-1} \qquad \forall i \in \{2, \ldots, m\} \tag{1}$$

The first period has the length of the earliest deadline $\pi_1 = D_1$. Figure 1 illustrates the periods and the active tasks within each period.

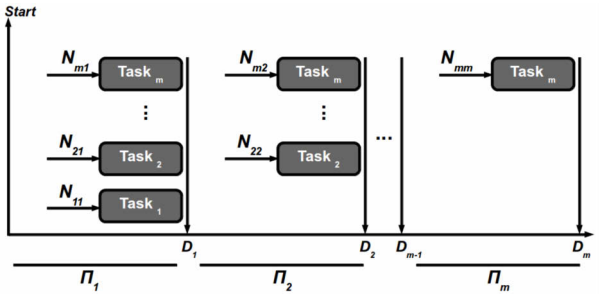

Figure 1: Active tasks over the defined periods

4.1 Required Number of Robots

The single robot performance on task T_i is expressed, as mentioned above, in terms of the time required by a single robot to accomplish one part of task T_i. This random time is modelled as a normally distributed variable with the mean μ_i and the standard deviation σ_i. Let us use k_i to denote the number of parts which could be accomplished by a single robot on task T_i within its deadline D_i. The value of k_i is taken within the discrete range $[0, +\infty[$, which represents the possible outcomes related to the number of parts could be accomplished by a single robot within D_i. Let us define the event $E_i(k_i)$ that a single robot accomplishes k_i parts on task T_i within D_i. We refer to the time spent by a single robot to accomplish k_i parts by $\tau_i(k_i)$, then we have the following two events equivalents:

$$E_i(k_i) \Longleftrightarrow \tau_i(k_i) \leqslant D_i \tag{2}$$

The probabilities of the equivalent events in Eq. (2) are equal and they represent the probabilities we are interested in:

$$\Pr(E_i(k_i)) = \Pr(\tau_i(k_i) \leqslant D_i) \tag{3}$$

As the time spent by the robot to accomplish *one* part of task T_i is normally distributed with the mean μ_i and the standard deviation σ_i, the right side of Eq. (3) is the probability that the sum of k_i random variable, each one being distributed using $Norm(\mu_i, \sigma_i)$, is smaller than or equal to D_i. It is well know that the sum of n random variable each one being distributed with $Norm(\mu, \sigma)$, is a random variable distributed normally with the mean $n\mu$ and the standard deviation $\sqrt{n}\sigma$. Consequently, the probability $\Pr(\tau_i(k_i) \leqslant D_i)$ in Eq. (3) represents the cumulative density function *CDF* of the normal distribution with the mean $k_i\mu_i$ and the standard deviation $\sqrt{k_i}\sigma_i$.

$$Pr(\tau_i(k_i) \leqslant D_i) = \frac{1}{2}[1 + erf(\frac{D_i - k_i\mu_i}{\sqrt{2k_i\sigma_i^2}})] \tag{4}$$

The allocation strategy applies the cumulative density function in Eq. (4) to find out the probability associated with the event $E_i(k_i)$ for each $k_i \in [0, S_i]$, where S_i is the size required to be accomplished on T_i within D_i. This probability $P(E_i(k_i))$ is referred to as the success probability of the event $E_i(k_i)$. The events $E_i(k_i)$ are distributed following a binomial distribution with the success probabilities $P(E_i(k_i))$, see Figure 2.
The expected value of a random variable X which is distributed according to a binomial distribution with n trials and the success probability p is given by:

$$E[X] = np \tag{5}$$

We map the number of trials n to the required number of robots, where each robot can accomplish k_i parts with a success probability $P(E_i(k_i))$. In order to find the required size

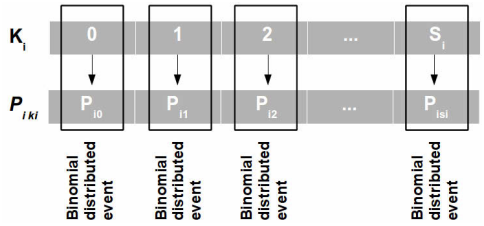

Figure 2: $E_i(k_i)$ for each $k_i \in [0, S_i]$ with the success probability associated to each of k_i values

of trials (number of robots), such that one robot of the n robots is expected to accomplished k_i parts within D_i, we substitute the expected value by 1 and the success probability by $P(E_i(k_i))$ in Eq. (5), so we have:

$$1 = n_i P(E_i(k_i)) \Longrightarrow n_i = \frac{1}{P(E_i(k_i))}$$

n_i is the number of robots required to achieve an excepted number of accomplished parts equal to k_i within D_i by one robot. However, we aim to achieve an expected number of accomplished parts equal to S_i within D_i, hence the required number of robots is calculated using the following equation:

$$N_i = \lceil \frac{n_i}{k_i} S_i \rceil \tag{6}$$

Let us consider the example of one task with the size $S_i = 10$ parts, the deadline $D_i = 10$ time units. The single robot performance on this task in terms of the time required by the single robot to accomplish one part, is normally distributed. We assume different means of the random time: $\mu_i \in \{2, 4, 6\}$ and a unique standard deviation $\sigma_i = 0.01$.
The number of parts k_i, which could be accomplished by a single robot within D_i can takes its value in $[0, 1, \ldots, +\infty[$. However the range of interest is $k_i \in \{0, 1, \ldots, 10\}$. Each event $E_i(k_i)$ of accomplishing k_i parts by a single robot is associated with a probability calculated using Equation (4). Figure 3 shows the different values of k_i with their success probabilities calculated for the different means. We consider the mean $\mu_i = 2$ for the rest of the example. Figure 4 shows how the required size of trials n_i changes with changing k_i. n_i, represents the size of the robots needed to have on average one robot accomplishing k_i parts within D_i. In the same figure we can see the total number of robots N_i required by task T_i calculated for each k_i value.

4.2 Optimal Robot Number and Allocation Probability Matrices

In the previous section the allocation strategy calculates the required number of robots N_i, which should be assigned to task T_i to accomplish S_i parts on average within the deadline D_i. There exist several possible number of robots N_i to be used. Each value of the N_i is associated with a unique value

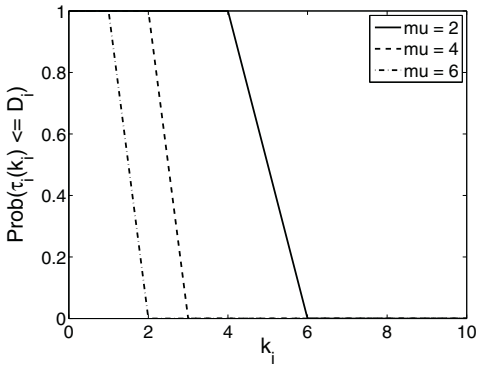

Figure 3: The relation between the number of parts could be accomplished by a single robot within the deadline and its probability of success

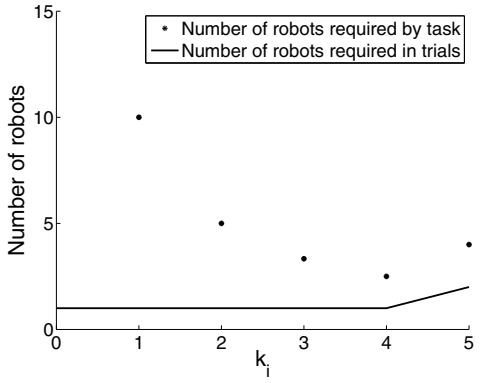

Figure 4: The relation between the number of parts k_i and the number of required robots in trials n_i and by the task N_i

of parts, k_i. The process starts by finding the success probability related to each event $E_i(k_i)$, where $k_i \in \{0, \ldots, S_i\}$. After that, the size of the trials (robots), which is required to have on average one robot accomplishing k_i parts within D_i, is determined. Finally the total number of required robots is calculated using Equation (6).
However, a robot swarm represents a limited resource with a given size, which applies a strict constraint on selecting the feasible number of robots N_i, could be assigned to each of the active tasks during period π_j:

$$\sum_{i=j}^{m} N_i \leqslant N \tag{7}$$

where N is the size of swarm used in the solution.
The developed strategy attempts to minimize the number of robots required by each task individually. Assigning the minimum of the feasible numbers of robots, reduces the impact of potential physical interactions and save robotic resources from unnecessary use. In addition, it provides a

higher chance to schedule newly arrived tasks. Hence, the allocation strategy aims to minimize, for all tasks, the objective function introduced in Eq. (6) under the constraint of Eq. (7) and consequently to find the optimal number of robots for any task T_i:

$$N_i^{opt} = min(\lceil \frac{n_i}{k_i} S_i \rceil) \tag{8}$$

The optimal number of robots is the robots number, which the allocation strategy aims to assign to each task though all the periods where this task is active. However, as the swarm size is limited to N robots, it is possible to not have enough robots in order to assign N_i^{opt} to task T_i over all its active periods. The allocation strategy starts, for each period, to satisfy the robot needs of the tasks according to their priorities, which are based on their deadlines. Lack of robots can occur on task T_i at any of its periods and this robot lack leads, in turn, to a lack in the amount of work was planned to be accomplished by the N_i^{opt} robots on task T_i within the considered period. Let us denote the missed number of robots on task T_i within the period π_j by $\delta_i(\pi_j)$. This number of robots was missed to work on task T_i within the time $\tau(\pi_j)$, which is the length of period π_j. We introduce the term of *robots lack density* to refer to the lack of robots happened over a specific period of time like $\tau(\pi_j)$. Let us denote the lack density happened on task T_i during period π_j, by $\alpha_i(\pi_j)$. This lack density can be calculated as in following:

$$\alpha_i(\pi_j) = \delta_i(\pi_j)\tau(\pi_j) \tag{9}$$

The lack density associated with all robot lacks, which happened on task T_i up to period π_j is denoted by $\alpha_i^{\pi_j}$ and represents the sum of the lack densities over all the task periods up to π_j:

$$\alpha_i^{\pi_j} = \sum_{k=1}^{j-1} \delta_i(\pi_k)\tau(\pi_k) \tag{10}$$

The goal now, while finding out the required number of robots to be assigned to task T_i during period π_j, is to cover the robot lacks happened on task T_i up to period π_j. The needed number of robots to cover the lacks that happened on T_i up to period π_j, should be scaled based on the length of π_j. The sum of lack densities associated with the previous periods is used to find out the number of robots, $\delta_i^{\pi_j}$, required to cover the lacks of the previous periods:

$$\delta_i^{\pi_j} = \lceil \frac{\alpha_i^{\pi_j}}{\tau(\pi_j)} \rceil \tag{11}$$

Finally, the allocation strategy can calculate the number of robots required to be assigned to task T_i during period π_j like follows:

$$N_i(\pi_j) = \begin{cases} N_i^{opt} + \delta_i^{\pi_j} & \text{if } N_{current} \geqslant N_i^{opt} + \delta_i^{\pi_j} \\ N_{current} & \text{if } N_{current} < N_i^{opt} + \delta_i^{\pi_j} \end{cases} \tag{12}$$

After calculating the number of robots to assign to each task over the periods where the task is active, the allocation strategy outputs a set of probability matrices associated with the defined periods. These matrices are used for the dynamic allocation of the robots. Robots use the probability matrix of the current period to allocate themselves to the tasks: each time a robot finishes working on one part of the current task, or at the beginning of current period all robots allocate themselves to the active tasks using the matrix of the period. This dynamic allocation is a self-organized process, as no central unit controls the robots for assignment. The allocation probability $P_i(\pi_j)$ of task T_i in period π_j will be then calculated using the following equation:

$$P_i(\pi_j) = \frac{N_i(\pi_j)}{\sum_{k=j}^{m} N_i(\pi_k)} \tag{13}$$

where $P_i(\pi_j)$ is the probability to switch from any task to task T_i in period π_j.

5 Numerical Example

We introduce in this section a numerical example to illustrate the mechanism of the developed allocation strategy. Let us assume to have a set of 5 tasks $\{T_1, T_2, T_3, T_4, T_5\}$ with the soft deadlines $\{30, 90, 150, 250, 500\}$ and the sizes $\{1000, 2000, 3000, 4000, 5000\}$. A homogeneous swarm of $N = 400$ is used to execute the tasks, where the performance of any individual robot belonging to this swarm on each task is normally distributed with a task-specific mean μ_i and a task-specific standard deviation σ_i. For our example the means of the single robots performance on the 5 tasks are as following $\{2, 3, 4, 2, 6\}$ and the standard deviations are $\{0.2, 0.3, 0.1, 0.01, 0.5\}$.

The possible outcomes in terms of task parts, which could be accomplished by a single robot within the deadline on the different tasks, are: $k_1 \in \{0, 1, \ldots, 1000\}$, $k_2 \in \{0, 1, \ldots, 2000\}$, $k_3 \in \{0, 1, \ldots, 3000\}$, $k_4 \in \{0, 1, \ldots, 4000\}$, $k_5 \in \{0, 1, \ldots, 5000\}$. Each event E_{k_i} of accomplishing k_i parts by a single robot on task T_i is associated with a success probability, that is calculated using Equation (4).

The allocation strategy selects the minimum number of robots to be assigned to each of the considered tasks during the periods where the tasks are active:

$$T_1 = 72 \quad T_2 = 72 \quad T_3 = 82 \quad T_4 = 33 \quad T_5 = 62$$

The sum of the required robots numbers is verified against the constraint in (7):

$$72 + 72 + 82 + 33 + 62 \leqslant 400$$

The lack of robots may occur on any of the 5 tasks is 0 over all periods, as the swarm size is large enough to cover their

robot needs. Hence, the required number of robots to be assigned to task T_i on any of its periods is given by:

$$N_i(\pi_j) = N_i^{opt}$$

The probability to apply by each robot for assigning itself to task T_i during period π_j is given by:

$$P_i(\pi_j) = \frac{N_i(\pi_j)}{\sum_{k=j}^{m} N_i(\pi_k)} = \frac{N_i^{opt}}{\sum_{k=j}^{m} N_k^{opt}}$$

Thus, the probability matrices over the 5 periods are as in following:

Period π_1:

$$\begin{pmatrix} P_1(\pi_1) & P_2(\pi_1) & P_3(\pi_1) & P_4(\pi_1) & P_5(\pi_1) \\ 72/321 & 72/321 & 82/321 & 33/321 & 62/321 \end{pmatrix}$$

Period π_2:

$$\begin{pmatrix} P_1(\pi_2) & P_2(\pi_2) & P_3(\pi_2) & P_4(\pi_2) & P_5(\pi_2) \\ 0 & 72/249 & 82/249 & 33/249 & 62/249 \end{pmatrix}$$

Period π_3:

$$\begin{pmatrix} P_1(\pi_3) & P_2(\pi_3) & P_3(\pi_3) & P_4(\pi_3) & P_5(\pi_3) \\ 0 & 0 & 82/177 & 33/177 & 62/177 \end{pmatrix}$$

Period π_4:

$$\begin{pmatrix} P_1(\pi_4) & P_2(\pi_4) & P_3(\pi_4) & P_4(\pi_4) & P_5(\pi_4) \\ 0 & 0 & 0 & 33/95 & 62/95 \end{pmatrix}$$

Period π_5:

$$\begin{pmatrix} P_1(\pi_5) & P_2(\pi_5) & P_3(\pi_5) & P_4(\pi_5) & P_5(\pi_5) \\ 0 & 0 & 0 & 0 & 62/62 \end{pmatrix}$$

We simulate the behaviours of the robots using a Monte-Carlo simulation which was repeated 500 times for the tasks specified in the example above. We assume a global synchronization among the robots. Thus, the allocation probability matrices calculated above, are used by robots each at the beginning of its related period to allocate themselves to the tasks during the whole period. Figure 5 shows a comparison between the average of the total number of parts accomplished by the swarm on each of the 5 tasks and the task size.

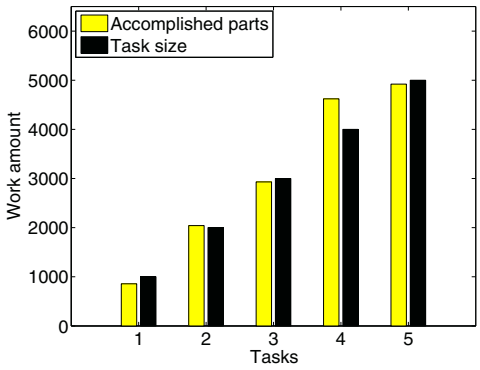

Figure 5: A comparison between the average of accomplished parts resulted from 500 repeat of Monte-Carlo simulation with the task sizes

6 Robotic Scenario

In this section we introduce a multi-task scenario where a homogeneous swarm of simple robots is used to work on a multi line production system. In the considered system, different kinds of objects, which are located initially in objects repositories, Figure 6, should be transported to their production areas. We assume 3 arenas of different sizes, where the robot swarm is used to accomplish the transportation tasks on the different arenas. Each arena is associated with a task of transporting a specific number of objects from their repositories to their production area within a specific deadline. The task sizes are: $\{70, 90, 110\}$ objects and the task deadlines are: $\{30000, 50000, 90000\}$ time units, where each time unit in our simulation represents $1/10$ of a second. The total size of the used swarm is $N = 20$ robots and the distance between the objects repository and the production area is for the left arena: 12 meters, for the middle arena 17 meters and for the right arena 23 meters. The robots are scattered initially in a robot reposi-

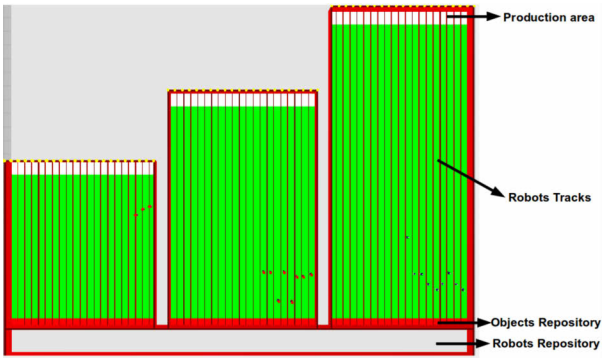

Figure 6: Multi-production scenario

tory and as soon as the task execution starts, robots use the designed probability matrices to allocated themselves to the different tasks. They start to transport the objects between their repositories and their production areas moving on separated tracks, where each track can be used by only one robot at a time. Applying the track system reduces the physical interferences between robots and allows to consider them as negligible. The only interferences present are those between the robot and its track boarders and among robots while using the robot repository area to pass to another working areas. The production areas are marked with lights to attract the robots towards them, while transporting the objects. The robots apply a *light_attraction* behaviour combined with an *obstacle_avoidance* while transporting the objects to their production areas and a *light_repulsion* behaviour combined with *obstacle_avoidance* while travelling

to fetch the objects. The simulator, [1]ARGoS Pinciroli et al. (2012), has been used to simulate the scenario, to measure the single robot performance and to calculate the average of the swarm performance on the 3 considered tasks. The single robot performance is modelled, like mentioned above, as a normally distributed random variable with a task-specific mean and a task-specific standard deviation. In our scenario, the single robot performance was measured via repeated high-level simulations in ARGoS in order to characterize the average time required by the single robot to transport one object on each of the 3 tasks. Figure 7 shows the probability density function associated with the single robot performance on each task. The measured means and standard deviations of this performance are as in following: $\mu = \{2238, 3297.7, 4482\}, \sigma = \{427.44, 576.69, 1071.9\}$. The allocation strategy finds the optimal numbers to be assigned to each of the 3 tasks within their deadlines following the steps explained in section 4. The optimal number of robots to be assigned to the tasks are as following:

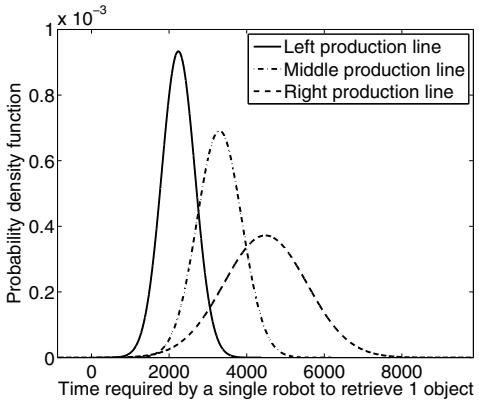

Figure 7: The probability distribution of the single robot performance on the 3 considered tasks

$$N_1 = 6 \qquad N_2 = 7 \qquad N_3 = 7$$

The sum of the required robots is equal to the swarm size $N = 20$ robots, thus the need of each task will be fulfilled over all the periods where the task is active. The allocation probability matrices are calculated following Equation (13) and are as in following:

Period π_1:

$$\begin{pmatrix} P_1(\pi_1) & P_2(\pi_1) & P_3(\pi_1) \\ 6/20 & 7/20 & 7/20 \end{pmatrix}$$

Period π_2:

$$\begin{pmatrix} P_1(\pi_2) & P_2(\pi_2) & P_3(\pi_2) \\ 0 & 7/14 & 7/14 \end{pmatrix}$$

Period π_3:

$$\begin{pmatrix} P_1(\pi_3) & P_2(\pi_3) & P_3(\pi_3) \\ 0 & 0 & 7/7 \end{pmatrix}$$

In this robotic example, we assume no global synchronization among the robots. Thus, we allow each robot to use the probability matrix related to the current period, each time it finishes transporting one object in order to select its next task. The fraction of robots, which is not needed in the current period, is kept idle in the robots repository.

We repeat the simulation for 10 times before calculating the average number of transported objects on each task. Figure 8 shows the comparison between the average numbers of accomplished parts on the tasks and their sizes. The small difference we can notice in Figure 8 between the number of transported parts and the number of parts required to be transported on task T_1 is based on the differences of the inter-intervals of robots decisions. As robots are allowed to select their tasks each time they finish transporting one object, so the time point of the decision is based on the mean μ_i and the standard deviation σ_i of the single robot performance on the considered task T_i. Hence, robots working on a specific task may be faster in taking their switching decisions than others working on other tasks, which risks keeping the robots fractions as required on all tasks over time. This is one of the weak points of allowing a dynamic switching decision each time the robot accomplishes one part of its current task. However, its effect is strongly related to the differences among the performances of single robot on the different tasks. In addition, it is necessary to use this kind of dynamic decisions, when no synchronization is available among robots to synchronize their decision points with the beginning of each period.

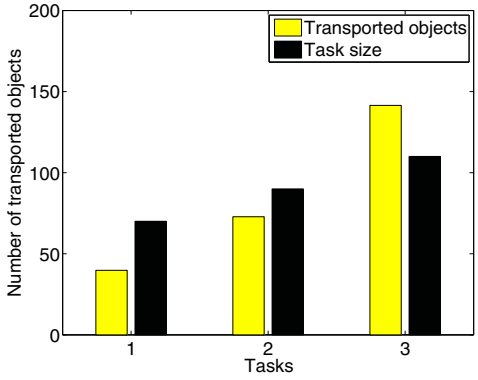

Figure 8: The Comparison between the average number of transported objects on each task and the task size

7 Conclusion

In this paper, we have introduced a novel task allocation strategy for swarm robotic systems in context of real-time tasks with soft deadlines. The developed strategy is a fully-

[1]ARGoS is a discrete-time physics-based simulation framework developed within the Swarmanoid project. It can simulate various robots at different levels of details, as well as a large set of sensors and actuators

autonomous one, that uses the tasks sizes, deadlines and the single robot performance on the considered tasks to output a set of allocation probability matrices. The resulting matrices are used by the robots, independently through the execution, to allocate themselves to the different tasks with the goal of executing them within their deadlines. The considered swarm is a homogeneous one, where no communication is exploited among robots. The developed allocation strategy is a dynamic one, where robots are allowed to switch among the tasks during their execution times. This kind of dynamic allocation offers several advantages including: the possibility to cope with non-predicted lack in performance and the ability to consider on-line arrival of tasks. However it requires to assume negligible switching costs among the considered tasks. A numerical example of the allocation strategy in addition to a robotic scenario were introduced, where the allocation probability matrices where derived to be used by individual robots in a set of simulations to verify the desired swarm performance. In the future work, the impact of physical interferences on the performance on the single robot could be taken into account while estimating this performance. Considering the influence, the physical interferences has on the performance, leads to more accurate analysis of the allocation.

References

Acebo, E. D. and Rosa, J. L. D. L. (2008). Introducing bar systems: A class of swarm intelligence optimization algorithms.

Agassounon, W. and Martinoli, A. (2002). Efficiency and Robustness of Threshold-Based Distributed Allocation Algorithms in Multi-Agent Systems. In *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems AAMAS-02*, pages 1090–1097.

Agassounon, W., Martinoli, A., and Goodman, R. (2001). A scalable, distributed algorithm for allocating workers in embedded systems. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 5, pages 3367 –3373 vol.5.

Bonabeau, E., Sobkowski, A., Theraulaz, G., and Deneubourg, J.-L. (1998). Adaptive task allocation inspired by a model of division of labor in social insects.

Dahl, T. S., Matarić, M., and Sukhatme, G. S. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robot. Auton. Syst.*, 57(6-7):674–687.

Dias, M. B., Zlot, R. M., Kalra, N., and Stentz, A. (2005). Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Pittsburgh, PA.

Ducatelle, F., Förster, A., Caro, G. A. D., and Gambardella, L. M. (2009a). New task allocation methods for robotic swarms. In *In 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions*.

Ducatelle, F., Förster, A., Caro, G. A. D., and Gambardella, L. M. (2009b). Task allocation in robotic swarms: new methods and comparisons. Technical report.

Gerkey, B. and Mataric, M. J. (September 2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotic Research*.

Gerkey, B. P. and Mataric, M. J. (2002). Sold!: Auction methods for multirobot coordination.

Guerrero, J. and Oliver, G. (2010). A multi-robot auction method to allocate tasks with deadlines.

Guerrero, J. and Oliver, G. (2011). Auction and swarm multi-robot task allocation algorithms in real time scenarios. In Yasuda, T., editor, *Multi-Robot Systems, Trends and Development*, pages 437–456. InTech.

Jones, E., Dias, M. B., and Stentz, A. (2007). Learning-enhanced market-based task allocation for oversubscribed domains. In *International Conference on Intelligent Robots and Systems, 2007. IROS 2007.*

Kalra, N. and Martinoli, A. (2006). A Comparative Study of Market-Based and Threshold-Based Task Allocation. In *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*.

Krieger, M. J. and Billeter, J.-B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots.

Lerman, K., Jones, C., Galstyan, A., and Mataríc, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. *Int. J. Rob. Res.*, 25(3):225–241.

Liu, W., Winfield, A. F. T., Sa, J., Chen, J., Dou, L., Liu, W., Winfield, A. F. T., Sa, J., Chen, J., and Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots.

McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA.

Nouyan, S., Ghizzioli, R., Birattari, M., and Dorigo, M. (2004). An ant-based algorithm for the dynamic task allocation problem. Technical report, IRIDIA, Universite Libre de Bruxelles.

Nouyan, S., Ghizzioli, R., Birattari, M., and Dorigo, M. (2005). An insect-based algorithm for the dynamic task allocation problem. *KI*, 19(4):25–31.

Pinciroli, C., Trianni, V., OGrady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G., Ducatelle, F., Birattari, M., Gambardella, L., and Dorigo, M. (2012). Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6:271–295.

Schneider, J., Apfelbaum, D., Bagnell, D., and Simmons, R. (2005). Learning opportunity costs in multi-robot market based planners. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*.

Zhang, D., Xie, G., Yu, J., and 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., and Tovey, C. (2006). Improving sequential single-item auctions. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*.