# A Swarm Robotics Approach to Task Allocation under Soft Deadlines and Negligible Switching Costs

Yara Khaluf[1], Mauro Birattari[2], and Heiko Hamann[1]

[1] Department of Computer Science, University of Paderborn
Zukunftsmeile 1, 33102 Paderborn, Germany
{yara,heiko.hamann}@uni-paderborn.de
http://upb.de/cs/si
[2] IRIDIA, Université Libre de Bruxelles
B-1050 Brussels, Belgium
mbiro@ulb.ac.be
http://iridia.ulb.ac.be/

**Abstract.** Developing swarm robotics systems for real-time applications is a challenging mission. Task deadlines are among the kind of constraints which characterize a large set of real applications. This paper focuses on devising and analyzing a task allocation strategy that allows swarm robotics systems to execute tasks characterized by soft deadlines and to minimize the costs associated with missing the task deadlines.

**Keywords:** Soft deadlines, Time-constrained tasks, Swarm robotics, Multi-agent systems.

## 1 Introduction

Considering the large number of real-world applications where swarm robotics represents an efficient system to be investigated, real-time tasks represent a remarkable category of these applications. Deadlines are often enough associated with real-world tasks, such as data gathering in large sensor networks (data-mules), recycling systems and pollution cleaning. A task is referred to as real-time task, when the correctness of the task results is not related only to the logical correctness, but also to the time at which these results are delivered. The point in time before which the real-time task should be executed is referred to as the task *deadline*. There are two main types of deadlines, as categorized in traditional real-time systems [20]. *Hard deadlines* are deadlines, that when missed, can lead to catastrophic consequences. *Soft deadlines* are deadlines, that affect the quality of the results, if not met. Thus, missing this type of deadlines is associated with specific costs. This paper focuses on soft-deadline tasks which can be considered as suitable tasks to be executed by a stochastic system such as a swarm of robots. Also natural systems need to cope with related task allocation problems but rather based on priorities than explicit deadlines. For example feeding the larvae of honey bees requires appropriate task allocation. Larvae have different hunger levels and stimulate nurse bees that react with different visiting rates [10]. In the following, a task allocation strategy is developed to allow the robots to assign themselves autonomously to tasks and to execute them in parallel under the considerations of their soft deadlines. The developed strategy is tested using the biologically inspired *foraging* task. To the best of our knowledge, there are only few works which did concentrate on task allocation in swarm robotics under time constrains. The auction strategy is one of the most frequently used, see [7,6]. In addition to the well established auction strategy, heuristics are introduced in the context of robot allocation to tasks with time constraints as in [1]. In [19] and [11], market-based task allocation strategies are introduced, where time is the main critical constraint. This is considered together with a reward mechanism associated to tasks which are successfully completed. Mathematical modeling is not intensively investigated in the context of swarm robotics, still some examples can be found in [16,8,15,2,14].

In this paper we present a novel approach based on mathematical modeling to analyze the performance of swarm robotics and the behavior of individual robots. The goal is to develop an appropriate task allocation under the constraints of the different tasks. The rest of the paper is organized as follows: Section 2 describes the problem on which we are focusing. Section 3 presents the mathematical models used in analyzing the swarm performance in addition to the behavior of individual robots. The developed allocation strategy is presented in details and verified using Monte-Carlo simulations in Section 4. In Section 5 a physics-based simulation is established using the robot simulator ARGoS to verify the allocation strategy and the paper is concluded in Section 6.

## 2 Problem Description

In this paper we focus on the problem of assigning a swarm of $N$ homogeneous robots to a set of tasks which are characterized by their soft deadlines. A task deadline is the point in time up to which the task should be executed and it is referred to as *soft* when missing it is not catastrophic, but associated with specific costs.

Consider a set of $M$ soft-deadline tasks $\{T_1, T_2, \ldots, T_M\}$ which need to be executed in parallel. Each of these tasks consists of discrete sub-tasks referred to as the task *parts*, where a single part requires one robot to execute it. Executing task $T_i$ is achieved when a specific number $S_i$ of its parts is accomplished. The robots' cooperation is necessary for any task execution, as the number of parts required to be accomplished within the task deadline, is not achievable using one robot. The parts of the different tasks are regenerated periodically, therefore after the execution of a part, a new part will replace it. The switching costs among the tasks, which represent the costs associated with the time required by a robot to stop the execution of one task and to start to work on another one, are assumed to be negligible. This condition is verified when the tasks are located on the same physical area or when the time required to travel among the different tasks is negligible in comparison to the task execution times, see [12].

Let us denote the number of parts that has been accomplished on task $T_i$ up to the deadline $D_i$ by $\rho_i(D_i)$. The costs associated with missing the deadline of task $T_i$ are directly related to the part of the task which remains unprocessed at its deadline. Therefore, we can define the cost function $\zeta_i$ as the difference between the number of parts required to be achieved $S_i$ and the number of parts $\rho_i(D_i)$ executed up to the deadline $D_i$

$$\zeta_i = S_i - \rho_i(D_i). \tag{1}$$

The goal, as mentioned above, is to design an allocation strategy that allows the robots to assign themselves autonomously to the different tasks under their time constraints (deadlines). This technique is efficient when switching costs between tasks are negligible. Robots select their tasks autonomously using a *decision matrix*, that defines the robots' behaviors probabilistically. It holds the transition probabilities $p_{i,j}$ to switch from task $T_i$ to task $T_j$. Controlling the behavior of the individual robots probabilistically is a common approach in stochastic systems such as swarm robotics. The decision matrices are designed to minimize the costs associated with missing the task deadlines.

## 3 System Modeling

In this section we present the techniques used to model both the performance of the robots swarm and the behavior of the individual robot. These techniques are exploited later for the development of the allocation strategy.

### 3.1 Swarm Performance Modeling

We define the swarm performance as the amount of work (task parts) accomplished by the swarm within the deadline of the task. Based on our task specification, the amount of work accomplished on any task within a specific time duration represents a discrete random variable. The swarm performance, as defined, is the sum of the individual contributions of all robots that worked on the task up to its deadline. The total number of completed parts increases over time as the robots succeed in accomplishing their individual parts.

The process associated with the evolution of the amount of work accomplished on task $T_i$ over time up to its deadline $D_i$ represents a time-continuous stochastic process $X(t)$. This process is modeled as a Poisson process [18] with a task-specific rate $\lambda_i$ for task $T_i$. The Poisson process is, by definition, a stochastic process that counts the events within a specific time period. By mapping the occurrence of an event to the completion of an individual task part, the Poisson process represents an appropriate modeling technique for the work progress on any of the considered tasks over time. When the amount of work that is accomplished on any task over time is modeled using a Poisson process, the swarm performance $\rho_i$ obtained within the deadline $D_i$ will follow the Poisson distribution with the parameter $\lambda_i D_i$

$$\rho_i \sim Poisson(\lambda_i D_i). \tag{2}$$

In general, when the robots finish executing some task, they become available to be assigned to other tasks. As the tasks we are focusing on in this paper are time-constrained tasks, the deadline of the task represents the point in time at which the task should be accomplished and the assigned robots become available again. Therefore, we divide the execution time of the tasks into periods called activation periods. The task stays active over all the periods which are included in its deadline. The $i$-th period is defined as

$$\eta_i = D_i - D_{i-1} \qquad \forall i \in \{2, \dots, M\} \tag{3}$$

where $\eta_1 = D_1$.

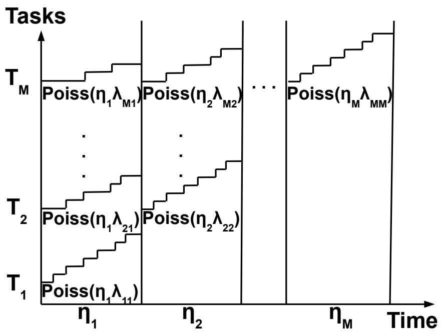

Tasks
$T_M$
$T_2$
$T_1$
Poiss($\eta_1\lambda_{M1}$)
Poiss($\eta_2\lambda_{M2}$)
Poiss($\eta_M\lambda_{MM}$)
Poiss($\eta_1\lambda_{21}$)
Poiss($\eta_2\lambda_{22}$)
Poiss($\eta_1\lambda_{11}$)
$\eta_1$
$\eta_2$
$\eta_M$
Time

**Fig. 1.** The Poisson processes modeling the progress of the accomplished amount of work on different tasks over their different activation periods

As the number of robots working on a task may change during its activation periods, the rate of accomplishing work of that task may change consequently. Therefore, individual Poisson processes are used to model the work progress on a specific task over its different activation periods. The evolution of work accomplished on task $T_i$ during the $j$th activation period is modeled as a Poisson process with the rate $\lambda_{ij}$. This random amount of work follows the Poisson distribution with the parameter $\lambda_{ij}\eta_j$.

Furthermore, it is well-known that the sum of Poisson processes is a Poisson process with the rate equal to the sum of the rates associated with the summed up processes. Therefore, the Poisson process modeling the evolution of work accomplished on task $T_i$ until its deadline $D_i$ has the rate $\lambda_i = \sum_{j=1}^{i} \lambda_{ij}$. The random amount of work that was achieved until the deadline $D_i$ of task $T_i$ is distributed following the Poisson distribution with the parameter $\sum_{j=1}^{i} \lambda_{ij}\eta_j$

Hence we have

$$\rho_i \sim \mathcal{P}oisson(\sum_{j=1}^{i} \lambda_{ij}\eta_j) \qquad \forall i \in \{1, \dots, M\}. \tag{4}$$

The expected value of a random variable which follows a Poisson distribution with the parameter $\lambda$ is equal to $\lambda$. Hence based on Eq.(4), the expected value of the performance that was achieved until the deadline $D_i$ is given by

$$\mathbb{E}(\rho_i(D_i)) = \sum_{j=1}^{i} \lambda_{ij}\eta_j .$$

However, the performance required to be achieved at the deadline $D_i$ is $S_i$ parts. Consequently, the expected value of the cost function defined in Eq.(1) can be written as

$$\begin{aligned}\mathbb{E}(\zeta_i) &= S_i - \mathbb{E}(\rho_i(D_i)) \\ &= S_i - \sum_{j=1}^{i} \lambda_{ij}\eta_j \\ &= \begin{cases} S_i - \sum_{j=1}^{i} \lambda_{ij}\eta_j & \text{when } \sum_{j=1}^{i} \lambda_{ij}\eta_j < S_i \\ 0 & \text{when } \sum_{j=1}^{i} \lambda_{ij}\eta_j \geqslant S_i . \end{cases}\end{aligned} \tag{5}$$

As we can notice in Eq.(5), no costs are associated with tasks on which more parts than their sizes are accomplished.

## 3.2 Individual Robot Modeling

The behavior of the individual robots can be described as follows. Each robot selects one of the tasks to work on and each time it finishes executing one part, it has the possibility to switch to another task or to continue on the same task.

The tasks are mapped to the states of a Markov chain and each robot is modeled as an individual process with the above described behavior over the $M$ states of the chain. The robot continues to work on the task it has selected for a random time, namely, the time required to accomplish one part of the current task. This represents a random time with a task-specific mean denoted by $\hat{\mu_i}$, which is assumed to be easily estimated over short-term experiments as it is performed in Section 5. After executing one part of the current task, the robot chooses its next task to visit where the choice is made among the available tasks including its current one. A specific probability matrix referred to as the *decision matrix* is used by the robots to select autonomously their next tasks. The described process associated with each robot, represents by definition a *semi-Markov* process [18], which has an invariant (limiting) probability measure, $\pi_i$, that can be obtained by solving the following system

$$\pi_i = \sum_{j=1}^{M} \pi_j p_{j,i} \qquad \text{where} \sum_{i=1}^{M} \pi_i = 1. \tag{6}$$

$\pi_i$ represents the proportion of transitions that take the robots into task $T_i$. The proportion of the time that the robot spends working on task $T_i$ in comparison to its total working time is given by Eq.(7). For time-constrained tasks, we are interested in the time spent by the robot on task $T_i$ within the deadline $D_i$. Let us denote this time by $\tau_i(D_i)$. Based on Eq.(7), $\tau_i(D_i)$ can be obtained by Eq.(8).

$$\tau_i = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} \quad (7) \qquad \tau_i(D_i) = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} D_i \quad (8)$$

When a swarm of $N$ robots is used to execute $M$ tasks and each single robot is modeled as a semi-Markov process with the above described behavior, the total time spent on task $T_i$ up to its deadline $D_i$ can be calculated using Eq.(9). Consequently, the number of times $n_i(N, D_i)$ that $T_i$ is expected to be visited by a swarm of $N$ robots within its deadline $D_i$, is calculated using Eq.(10).

$$\tau_i(N, D_i) = \frac{\pi_i \hat{\mu}_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} D_i N \quad (9) \qquad n_i(N, D_i) = \frac{\pi_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} D_i N \quad (10)$$

The rate of the visits to task $T_i$ within its deadline $D_i$ by the swarm of $N$ robots, which represents the number of the parts expected to be processed within the task deadline, is obtained by dividing Eq.(10) by the task deadline yielding

$$\lambda_i = \frac{\pi_i}{\sum_{j=1}^{M} \pi_j \hat{\mu}_j} N. \quad (11)$$

## 4 Task Allocation Strategy

The expected value of the cost function associated with the swarm performance was calculated in Section 3.1 using Eq.(5)

$$\mathbb{E}(\zeta_i) = \begin{cases} S_i - \sum_{j=1}^{i} \lambda_{ij} \eta_j & \text{for } \sum_{j=1}^{i} \lambda_{ij} \eta_j < S_i \\ 0 & \text{for } \sum_{j=1}^{i} \lambda_{ij} \eta_j \geqslant S_i. \end{cases}$$

The rate $\lambda_{ij}$ of the Poisson process in the $j$-th activation period of task $T_i$, can be written in terms of the transition probabilities based on Eq.(11)

$$\lambda_{ij} = \frac{\pi_{ij}}{\sum_{k=j}^{M} \pi_{kj} \hat{\mu}_k} N \qquad \forall i \in \{1, \ldots, M\}, \forall j \in \{1, \ldots, M\}.$$

So the expected value of the cost function associated with task $T_i$ can be written in terms of the transition probabilities, as

$$\mathbb{E}(\zeta_i) = \begin{cases} S_i - \sum_{j=1}^{i} \frac{\pi_{ij}}{\sum_{k=j}^{M} \pi_{kj} \hat{\mu}_k} N \eta_j & \text{for } \sum_{j=1}^{i} \frac{\pi_{ij}}{\sum_{k=j}^{M} \pi_{kj} \hat{\mu}_k} N \eta_j < S_i \\ 0 & \text{for } \sum_{j=1}^{i} \frac{\pi_{ij}}{\sum_{k=j}^{M} \pi_{kj} \hat{\mu}_k} N \eta_j \geqslant S_i \end{cases} \quad (12)$$

where $i \in \{1, \ldots, M\}$.

The developed allocation strategy aims to minimize the costs associated with the part left unprocessed at the task deadline. Minimizing the costs is required for all tasks based on their priorities. The task priority is related, in our case, to the tightness of the task deadline. Therefore, the task with an earlier deadline or a larger size has a higher priority. A simple way to define the priority of task $T_i$ is given by

$$\beta_i = \frac{S_i/D_i}{\sum_{j=1}^{M} S_j/D_j}. \tag{13}$$

The sum of the task priorities is always 1.

The minimization problem, that we consider, represents a multi-objective optimization problem with $M$ objective functions.

$$\underset{\pi_{ij}}{\text{minimize}} \begin{cases} \mathbb{E}(\zeta_1) = f(\pi_{11}) \\ \mathbb{E}(\zeta_2) = f(\pi_{21}, \pi_{22}) \\ \vdots \\ \mathbb{E}(\zeta_M) = f(\pi_{M1}, \pi_{M2}, \ldots, \pi_{MM}) \end{cases} \tag{14}$$

It is solved using the well-known methodology of scalarizing the multi-objective optimization problem and formulating a single-objective optimization problem. We sum the $M$ objective functions, where each is weighted by the priority of its task and the obtained objective function is

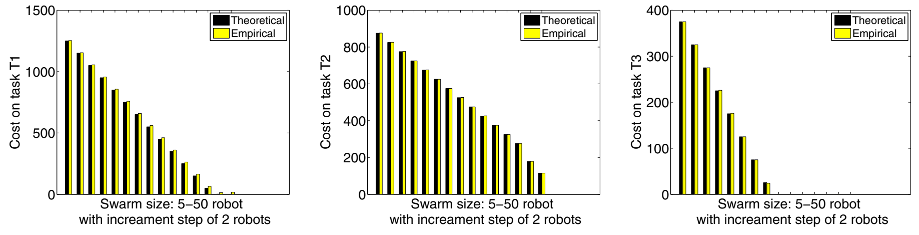
$$\underset{\pi_{ij}}{\text{minimize}} \sum_{i=1}^{M} \beta_i \mathbb{E}(\zeta_i). \tag{15}$$

The described optimization problem is solved *off-line* and the robots are provided with the resulting decision matrices (one per activation period) which hold the transition probabilities between tasks. After that, each robot can start to use the decision matrix of the current period, independently, to assign itself and switch among the different tasks.

Let us consider an example of 3 tasks which are characterized by their soft deadlines $\{500, 750, 1000\}$ time units and their sizes $\{1500, 1000, 500\}$ parts. The size of the swarm, used to execute these tasks, varies over the range $N \in [5, 50]$ robots with an increment step of 2 robots. The task priorities are calculated using Eq.(13). The optimization problem is solved for the different swarm sizes. Hence, the decision matrices are obtained and then used to predict and simulate the costs of missing the deadlines. Figure 2 shows the value of the cost function over all examined sizes of the swarm. The cost is estimated using Eq.(12), after that it is averaged over 100 runs of Monte-Carlo simulation. The figure shows a high level of consistency between the calculated cost and the simulated one.

## 5 Physics-based Simulation

In this section we consider the task of *multi-foraging* to verify the developed allocation strategy through performing physics-based simulations. Foraging behavior, in its simple form, is the behavior of exploring the environment searching



**(a)** Task $T_1$ **(b)** Task $T_2$ **(c)** Task $T_3$

**Fig. 2.** Cost function theoretically calculated based on Eq.(12) and compared with its value averaged over 100 runs of Monte-Carlo simulation

for food items and retrieving them to a safe area referred to as the nest. It can be observed in a large number of social insect colonies and animal societies [5]. Foraging was intensively investigated in swarm robotics systems [13,4,15,9]. Multi-foraging refers to the task of retrieving different types of items.

For our simulations we use the robot simulator *ARGoS*[1], where a homogeneous swarm of foot-bot[2] robots is simulated. We consider three types of items 500 *red* parts, 300 *black* parts and 150 *white* parts, which are uniformly scattered. The nest is marked with a set of lights which can be sensed by the working robots. The robots apply the *diffusion* behavior [3] combined with obstacle avoidance to maximize the exploration of the arena. The robot's motion is governed by a *light-repulsion* behavior to move out of the nest towards the arena. As soon as it finds some part to retrieve, it starts to apply a *light-attraction* behavior to move towards the nest. Both light-attraction and light-repulsion behaviors are combined with obstacle avoidance. As the three types of items are uniformly scattered on the arena, the switching costs among the tasks can be considered as negligible. We use a homogeneous swarm with a varying size between $[5, 50]$ robots, with an increment step of 5 robots. The time required by a single robot to retrieve one part of each of the three types is averaged over 10 runs of *ARGoS* simulations each for a duration of 500 seconds. In our scenario, the average of this time differs for the different types of items due to the different number of parts available of each type. The average of the time required by a single robot to retrieve one part increases by increasing the swarm size due to the spatial interferences among the robots. We use the average measured when the largest swarm is used (50 robots). This design decision is based on performing a worst case estimation of the swarm performance and consequently of the cost function. The average time of retrieving one part of each of the three items is respectively $\{98.5, 130, 197\}$. The soft deadlines associated with the tasks are: $\{500, 1500, 3000\}$ seconds and the task sizes are $\{150, 300, 500\}$ parts. The de-

[1] ARGoS is a state-of-the-art, open source $3D$ robot simulator. Its design allows for the simulation of large homogeneous and heterogeneous swarms of robots [17].

[2] Foot-bot is a wheeled robot with 17 cm diameter $\times$ 29 cm hight and weights 1.8 kg. It is equipped with a set of sensors and actuators in addition to an on-board CPU.

cision matrices are calculated, as described in Section 4, to minimize the cost associated with missing any of the task deadlines. Figure 3 shows how the cost calculated using Eq.(12) agrees with the one averaged over 20 runs of ARGoS simulations, when the designed decision matrices are applied in both cases. In addition it shows how both calculated and simulated costs decrease while increasing the size of the swarm.

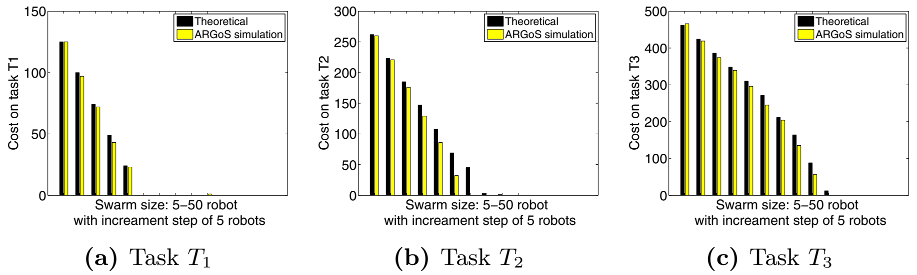


**Fig. 3.** Cost function theoretically calculated based on Eq.(12) and compared with its value averaged over 20 runs of *ARGoS* simulation

## 6 Conclusion

This paper focuses on developing a task allocation approach that allows a swarm of robots to execute a set of tasks under the consideration of soft deadlines. The resulting behavior of the robots can adapt itself efficiently to dynamic changes such as changes in the swarm size (e.g. robots failures). The probabilistic design of this behavior allows for a higher level of self-adaptivity than for example the one obtained by assigning fixed-size groups of robots to the tasks. Addressing robot-to-robot communication as a part of our future work may increase significantly the self-adaptivity of the system, as exchanging knowledge about dynamic changes in the environment, tasks or swarm, will be possible. In addition, a part of our future work is the estimation of the mean time $\hat{\mu_i}$ as a function of the number of robots working on task $T_i$.

## References

1. Acebo, E.D., de la Rosa, J.: Introducing bar systems: A class of swarm intelligence optimization algorithms. In: AISB Convention Communication, Interaction and Social Intelligence, Aberdeen, Scotland, pp. 18–23 (2008)
2. Agassounon, W., Martinoli, A.: A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. In: Proceedings of the IEEE Conference on System, Man and Cybernetics (SMC), Hammamet, Tunisia, pp. 250–255 (2002)

3. Beal, J.: Superdiffusive dispersion and mixing of swarms with reactive Levy walks. In: IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2013, pp. 141–148 (2013)
4. Campo, A., Dorigo, M.: Efficient multi-foraging in swarm robotics. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 696–705. Springer, Heidelberg (2007)
5. Danchin, E., Giraldeau, L., Cézilly, F., et al.: Behavioural ecology. Oxford University Press, Oxford (2008)
6. Guerrero, J., Oliver, G.: Multi-robot task allocation method for heterogeneous tasks with priorities. In: Distributed Autonomous Robotic Systems 6, pp. 181–190. Springer, Berlin (2007)
7. Guerrero, J., Oliver, G.: Auction and swarm multi-robot task allocation algorithms in real time scenarios. In: Yasuda, T. (ed.) Multi-Robot Systems, Trends and Development, pp. 437–456. InTech (2011)
8. Hamann, H.: Space-Time Continuous Models of Swarm Robotics Systems: Supporting Global-to-Local Programming. PhD thesis, University of Karlsruhe, Germany (November 2008)
9. Hamann, H., Wörn, H.: An analytical and spatial model of foraging in a swarm of robots. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) Swarm Robotics Ws. LNCS, vol. 4433, pp. 43–55. Springer, Heidelberg (2007)
10. Huang, Z., Otis, G.: Inspection and feeding of larvae by worker honey bees (hymenoptera: Apidae): Effect of starvation and food quantity. Journal of Insect Behavior 4(3), 305–317 (1991)
11. Jones, E., Dias, M., Stentz, A.: Learning-enhanced market-based task allocation for oversubscribed domains. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp. 2308–2313 (2007)
12. Khaluf, Y.: Task Allocation in Robot Swarms for Time-Constrained Tasks. PhD thesis, University of Paderborn, Germany (2014)
13. Labella, T., Dorigo, M., Deneubourg, J.: Division of labor in a group of robots inspired by ants' foraging behavior. ACM Transactions on Autonomous and Adaptive Systems 1(1), 4–25 (2006)
14. Lerman, K., Galstyan, A.: Mathematical model of foraging in a group of robots: Effect of interference. Autonomous Robots 13(2), 127–141 (2002)
15. Lerman, K., Galstyan, A., Martinoli, A., Ijspeert, A.: A macroscopic analytical model of collaboration in distributed robotic systems. Artificial Life 7(4), 375–393 (2001)
16. Lerman, K., Martinoli, A., Galstyan, A.: A review of probabilistic macroscopic models for swarm robotic systems. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 143–152. Springer, Heidelberg (2005)
17. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G., Ducatelle, F., Birattari, M., Gambardella, L., Dorigo, M.: ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence 6, 271–295 (2012)
18. Ross, S.: Applied probability models with optimization applications. Dover Publications Inc., New York (1992)
19. Schneider, J., Apfelbaum, D., Bagnell, D., Simmons, R.: Learning opportunity costs in multi-robot market based planners. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, pp. 1151–1156 (2005)
20. Stankovic, J.: Deadline scheduling for real-time systems: EDF and related algorithms. Springer (1998)