Decentralized Multi-tasks Distribution in Heterogeneous Robot Teams by Means of Ant Colony Optimization and Learning Automata

Javier de Lope[1,2], Dario Maravall[1], and Yadira Quiiionez[1]

[1] Computational Cognitive Robotics Group
Dept. Artificial Intelligence
Universidad Politecnica de Madrid
[2] Dept. Applied Intelligent Systems
Universidad Politecnica de Madrid
Javier.delopeOupm.es, **dmaravall@fi.upm.**es, ay.**quinonezQalumnos.upm.es**

Abstract. This paper focuses on the general problem of coordinating multiple robots. More specifically, it addresses the self-election of heterogeneous specialized tasks by autonomous robots. In this paper we focus on a specifically distributed or decentralized approach as we are particularly interested on decentralized solution where the robots themselves autonomously and in an individual manner, are responsible of selecting a particular task so that all the existing tasks are optimally distributed and executed. In this regard, we have established an experimental scenario to solve the corresponding multi-tasks distribution problem and we propose a solution using two different approaches by applying Ant Colony Optimization-based deterministic algorithms as well as Learning Automata-based probabilistic algorithms. We have evaluated the robustness of the algorithm, perturbing the number of pending loads to simulate the robot's error in estimating the real number of pending tasks and also the dynamic generation of loads through time. The paper ends with a critical discussion of experimental results.

Keywords: Multi-robot Systems, Stochastic Learning Automata, Ant Colony Optimization, Multi-tasks Distribution, Self-Coordination of Multiple Robots, Reinforcement Learning, Multi-Heterogeneous Specialized Tasks Distribution.

1 Introduction

In multi-robots systems, optimal task/job allocation or assignment is an active research problem [1], in which several central or global allocation methods have been proposed [2,3]. Some authors have also introduced decentralized or autonomous solutions, in particular inspired in the social labor division observed in some species of social insects [4,5].

In this work we take a specifically distributed or decentralized approach as we are particularly interested in experimenting with truly autonomous and decentralized techniques in which the robots themselves are responsible of choosing a particular task in an autonomous and individual manner. Under this approach we can speak of multitasks selection instead of multitasks allocation, as the agents or robots select the tasks instead of being assigned a task by a central controller.

We have already experimented with two different techniques. First, we applied the well-known threshold models inspired in the labor division of social insects [6]. Second, we employed stochastic reinforcement learning algorithms based on Learning Automata theory [7]. In this paper we also employ stochastic reinforcement learning algorithms as well as ants colony optimization-based deterministic algorithms as explained in the sequel.

Summarizing, this work focuses on the general problem of coordinating multiple robots, we propose a solution using two different approaches by applying Ant Colony Optimization-based deterministic algorithms as well as Learning Automata-based probabilistic algorithms to solve the corresponding multi-tasks distribution problem. We have considered several experiments to evaluate the system performance index for both approaches, and the results obtained are shown in article. This paper is structured as follows: section 2 describes the formal description of the problem and experimental scenario. Section 3 presents a brief introduction, basic definitions and stochastic reinforcement algoritms about learning automata methods. Section 4 briefly describes ant colony optimization methods. Section 5 describes experimental results of the evaluation of performance index, the conclusions and further work are presented at Section 6.

2 Formal Definitions

2.1 Formal Description of the Problem

The optimal multi-task allocation problem in multi-robot systems can be formally defined as follows: "Given a robot team formed by *N* heterogeneous robots, and given *K* different types of heterogeneous specialized tasks or equivalently, given *K* different robots roles or robots jobs and given a particular time-dependent load or number of tasks to be executed *L* = *{h(t), h(t),* •••, *li((t)}* obtain an optimal distribution of the *K* tasks among the *N* robots in such a way that the robots themselves, autonomously and in an individual manner, select a particular task such that all the existing tasks are optimally executed".

Let *L* = {/i(t),/2(t),...,//<-(£)} be the different specialized tasks. Each *lj* G *L* has a number of *j* sub-tasks or pending loads. Let *R* = {ri, *ri,* •••, rw} be the set of *N* heterogeneous mobile robots. To solve the problem, we have supposed that all members *R* = {ri,r2, ...*rj^}* are able to participate in any sub-task *lj.*

2.2 Experimental Scenario

We have established the following experimental scenario (Fig. 1) in order to analyze a particular strategy or solution for the coordination of multi-robot systems as regards the optimal distribution of the existing tasks. Given a set of

N heterogeneous mobile robots in a region, achieving an optimal distribution for different types of tasks. The set of *N* robots will form sub-teams for each type of task *lj*. The sub-teams are dynamic over time, i.e. the same robots will not be always part of the same sub-team, but the components of each sub-team can vary depending on the situation.

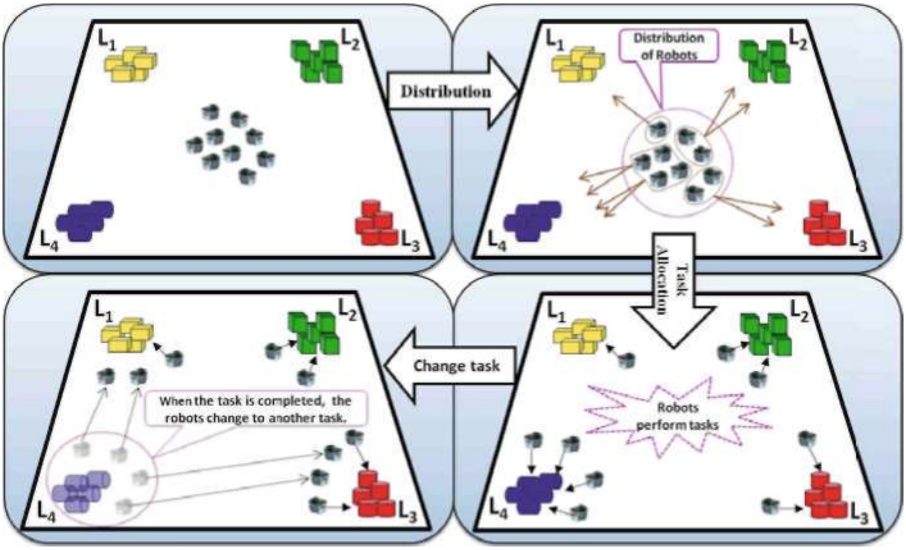


Fig. 1. Experimental scenario

Most of the proposed solutions in the technical literature are of a centralized nature, in the sense that an external controller is in charge of distributing the tasks among the robots by means of conventional optimization methods and based on global information about the system state [8]. However, we are mainly interested on truly decentralized solutions in which the robots themselves, autonomously and in an individual and local manner, select a particular task so that all the tasks are optimally distributed and executed. In this regard, we have experimented with stochastic reinforcement learning algorithms based on Learning Automata theory to tackle this hard self-coordination problem as described in the sequel.

3 Learning Automata Methods

3.1 A Brief Introduction

Learning automata have made a significant impact and have attracted a considerable interest in last years [9]. The first researches on learning automata models were developed in Mathematical Psychology, that describe the use of stochastic automata with updating of action probabilities which results in reduction in the number of states in comparison with deterministic automata. They can be applied to a broad range of modeling and control problems, control of manufacturing plants, pattern recognition, path planning for manipulator, among other. An important point to note is that the decisions must be made with very little knowledge concerning of the environment, to guarantee robust behavior without the complete knowledge of the system. In a purely mathematical context, the goal of a learning system is the optimization of a function not known explicitly [10].

Learning is defined as any permanent change in behavior as a result of past experience, and an automata is a machine or control mechanism designed to automatically follow a predetermined sequence of operations or respond to encoded instructions [11]. The objective of stochastic learning automata is to determine how the choice of the action at any stage should be guided by past actions and responses, so when a specific action is performed the environment provides a random response which is either favorable or unfavorable [12].

3.2 Basic Definitions

A learning automaton is a sextuple < *x, Q, u, P(t), G, 1Z* >, where *x* is the finite set of inputs, *Q* = {</i, <⁄2, • • •, q_m} is a finite set of internal states, *u* is the set of outputs, *P(t)* = *{pi(t),p2(t),* • • • *,*$p_m(t)$} is the state probability vector at time instant *t*, *G : Q* —> *u* is the output function (normally considered as deterministic and one-one), and *1Z* is an algorithm called the reinforcement scheme, which generates *P(t* + 1) from *P(t)* and the particular input at a discrete instant *t*.

The automaton operates in a random environment and chooses its current state according to the input received from the environment. The new state probabilities distribution *P(t* + 1) reflects the information obtained from the environment. The random environment has a set of inputs *u* and its set of outputs is frequently binary {0,1}, with '0' corresponding to the reward response and '1' to the penalty response. If the input to the environment is *ui* the environment produces a penalty response with probability c$.

Fig. 2 shows the feedback configuration of a learning automaton operating in a random environment. At each instant *t* the environment evaluates the action of the automaton by either a penalty '1' or reward '0'. The performance of the automaton's behaviors is the average penalty

$$\mathbf{J(t) = \overset{\text{-.}}{-}]\overset{m}{\underset{\mathit{i=l}}{[}}>(t)_{c\,j}} \tag{1}$$

which must be minimized. In order to minimize the expectation of penalty (1), the reinforcement scheme modifies the state probability vector *P*. The basic idea is to increase *pi* if state <j generates a reward and to decrease *pi* when the same state has produced a penalty. A great number of reinforcement schemes for minimizing the expected value of penalty have been studied and compared. One

of the most serious difficulties that arise in learning automata is the dichotomy between learning speed and accuracy. If the speed of convergence is increased in any particular reinforcement scheme, this action is almost invariably accompanied by an increase of convergence to the undesired state [13,14].

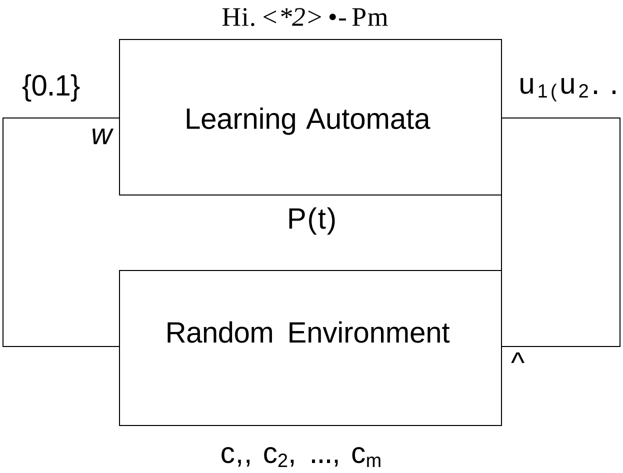


Fig. 2. Interaction of learning automaton with random environment

3.3 Stochastic Reinforcement Algorithms in Learning Automata Theory

In the technical literature a widely used stochastic reinforcement algorithms is *LR-I,* which stands for Linear Reward-Inaction algorithm.

Let us suppose that the action chosen by the automaton at instant *t* is </>$, for the *LR-I* the updating of the action probabilities is as follows:

$$Pi(t+l)={}_{Pi}(t)\ +\ \backslash p(t)[l-Pi(t)] \qquad (2)$$

$$_{Pj}(t+l)={}_{Pj}(t)\text{-}\backslash p(t)P3(t) \qquad VJ\pm iA<3<N \qquad (3)$$

where 0 < A < 1 is the learning rate and *(3(t)* is the environment's response: *(3* = 1 (favorable response or reward) or *(3* = 0 (unfavorable response or penalty in which case the algorithm do not change the probability, i.e. inaction).

Let's suppose that there are *K* different specialized tasks, then we designate by *Pij(t),* the probability at instant *t* that robot r-j selects task *lj* these probabilities hold:

$$0 <Pij(t) < 1;\ \ \overset{N}{\underset{i=i}{\wedge 2}}pij(t) = 1;\ \ i = 1,2, ...,N \text{ robots};\ \ j = 1,2,\ldots ,K \text{ tasks} \qquad (4)$$

Initially, without previous robot's experience these probabilities are initialized at the "indifference" position as follow:

$$-Pij'(O) ^{=} \frac{\sim 7?}{K} f^{or} * = 1, 2,..., N \textit{ robots and } j = 1, 2, .. ., K \textit{ tasks} \qquad (5)$$

Afterwards it starts the learning process in which each robot updates its election probabilities according to the following conventional updating rule:

$$p_{ij}(t+l)=_{Pij}(t)+\backslash p(t) \ [1 \ -_{Pij}(t)] \qquad (6)$$

where 0 < A < 1 is the learning rate with a fixed value of 0.2; *(3(t)* is the usual reward signal generated by the environment of the learning automata with the following interpretation: *(3(t)* = 1; reward if and only if for the corresponding task *lj* at instant *t* it holds that *#Rj(t)* < *#Lj(t),* i.e. the number of robots performing task *lj* is lower than the number of tasks *lj* to be executed; *(3(t)* = 0; penalty if and only if *#Rj(t)* > *#Lj(t);* i.e. when the number of robots performing task *lj* is greater than the number of tasks *lj* or whenever there are not pending tasks to be executed the automata receives a penalty signal. In few words: at each instant *t* the environment evaluates the action of the automata, when the response generated by environment is 1 means that the action is "favorable" and if the response value is 0 corresponds to an "unfavorable" as follow:

$$R_{PL*()} \ m = \frac{t\wedge\pm}{\#Lj} = J \begin{matrix} \textbf{lf} \wedge \textbf{: then reward } l^{3=l} \\ \backslash \text{ If } > 1 \text{ then penalty } /3 = 0 \end{matrix} \qquad n\backslash_{[)}$$

4 Ant Colony Optimization Methods

For over many years, communities or colonies of social insects have been deeply studied by some researchers, as they provide fascinating examples of functional collective behavior. Ant Colony Optimization (ACO) is a meta-heuristic approach that was introduced in the early 1990's by Dorigo et al. in [15,16]. The general idea of ACO approach is to solve combinatorial optimization problems based by the behavior of real ants, more specifically, the inspiring source is how ants can find shortest paths between food sources their nest. ACO algorithms are stochastic search procedures based on a parameterized probabilistic model [17], called by the authors "the pheromone model".

In this case, a generic robot r-j selects the tasks in a deterministic way based on "forces" *fij(t).* These forces are updated, after being initialized at the "indifference" position, as follows:

$$fij(t + 1) = Pfij(t) + (1 \text{ " } P)\wedge)i \ ^{\circ} < P < ^{l} \qquad (8)$$

where *p* is the usual learning rate of ant colony optimization-like algorithms and *(3(t)* is the reward/penalty signal at instant *t* with the same exact interpretation than for the learning automata-based probabilistic algorithms.

5 Experimental Results

We have carried out a series of experiments to evaluate the system performance index by applying Ant Colony Optimization-based deterministic algorithms as well as Learning Automata-based probabilistic algorithms to solve the optimal distribution of the tasks among the *N* robots; so that all of them are executed by means of the minimum number of robots. The ideal objective is that the performance index or learning curve corresponding to the load *lj(t)* of each task tend asymptotically to zero for all curves in the minimum time and using the minimal possible number of robots for task execution.

In the simulations we have considered some variants such as: the multi-robot system size, different loads *lj(t)* for each type of task, two different ways to carry out the tasks selection, the additive noise generation to simulate the robot's error and the dynamic generation of tasks *lj(t)* over time. According to the results obtained with eq. 6 and eq. 8 we have used for the learning automata-based probabilistic algorithms and for ant colony optimization-based deterministic algorithms two mechanisms for the selection of tasks:

1. Maximum principle: at each instant *t* choose the task that has the highest probability for all *Pij(t)*.
2. The strictly random method: using the probabilities *Pij(t)* in the strict sense of the word, it generates a random number with uniform distribution (0 — 1) and it selects the appropriate task to the value obtained by the method of inversion of discrete probability distributions.

Table 1. Shows a scheme of the experiments performed with their respective variants

		Without Noise		With Noise	
	Mechanisms Approaches	Maximum principle	Strictly random method	Maximum principle	Strictly random method
Not dynamic task	Ant Colony Optimization	Fig.4(a)		Fig.4(b)	
	Learning Automata	Fig.5(a)		Fig.5(b)	
Dynamic task	Ant Colony Optimization	Fig.6(a)		Fig.6(b)	
	Learning Automata	Fig.7(a)		Fig.7(b)	

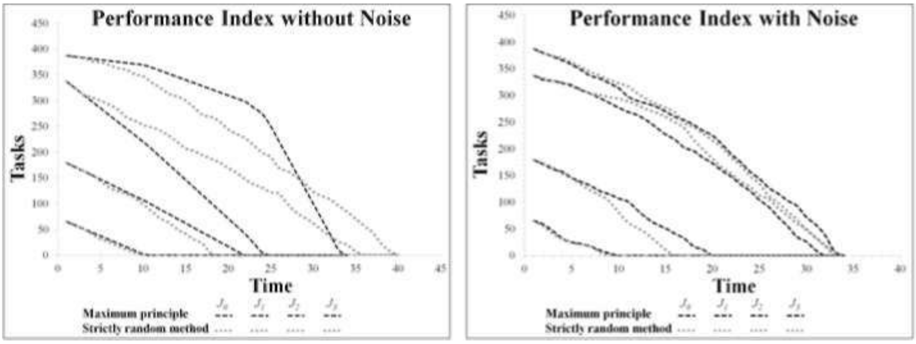
5.1 Evaluation of the Performance Index

To evaluate the evolution of the performance index we have introduced additive noise, perturbing the number of pending loads to simulate the robot's error in estimating the real number of pending tasks. The noise generated is modeled using a normal distribution ("White Noise") as follows:

$$Noise = R + R*S = R(l + S) \tag{9}$$

where *Noise* is the noise generated to the number of pending loads *h(t),* which is proportional to the amplitude of the noise *R* without perturbing, *S* is a Gaussian distribution with a mean of '0' and a typical deviation '0.005' *N(0,* 0.005).

Fig. 3 and Fig. 4 show the evolution of the system performance index obtained for self-election of heterogeneous specialized tasks through ant colony optimization-based deterministic algorithm as well as learning automata-based probabilistic algorithms, using both mechanisms: maximum principle and the strictly random method, with a team of robots formed by 20 - 30 heterogeneous robots and 4 types of heterogeneous specialized tasks with different loads. Each experiment has been run 10 times and the results shown are the mean of all.

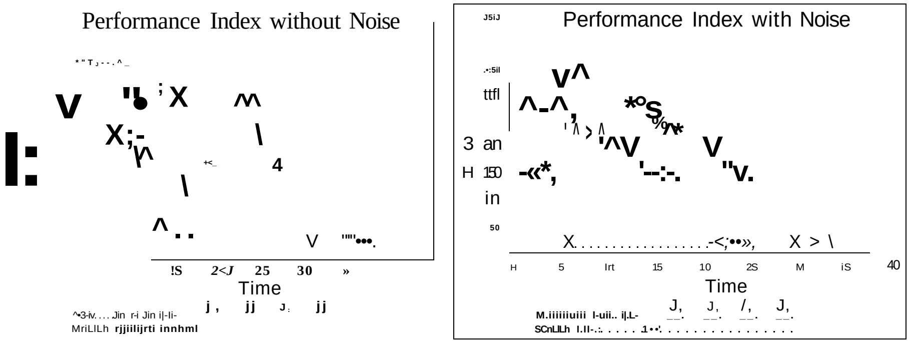


(a) (b)

Fig. 3. Learning curves with the evolution of the system performance index for self-election of tasks using Ant Colony Optimization-based deterministic algorithms

Fig. 3(a) shows the performance index without noise for both mechanisms, it can be observed that the maximum principle provides better performance instead of strictly random method. However, Fig. 3(b) shows the performance index perturbing the number of pending loads and for both mechanisms the performance is better because they finish in fewer time than in Fig. 3(a). In this case, the best results are obtained with strictly random method instead of the maximum principle.

Fig. 4(a) shows the performance index without noise for both mechanisms and Fig. 4(b) presents the performance index generating additive noise in the number of pending tasks. It can be noted that Fig. 4(b) obtains better results for both mechanisms than Fig 4(a) using Learning Automata-based probabilistic algorithms. It can be observed that learning curves corresponding to the load *lj(t)* of each task tend asymptotically to zero for both methods. Also the results shown that the generation of additive noise does not affect the performance of the approach, on the contrary, in some cases better results are obtained with the generation of noise.



(a) (b)

Fig. 4. Learning curves with the evolution of the system performance index for self-election of tasks using Learning Automata-based probabilistic algorithms

5.2 Dynamic Tasks Generation

In the previous experiments, the number of loads for each type of task is determined from the beginning of the simulation and there is not any change until the end of the execution. To evaluate the performance of the algorithm we have generated dynamic tasks. This idea was rescued from classical models of queues simulation, so we have used Poisson distribution to determine the probability of generating a number of tasks through time:

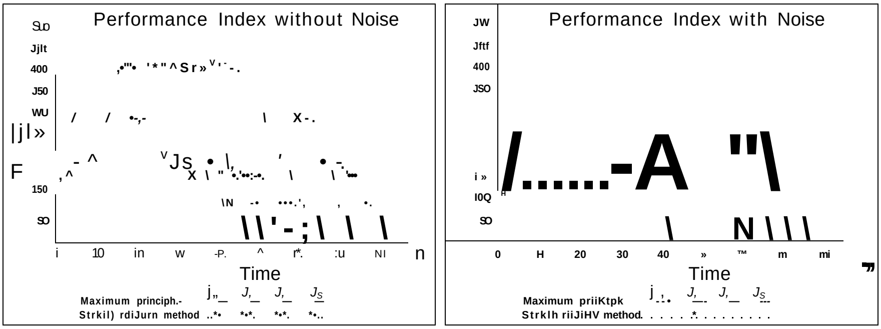
$$f(k;X) \qquad \frac{e^{A}A}{\sim fcT} \tag{10}$$

Specifically we will have a different distribution for k = 1 to 100. Each A is a positive real number that representing the number of tasks expected to be generated during a time interval. For that expected number of tasks generated is decreasing, and therefore the system is stable, we have parameterized this constant A as follows:

$$\backslash(t)=a-a*t \tag{11}$$

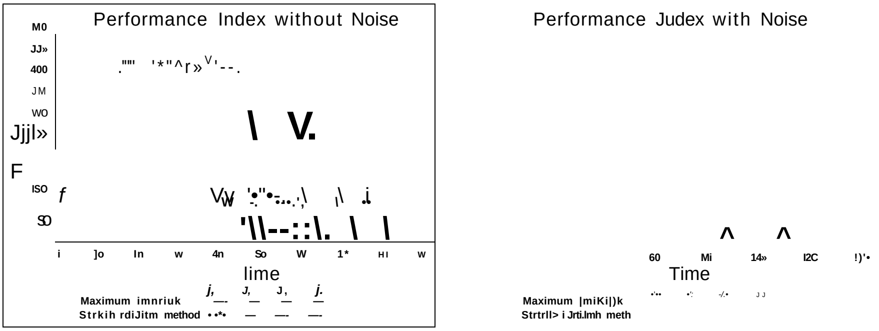
where a is the initial value (for example, 10 or 20) and a is a factor of "reduction tasks" that initially we have defined to 1. Finally, t corresponds the time of execution at each instant.

Fig. 5 and Fig. 6 show the evolution of the system performance index with dynamic tasks generation through time using the Poisson distribution. Experiments have been performed 10 times and the results shown are the mean of all, we have also additive noise generated in the loads with the maximum principle and the strictly random method. In the results it can be observed dynamic tasks generation, the tasks number generated is decreasing over time. All learning curves tend to zero in both methods and not affected the performance of the approach with the generation of additive noise, sometimes there are better results with noise.



(a) (b)

Fig. 5. Dynamic task generation: learning curves with the evolution of the system performance index using Ant Colony Optimization-based deterministic algorithms



(a) (b)

Fig. 6. Dynamic task generation: learning curves with the evolution of the system performance index using Learning Automata-based probabilistic algorithms

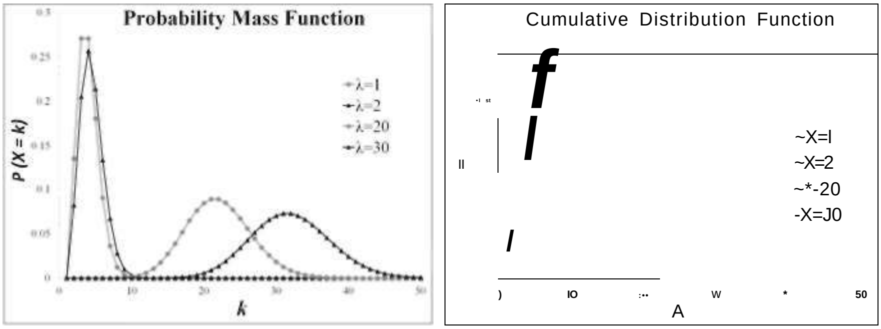


Fig. 7. The index *k* represents the number of tasks expected to be generated during a time interval for different values of A and *P(X = k)* describes the probability that a value of variable *X* with a given probability distribution is equal to *k*

Fig. 7 shows the probability mass function and the cumulative distribution function obtained in experiments with dynamic task generation using the Poisson distribution.

6 Conclusions and Further Work

In this paper we have applied two different approaches to the self-coordination problem of multi-robot systems in the heterogeneous multi-tasks distribution by applying Ant Colony Optimization-based deterministic algorithms as well as Learning Automata-based probabilistic algorithms. To carry out the selection of tasks in both approaches we used two mechanisms: maximum principle and the strictly random method and, in most experiments the best results are obtained with strictly random method instead of the maximum principle. We have generated additive noise to evaluate the robustness to both approaches, perturbing the number of pending load, to simulate the robot's error in estimating the real number of pending tasks, according to the results obtained the noise generated does not affect the performance of the approaches since the best result are obtained by generating noise in the pending loads. We have also studied the performance index with dynamic generation of loads through time and the results confirm that the robots are capable to select in an autonomous and individual manner the existing tasks without the intervention of any global and central tasks scheduler. We have shown that both approaches can be efficiently applied to solve this self-coordination problem in multi-robot systems obtaining truly decentralized solutions.

References

1. Gerkey, B., Mataric, M.: Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In: IEEE International Conference on Robotics and Automation, pp. 3862-3868 (2003)
2. Gerkey, B., Mataric, M.: A formal analysis and taxonomy of task allocation in multi-robot systems. Intl. J. of Robotics Research, 939-954 (2004)
3. Farinelli, A., Locchi, L., Nardi, D.: Multirobot systems: a classification focused on coordination. IEEE Transactions on Systems, Man and Cybernetics, 2015-2028 (2004)
4. Oster, G., Wilson, E.: Caste and ecology in the social insects. Monographs in Population Biology. Princeton Univ. Press (1978)
5. Robinson, G.: Regulation of division of labor in insect societies. Annu. Rev. Entomol., 637-665 (1992)
6. Quifionez, Y., de Lope, J., Maravall, D.: Bio-inspired Decentralized Self-coordination Algorithms for Multi-heterogeneous Specialized Tasks Distribution in Multi-Robot Systems. In: Ferrandez, J.M., Alvarez Sanchez, J.R., de la Paz, F., Toledo, F.J. (eds.) IWINAC 2011, Part I. LNCS, vol. 6686, pp. 30-39. Springer, Heidelberg (2011)
7. Quifionez, Y., Maravall, D., de Lope, J.: Stochastic Learning Automata for Self-coordination in Heterogeneous Multi-Tasks Selection in Multi-Robot Systems. In: Batyrshin, I., Sidorov, G. (eds.) MICAI 2011, Part I. LNCS, vol. 7094, pp. 443-453. Springer, Heidelberg (2011)

8. Gerkey, B., Mataric, M.: Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In: IEEE International Conference on Robotics and Automation, pp. 3862-3868 (2003)
9. Narendra, K., Thathachar, M.: Learning Automata: An Introduction. Prentice-Hall, Englewood Cliffs (1989)
10. Narendra, K., Thathachar, M.: Learning Automata: A Survey. IEEE Transactions on Systems, Man, and Cybernetics, 323-334 (1974)
11. Obaidat, M., Papadimitriou, G., Pomportsis, A.: Guest Editorial Learning Automata: Theory, Paradigms, and Applications. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 706-709 (2002)
12. Maravall, D., De Lope, J.: Fusion of Learning Automata Theory and Granular Inference Systems: ANLAGIS. Applications to Pattern Recognition and Machine Learning. Neurocomputing 74, 1237-1242 (2011)
13. Narendra, K., Wright, E., Mason, L.: Applications of Learning Automata to Telephone Traffic Routing and Control. IEEE Transactions on Systems, Man, and Cybernetics, 785-792 (1977)
14. Narendra, K., Viswanathan, R.: A Two-Level System of Schotastic Automata for Periodic Random Environments. IEEE Transactions on Systems, Man, and Cybernetics, 285-289 (1972)
15. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: an autocatalytic optimizing process, Technical Report TR91-016, Politecnico di Milano (1991)
16. Dorigo, M.: Optimization, learning and natural algorithms. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan (1992)
17. Dorigo, M., Blum, C: Ant colony optimization theory: A survey. Theoretical Computer Science 344(2-3), 243-278 (2005)