Minimal Time Dynamic Task Allocation for a Swarm of Robots

Maha A. Alshawi
AUC/Robotics, Control and Smart System Department, Cairo, Egypt
Email: mahaelshawi@aucegypt.edu

Mohamed B. Shalan
AUC/Computer Science Department, Cairo, Egypt
Email: mshalan@aucegypt.edu

***Abstract*—This paper discusses a solution to one of the key issues in the swarm robotics field which is the dynamic task allocation problem in which a group of robots needs to be allocated to a set of tasks scattered in the environment in an efficient and decentralized way. The application considered in this context is the foraging application which can be addressed as a searching job followed by a transportation job. The near-optimal allocation scheme is found by using the Particle swarm optimization (PSO) technique to handle the whole task execution in a minimal time. Two case studies have been considered using different swarm sizes and the implemented code has been executed for a distinctive number of iterations. A stability proof for the PSO technique's parameters choices is presented. Simulation results were verified by comparing the proposed algorithm with the simulated annealing optimization technique in terms of computational time, number of iterations needed and quality of solution to demonstrate the robustness and efficiency of the algorithm.**

***Index Terms*—dynamic task allocation, swarm robotics, particle swarm optimization, simulated annealing, swarm optimization, homogeneous robots.**

I. Introduction

One of the major problems exists in robotics field is the task planning problem. Task planning consists of two main parts. First, there is a decomposition of a complex task into small simple sub-tasks in which a swarm of robots must cooperate with each other and with the environment to perform those tasks. Second, a task allocation (TA) is needed to distribute and schedule a set of tasks to be accomplished by a group of robots to optimize the performance while satisfying operational constraints. TA is one of the main issues to be elaborated on in multi-robot systems. Dynamic task allocation (DTA) implies that robots have no prior knowledge about the number of robots, the number of tasks, and the temporal and spatial distribution of tasks in the environment. However, they should communicate periodically with each other to adapt online to the environment [1]. Minimal Time Dynamic Task Allocation (MTDTA) seeks to allocate tasks dynamically to the robots while minimizing the total time needed to accomplish the whole task.

Due to the advantages of stability, scalability, energy efficiency, economist, parallelism, technical progress and the declining cost of robotic mobility, interest in this area of application has grown significantly in recent years. The DTA problem is encountered in various application domains of multi-robot systems, such as human rescuing, geological survey, agricultural foraging, military applications, UAV controlling, demining, warehousing, toxic waste clean-up, collection of terrain samples, cooperative transportation, autonomous exploration and mapping, distributed monitoring and surveillance, etc.

As has been said before, the focus on this paper is on the foraging problem which can be addressed as a searching job followed by a transportation job [1]. Each job consists of a number of tasks. Each robot in the foraging problem has to decide whether it will explore a new object or it is going to transport an explored object to the nest.

The final destination of the objects is called the nest or the objects prey. These set of objects can be one of two types, either a single-robot object which needs only one robot to carry it from its current location to the nest or a multi-robot task which requires at least two robots working simultaneously in order to retrieve the object to its final destination. Also, if the robot chooses the transportation job, it has also to decide which one of the scattered objects it will select to return to the nest. The robot choice is not random. However, each robot has to choose the most appropriate object to maximize the overall performance of the group [1]. The DTA problem in a cooperative foraging scenario with shared task execution by multiple robots is an NP-hard problem.

The proposed algorithm will be decentralized which means that neither a global knowledge, nor messages' broadcast nor a multi-hop communication would ever be used. There is no central unit to take care of the task allocation, and each individual in the swarm has to identify the task it must perform via the communication with only their neighbors. This decentralization is essential to avoid the restrictions that could result from the communication range limitations.

Manuscript received March 16, 2017; revised August 25, 2017.

© 2017 Int. J. Mech. Eng. Rob. Res.

doi: 10.18178/ijmerr.6.6.481-487

The rest of the paper is structured as follows: in the next section, related research work is outlined. Section 3 describes the minimal time dynamic task allocation within the framework of the foraging problem with a changing number of tasks and robots. Section 4 outlines the proposed algorithm. Simulation results discussion and a comparison with the simulated annealing (SA) technique are presented in section 5. Finally, the conclusion and future work are covered.

II. Related Work

Recently, several algorithms have been proposed in the literature for dynamic task allocation problems. One of the most common available taxonomies has implemented the algorithm classification according to the behavioral approach, the market laws or the bio-inspired approach [2]. Tasks are divided into behavioral groups in the behavioral approach in which each group contains a set of tasks to be executed that have relations among each other. The most common algorithm in the behavioral approach is ASYMTRE [3, 4]. In the market laws approach, algorithms aim at maximizing or minimizing cost functions, for example speed and convergence time. Algorithms based on market laws could follow any optimizing technique like auction and thresholds-based methods [5], Markov chain search process along with SA technique [6], or heuristic search-based task allocation technique [7]. Some algorithms may specify a set of constraints on the task allocation process, such as determining a deadline for the task assignment accomplishment, consider the size of each task, or specify the robots' performance in their assigned tasks [8]. The most commonly used approach in terms of research and publications is the bio-inspired technique. It is derived from the social insect's behavior. A series of algorithms based on the bio-inspired technique have been proposed in the literature, such as ant-inspired algorithms [9, 10], a parallel PSO algorithm [11, 12], and an artificial bee colony algorithm [13]. As mentioned by Krieger et al. [9], it is shown that a higher level of energy could be maintained in sets of robots which use ant-inspired optimization techniques than in single robots. They could also forage more expeditiously [9]. The bio-inspired techniques exhibit some praiseworthy properties such as self-organizing capacity and flexible behavior towards environmental changes.

Other taxonomies have addressed the algorithm classification base on the centralized versus the distributed approach. In the centralized approach, there is a leader or central unit which is responsible for the task assignment to robots. For example, Gigliotta et al. have evolved a dynamic task allocation rules by communicative interactions in a group of homogeneous robots. They focused on the development of a team of robots in which one and only one individual robot (the 'leader') must differentiate its communicative attitude from that of all the others ('non-leaders'). The robots have evolved for their capability to distinguish their roles by the discrimination of their signals. The leader robot has to maximize the value of its communicative signal, while all other robots have to minimize their signals' value. The leader robot tends to send high signal's value, while the non-leaders tend to send a low signal's value. The fitness of a group of robot has been calculated in the following way. The average of the differences between the output signal of the current leader which has the maximum value and the output signals of all other non-leader robots has been calculated every iteration. A number of trials need to be implemented. At the end, the average of the calculated value for all iterations of all the trials can be considered as the fitness value [14]. Formally, this is how fitness value was calculated:

$$F = \frac{\sum_j^C \sum_i^N Max - O_i}{C(N-1)}, \quad (1)$$

where the number of robots in each group represented by N, such as 10. While, the total number of iterations of each individual called C, such as 1000 iterations * 40 trials = 40000. Also, the signal's value of the current leader is Max and the signal's value of robot j is O_j.

While in the distributed approach, there is no central unit to take care of the task allocation. Thus, each robot in the swarm has to identify the task it must perform. A lot of algorithms have been proposed in this approach, like the self-organized method which neither relies on global knowledge nor centralized components and it does not require massive robots' interactions [15-17]. Also, a fully decentralized approach which needs neither a global broadcast nor a multi-hop communication protocol has been enhanced [13]. Zhang, Xie, Yu, and Wang have established a hierarchical assignment architecture and have utilized a "local blackboard" communication mechanism for knowledge sharing to avoid using a centralized leader [18].

Finally, other taxonomies classified the algorithms based on a single robot-task approach against a multi-robot task approach [19]. In a single robot-task, only a single robot is needed to perform each task. However, in a multi-robot task, a task has to be cooperatively carried out by several robots. This DTA problem is a strongly NP-hard problem and the complexity significantly depends on the number of robots required by each multi-robot task. Some proposed algorithms in this taxonomy have identified an efficient multi-foraging behavior in which each task required multiple robots to share the task's execution to complete the task [1, 20]. Liu and Kroll [21] have discussed the fact that it can be more challenging when tightly coupled multi-robot tasks are taken into consideration due to the resulting temporal and spatial constraints. Additionally, the complexity of the task allocation increases exponentially with rising tasks' varieties. They have also presented a novel mimetic algorithm combining a genetic algorithm with two local search schemes to solve the multi-robot task allocation problem in inspection problems [21]. Ducatelle et al. [22] have proposed a straightforward reactive technique in which robots interact with each other using light signals instead of the traditional way of communications. Also, the Random-Choice, Extreme-Comm, Card-Dealer, and

Tree-Recolor algorithms have been implemented by McLurkin and Yamins [23] on a set of 25 robots to compare between their accuracies.

III. Minimal Time Dynamic Task Allocation

The MTDTA problem can be broadly defined as follows, given a set of tasks, a set of robots that can perform the tasks, and a fitness function that measures the completion time of different combinations of robots from the robot set in performing the tasks, find a suitable matching or assignment between the set of tasks and the group of robots which minimizes the total time of the fitness function. Thus, the MTDTA algorithm model can be described as follows:

Let

$$T = \{t1, t2, ..., t^{Nt}\} \quad (2)$$

be the set of task identifiers to be allocated to the robots in the swarm, and let

$$R = \{r1, r2, ..., r^{Nr}\} \quad (3)$$

be the set of robots' identifiers in the robotic swarm. So, the problem is composed of Nt valid tasks and Nr robots. The swarm allocation is represented by:

$$A = \{a1, a2, ..., a^{Nr}\} \quad (4)$$

where a_j identifies the task allocated to robot r_j. Hence, the solution of the minimal time dynamic task allocation problem is achieved by finding an allocation A*, which represents the allocation of the group of Nr robots to the set of Nt tasks in a minimum completion time of the whole task.

The simulation experiment was done by using MATLAB [24]. The arena is a square space with the area equals to 100*100 cm^2. Robots were scattered randomly in the environment at positions represented in the form of points' pairs, for example (x,y). Robots could perceive the objects, obstacles, and the nest by using infrared red proximity and ground sensors. Prey were also distributed randomly in the environment at the experiment start time. The environment was totally dynamic. New prey could be added any time during the experiment. Also, the number of robots could be increased by adding extra robots, or decreased if one of the robots has broken or needs to be recharged.

Due to the robots' simple mobility requirements and the inexpensive perception sensors, this research deals with deploying a large number of simple inexpensive robots. Two case studies are discussed in this paper. The first case study was implemented by introducing 7 robots to transport 10 prey to the nest. The second case study was represented by using 10 tasks and 20 robots. All the robots move with the same speed of 30 m/s. Therefore, the distance which the robots move to reach to the target position could be calculated instead of the time robots took to reach their tasks.

IV. The Proposed Algorithm

The proposed algorithm is based on the PSO technique. The advantages of the adaptation capability to the dynamic environment and the objective function's continuity with low constraint make PSO one of the most promising swarm intelligence techniques. As can be seen from the pseudo code, the algorithm steps can be summarized as follows:

1. First, a random allocation is chosen for each particle in the swarm A.
2. Then, the personal best (A_{pbest}) and the global best (A_{gbest}) for each particle is initialized to be equal to its own initial allocation.
3. Then, the algorithm starts to iterate for a specific number of iterations (maxit).
4. The fitness value of each particle allocation is calculated according to the fitness function which is the minimum distance from each robot to its corresponding task (Euclidean distance).
5. After that, a comparison is made between the current calculated fitness value for each particle and the previous calculated best fitness. If the new allocation fitness is smaller, it is chosen to be the particle personal best. Then, the smallest fitness among all the particles is chosen as the global best.
6. Then, the particle which has the global best allocation sends its allocation to all other particles.
7. Moreover, each particle calculates the velocity which it uses to move towards its new allocation according to the velocity update equation of the PSO as specified in equation (5).

$$v(t+1) = w * v(t) + c1 * rand * (pbest-x(t)) + c2 * rand * (gbest-x(t)) \quad (5)$$

   Where, constants c1 and c2 is chosen to be 2, and rand is a random number from 0 to 1.
8. Furthermore, the position or allocation for each particle is updated by using equation (6).

$$x(t+1) = x(t) + v(t+1) \quad (6)$$

9. Finally, if the number of iterations reaches the maximum iteration, the algorithm stops, and each robot begins its task execution according to the global best allocation solution found (A^*). If not, steps from 4 to 9 are repeated.

Through the stability analysis of the PSO's equations which has been proposed by Tian [22], the particles' stability constriction condition has been deduced to be:

$$v(t+1) = k * [v(t) + c1 * rand * (pbest-x(t)) + c2 * rand * (gbest-x(t))] \quad (7)$$

where $k = 2 / |2 - c - (c^2 - 4c)^{1/2}|$. Thus, It has been realized that if the parameters in 4 and 5 match this condition and if $c=c1+c2 \geq 4$, then the particle's trajectory in the PSO system is stable. That's why, in this paper, the proposed algorithm's parameters have been chosen to be c1=c2=2 and the inertia weight w=1 [25].

V. Simulation Results and Comparison

Considering the two case studies in the experiment, the algorithm in the first case study needed to find the best allocation of the 10 tasks to the 7 robots. In such a case,

only 7 tasks were allocated while the remaining 3 tasks were waiting in the waiting list. The algorithm was executed periodically. Thus, if any robot finished its allocated task, left for recharging or new tasks were explored which need to be allocated, the algorithm should calculate a new allocation for the robots in the swarm. In the second case study, there are 10 robots and 20 tasks. Thus, only 10 tasks were allocated at the beginning. The allocation for each particle in the swarm is represented as a raw with a number of elements equals to the number of robots in the swarm and the number which is written in each cell represents the task allocated to the corresponding robot.

a) Main code

```
1: Initialization (Nt, Nr);
2: While it ≥ maxit {
3:      Personal best update(A, Apbest, Agbest);
4:      Global best update(Pbest);
5:      Global best allocation diffusion(idgbest);
6:      Calculate particle's velocity;
7:      Update particle's position;
8: }
9: Execute;
```

b) Initialization

```
Input: Nt, Nr
Output: A, Apbest, Agbest
1: For i = 1 to Nr {
2:      A[i] = random generator (1, …, Nt);
3: }
4: Apbest = A, Agbest = A;
```

c) Personal best update

```
Input: A, Apbest, Agbest
Output: Pbest
1: Calculate fitness value f (A);
2: if f (A) ≥ f (APbest) {
3:      Apbest = A;
4: }
5: msg ← (id, f(APbest));
6: send fitness value to all other robots;
7: For every other robot {
8:      Exchange the fitness value;
9: }
10: For i = 1 → Nr {
11:     Pbest[i] = f(APbest) for robot id = i;
12: }
```

d) Global best update

```
Input: Pbest
Output: idgbest
1: idgbest = 1;
2: Pbestmin = Pbest [1];
3: for i = 2 → Nr {
4:      if Pbestmin > Pbest[i] {
5:           idgbest = i;
6:           Pbestmin = Pbest[i];
7:      }
8: }
```

e) Global best allocation diffusion

```
Input: idgbest
Output: Agbest
1: if idgbest is my id {
2:      Send (Agbest);
3: }
4: Else {
5:      Receive (Agbest);
6: }
```

For example, let the task allocation be as shown in the second row of Table I. The first row is just an identifier for the robot number to which the corresponding underneath task is allocated. In this case, task 3 is allocated to robot 1, task 5 is allocated to robot 2, task 8 is allocated to robot 3, task 2 is allocated to robot 4, task 9 is allocated to robot 5, task 4 is allocated to robot 6, and task 1 is allocated to robot 7. This allocation is corresponding to the first case study.

TABLE I. TASK ALLOCATION IN THE 1ST STUDY CASE

r1	r2	r3	t4	r5	r6	r7
t3	t5	t8	t2	t9	t4	t1

Another example on the allocation corresponding to the second case study is as shown in Table II. 20 tasks are allocated to 10 robots.

TABLE II. TASK ALLOCATION IN THE 2ND STUDY CASE

r1	r2	r3	t4	r5	r6	r7	r8	r9	r10
t6	t14	t9	t11	t2	t19	t3	t17	t8	t5

In the simulation, tasks were displayed as yellow circles, while robots were represented as red squares to be able to differentiate between tasks and robots. Allocations between robots and tasks were addressed as a black line extended from each robot to its allocated task. Fig. 1 and 2 show the best solution A* resulting after 500 iterations for both PSO and SA, respectively. Fig. 1 show that PSO gives a better allocation as each robot was assigned to its nearest task. However, in f Fig. 2 some robots select a far task rather than the nearest one.

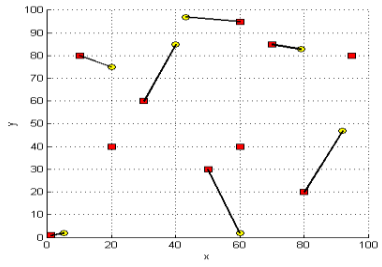

Figure 1. Simulation by PSO for the 1st study case

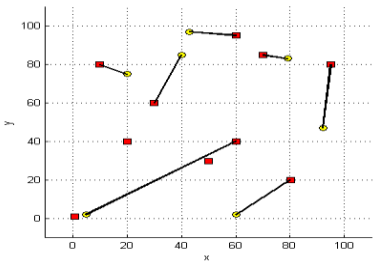

Figure 2. Simulation by SA for the 1st study case

Simulation results corresponding to the second case study are shown in Fig. 3 and 4 for PSO and SA, respectively. If the distance between the robots and their corresponding tasks is tracked in both cases, it could be

easily explored that the distance traveled by robots in Fig.3 is much smaller than the traveled distance in Fig. 4.

A comparison about the MTDTA performance has been made between PSO and SA. Both of them have been executed for 100, 200, 300, 400, and 500 iterations and the corresponding algorithm run time for each case has been also calculated.

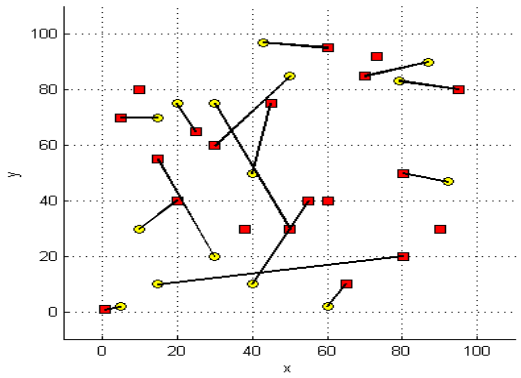

Figure 3. Simulation by PSO for the 2nd study case

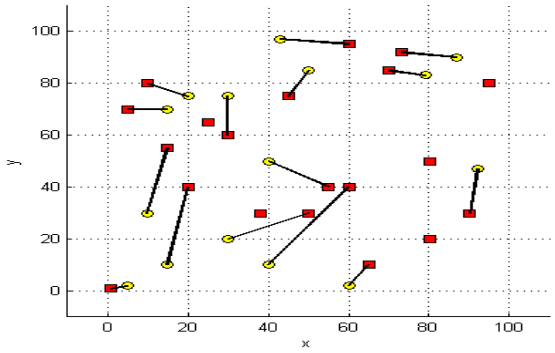

Figure 4. Simulation by SA for the 2nd study case

Table III and IV show the results from the first and second case studies, respectively. It can be deduced from Table III that the distance travelled using PSO is smaller than that of the SA in almost all cases and the gap between the PSO's and SA's fitness kept increasing till reaching its maximum value after 500 iterations. For example, the difference between the travelled distance by the robots in PSO and SA after 100 iterations equals (207.9 cm - 165.4 cm= 42.5 cm). However, the difference after 500 iterations equals (199.6 cm - 128.6 cm= 71 cm). Furthermore, it can be shown that after 150 iterations, SA's fitness has been trapped in a local minimum at 199.6 cm and it never changes. However, PSO's fitness kept changing until reaching the global minimum value at 128.6 cm.

By looking at Table IV, it can be figured out that the PSO's travelled distance (419.3 cm) is smaller than that of the SA (426.7 cm) for small number of iterations. However, when the number of iterations increases above 250, the PSO's travelled distance which is (356.5 cm) becomes larger than that of the SA which is (250.9 cm). Moreover, in this case study, the PSO's fitness value has been trapped in a local minimum at 356.5 cm after 400 iterations, while the SA's fitness value kept decreasing until reaching the global minimum solution at 250.9 cm.

TABLE III. PERFORMANCE RESULTS FOR THE 1ST CASE STUDY

Optimization technique	Number of iterations	Algorithm run time	Distance travelled by robots
PSO	100	4.9 sec	165.4 cm
	200	5.8 sec	157.7 cm
	300	9.5 sec	153.5 cm
	400	9.6 sec	143.2 cm
	500	14.6 sec	128.6 cm
SA	100	14.3 sec	207.9 cm
	200	18.9 sec	199.6 cm
	300	18.9 sec	199.6 cm
	400	24.1 sec	199.6 cm
	500	23.8 sec	199.6 cm

TABLE IV. PERFORMANCE RESULTS FOR THE SECOND CASE STUDY

Optimization technique	Number of iterations	Algorithm run time	Distance travelled by robots
PSO	100	7 sec	431.9 cm
	200	6.4 sec	419.3 cm
	300	14.1 sec	411.8 cm
	400	12.7 sec	356.5 cm
	500	20.8 sec	356.5 cm
SA	100	19.1 sec	444.3 cm
	200	27.5 sec	426.7 cm
	300	25.3 sec	384.2 cm
	400	32 sec	319.2 cm
	500	31.6 sec	250.9 cm

To sum up, it can be concluded by looking at Fig. 5 and 6 that it is always recommended to use the PSO technique than the SA technique for a swarm with a small number of robots and tasks (1st case study). However, if the swarm size is large (2nd case study), techniques' preference depends on the number of iterations. Thus, it is a compromise between a global optimal solution that takes long time to be detected, and a good, but not optimum, solution that could be found faster.

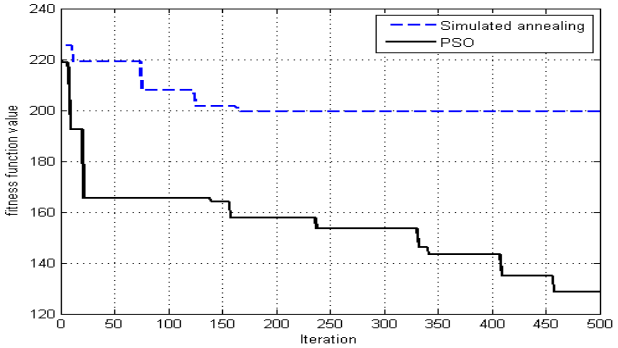

Figure 5. PSO vs. SA after 500 iterations for 1st study case

© 2017 Int. J. Mech. Eng. Rob. Res.

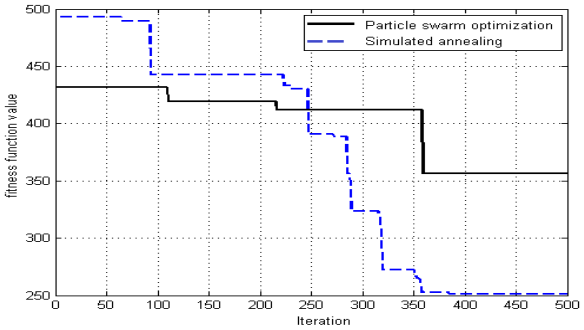

Figure 6. PSO vs. SA after 500 iterations for 2nd study case

VI. Conclusion

In this paper, an optimization algorithm is proposed to perform a minimal time dynamic task allocation in a swarm of robots. MTDTA algorithm gives a promising solution for tasks assignment among robots in the environment. The task allocation decision process is carried out independently by each robot using a clear and straightforward procedure to accomplish the whole task in a minimal time. The simulation results of the algorithm show the execution time of the MTDTA algorithm, as well as the number of iterations needed to achieve the solution and the distance travelled by robots in centimetres. The distance generated from the PSO technique's execution is compared with the distance produced from other existing solution based on the SA technique to demonstrate the robustness and efficiency of the proposed algorithm.

This work could be extended in the future by improving the capability of the proposed algorithm. A set of heterogeneous robots could be used instead of homogenous ones. Tightly coupled tasks in which tasks depend on each other is a very challenging problem that could be considered, and precedence among these tasks must be taken into account. Multi-robot tasks which need more than two robots working simultaneously to implement a single task can also be addressed in the future. This type of tasks adds a level of complexity to the problem because lots of constraints could show up.

References

[1] A. Campo and M. Dorigo, "Efficient multi-foraging in swarm robotics," in *Proc. Advances in Artificial Life*, Springer, 2007, pp. 696-705.

[2] Y. Zhang and S. Liu. (2008). "Survey of multi-robot task allocation". *CAAI Transactions on Intelligent Systems*. 3 (2). pp. 115–120.Available: http://en.cnki.com.cn/Article_en/CJFDTOTALZ XT200802006.htm.

[3] F. Tang and L. E. Parker, "Automated synthesis of multi-robot task solutions through software reconfiguration," in *Proc. of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 1501-1508.

[4] A. Tsalatsanis, A. Yalcin, and K. P. Valavanis. (2009). Dynamic task allocation in cooperative robot teams. *International Journal of Advanced Robotic Systems.* [Online]. *6*(4). p. 35. Available: http://journals.sagepub.com/doi/abs/10.5772/7257.

[5] J. Guerrero and G. Oliver. (2003). Multi-robot task allocation strategies using auction-like mechanisms. *Artificial Intelligence Research and Development.* [Online]. 100(12). pp. 111-122.

[6] K. Zhang, E. Collins, and D. Shi. (2012). "Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Transactions on Autonomous and Adaptive Systems.* [Online]. 7(2). pp. 1-22. Available: http://dl.acm.org/citation.cfm?id=2240171.

[7] T. Nagarajan and A. Thondiyath. (2014). An algorithm for cooperative task allocation in scalable constrained multiple robot systems. *Intelligent Service Robotics*. [Online]. 7(4). pp. 221-233. Available:https://link.springer.com/article/10.1007/s11370-014-0154-x.

[8] Y. Khaluf and F. Rammig, "Task allocation strategy for time-constrained tasks in robot swarms," in *ECAL*, 2013, pp. 737-744.

[9] M. Krieger, J. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, pp. 992-995, 2000.

[10] S. Momen and A. J. C. Sharkey. (2009). An ant-like task allocation model for a swarm of heterogeneous robots. *Swarm Intelligence Algorithms and Applications Symposium.* [Online]. pp. 31-38. Available:https://www.researchgate.net/profile/Amanda_Sharkey2/publication/228950263_An_ant_like_task_allocation_model_for_a_swarm_of_heterogeneous_robots/links/0046352d65c75b570a000000.pdf.

[11] H. Liu, P. Zhang, B. Hu, and P. Moore. (2015). A novel approach to task assignment in a cooperative multi-agent design system. *Applied Intelligence*. [Online]. 43(1). pp. 162-175. Available: https://link.springer.com/article/10.1007/s10489-014-0640-z.

[12] N. Nedjah, R. Mendonça, and L. Mourelle. (2015). PSO-based distributed algorithm for dynamic task allocation in a robotic swarm. *Procedia Computer Science*. [Online]. 51. pp. 326-335. Available:http://www.sciencedirect.com/science/article/pii/S1877050915010583.

[13] L. Liu, N. Michael, and D. Shell. (2015). Communication constrained task allocation with optimized local task swaps. *Autonomous Robots*. [Online]. 39(3). pp. 429-444. Available: https://link.springer.com/article/10.1007/s10514-015-9481-9.

[14] O. Gigliotta, M. Mirolli, and S. Nolfi. (2014). Communication basesd dynamic role allocation in a group of homogeneous robots. *Natural Computing*. [Online]. 13(3). pp. 391-402. Available: https://link.springer.com/article/10.1007/s11047-014-9443-8.

[15] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo. (2012). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems.* [Online]. 28(1). pp. 101-125. Available: https://link.springer.com/article/10.1007/s10458-012-9212-y.

[16] D. Zhang, G. Xie, J. Yu, and L. Wang. (2007). Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*. [Online]. 55(7). pp. 572-588. Available: http://www.sciencedirect.com/science/article/pii/S0921889007000267.

[17] B. Gerkey. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*. [Online]. 23(9). pp. 939-954. Available: http://journals.sagepub.com/doi/abs/10.1177/0278364904045564.

[18] C. Liu and A. Kroll, "Mimetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks," *Soft Computing*, pp. 567-584, 2014.

[19] R. Mathias de Mendonça, N. Nedjah, and L. de Macedo Mourelle. (2016). Efficient distributed algorithm of dynamic task assignment for swarm robotics. *Neurocomputing.* [Online]. 172. pp. 345-355.Available:http://www.sciencedirect.com/science/article/pii/S0925231215010516.

[20] S. Keshmiri and S. Payandeh. (2013). Multi-robot dynamic task allocation: a case study. *Intelligent Service Robotics*. [Online]. 6(3). pp. 137-154. Available: https://link.springer.com/article/10.1007/s11370-013-0130-x.

[21] P. Dasgupta, "Multi-Robot task allocation for performing cooperative foraging tasks in an initially unknown environment," in *Innovations in Defence Support Systems -2*, Springer, 2011, pp. 5-20.

[22] F. Ducatelle, A. Forster, G. A. Di Caro, and L.M. Gambardella, "New task allocation methods for robotic swarms," in *Proc. 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions*, 2009.

[23] J. McLurkin, D. Yamins. (2005). Dynamic task assignment in robot swarms. *Robotics: Science and Systems*. [Online].8.

© 2017 Int. J. Mech. Eng. Rob. Res.

Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.435.2371&rep=rep1&type=pdf.
[24] Mathworks. [Online] Available: https://fr.mathworks.com/.
[25] D. P. Tian. (2013). A review of convergence analysis of particle swarm optimization. *International Journal of Grid and Distributed Computing.* [Online]. 6(6). pp. 117-128. Available: http://citeseerx .ist.psu.edu/viewdoc/download?doi=10.1.1.587.1006&rep=rep1&type=pdf.

Maha Alshawi was born in 1991 in Bahrain. In 2012, she graduated from computer engineering school at Mansoura University in Egypt. Currently, she is a teaching and research assistant at the American University in Cairo, and is doing her research on robotics, control, and smart systems field. Her main research interests include: embedded systems, artificial intelligence, swarm robotics, automatic control Systems, automata theory, and algorithms.

Mohamed Shalan is an Associate Professor at the Department of Computer Science and Engineering, The American University in Cairo. He received his Ph.D. in computer engineering from Georgia Institute of Technology (GaTech) in 2003. He received his B.Sc. (with Honors) and M.Sc. in Computer and Systems engineering from Ain Shams University, Cairo, Egypt in 1993 and 1997. His research interests are in the area of computer engineering, with focus on embedded systems, digital design, energy-efficient, computing systems, and design automation. Prof. Shalan has over 30 refereed conference and journal papers. Also, he holds 2 US patents.

© 2017 Int. J. Mech. Eng. Rob. Res.