Multi-robot Task Allocation Using Clustering Method

Farzam Janati, Farzaneh Abdollahi, Saeed Shiry Ghidary, Masoumeh Jannatifar, Jacky Baltes and Soroush Sadeghnejad

Abstract This paper introduces an approach to solve the task assignment problem for a large number of tasks and robots in an efficient time. This method reduces the size of the state space explored by partitioning the tasks to the number of robotic agents. The proposed method is divided into three stages: first the tasks are partitioned to the number of robots, then robots are being assigned to the clusters optimally, and finally a task assignment algorithm is executed individually at each cluster. Two methods are adopted to solve the task assignment at each cluster, a genetic algorithm and an imitation learning algorithm. To verify the performance of the proposed approach, several numerical simulations are performed. Our empirical evaluation shows that clustering leads to great savings in runtime (up to a factor of 50), while maintaining the quality of the solution.

F. Janati (✉) · M. Jannatifar · S. Sadeghnejad
Robotics Institute, AmirKabir University of Technology,
424 Hafez Ave, Tehran, Iran
e-mail: farzam.janati@aut.ac.ir
URL: http://arc.aut.ac.ir/en/

M. Jannatifar
e-mail: m.jannatifar@aut.ac.ir

S. Sadeghnejad
e-mail: s.sadeghnejad@aut.ac.ir

F. Abdollahi
Department of Electrical Engineering, AmirKabir University of Technology,
424 Hafez Ave, Tehran, Iran
e-mail: f_abdollahi@aut.ac.ir

S.S. Ghidary
Department of Computer Engineering and Information Technology,
AmirKabir University of Technology, 424 Hafez Ave, Tehran, Iran
e-mail: shiry@aut.ac.ir

J. Baltes
University of Manitoba, Winnipeg, MB R3T 2N2, Canada
e-mail: jacky@cs.umanitoba.ca

J.-H. Kim et al. (eds.), *Robot Intelligence Technology and Applications 4*,
Advances in Intelligent Systems and Computing 447,
DOI 10.1007/978-3-319-31293-4_19

Keywords Multi-agent · Task assignment · Clustering · Genetic algorithm · Imitation learning

1 Introduction

Cooperation between groups of robots improves the performance of the mission and enables the agents to accomplish a goal they are not able to do individually. The objective of cooperation is to ensure that every single decision, made by agents, results in optimal decisions for the whole team. A multi-agent system consists of multiple independent agents interacting together. Each agent, in the multi-agent system (MAS), makes decisions and acts autonomously, based on its observations and also shared information of other agents, in order to reach a joint goal in the mission. MAS has some potential advantages over single robot systems, MAS can improve performance, scalability, and robustness of a mission by parallelizing actions [1], Also it can distribute the computational resources through the group of agents in decentralized systems. The fields of application of MAS have a wide range of applications, such as autonomous surveillance, reconnaissance, and exploration missions [2–4].

Task allocation and path planning are necessary for efficient operation of multiple robots. The goal of task allocation is to find a match between agents and tasks that maximizes the overall utility of the team. The optimization criteria vary due to purpose of the mission. Some of them minimizing the time of accomplishing the mission, minimizing the overall path of robots, or minimizing the energy consumption of all robots.

Task allocation for robots can be formulated as a multiple traveling salesman problem (MTSP), which is a challenging problem and considered as a NP-hard combinatorial optimization problem. Therefore there is not a specific method for finding an optimal solution. Many approaches have been developed to overcome the complexity and find optimal solutions. One of the approaches is Mixed Integer Linear Programming (MILP), which is a mathematical programming method [5–7]. Although solutions provided by MILP are acceptable, it is computationally expensive. Another approach is meta-heuristic algorithms such as Genetic Algorithm (GA) [8–10] and particle swarm optimization (PSO) [11]. GA and PSO are generic methods for finding suboptimal solutions. The meta-heuristic approaches obtain solutions quickly, however the quality of solutions might be poor, and also they might easily become intractable for large-sized problems. Another promising solution for solving the task allocation problem is introduced by learning algorithms. These approaches use a set of examples and extract the policy and build a model to determine optimal solution in test conditions. Imitation learning is one category of learning algorithms that benefits from human expert guidance thorough learning process to extract policies. These approaches are also time-consuming in large-scaled problems.

Increasing the number of tasks and robots expands the size of the state space dramatically and affect the performance of approaches. Since this process requires high computational time, this paper describes a method that reduces the size of the state space explored, by partitioning the tasks to the number of robotic agents. Therefore, the problem becomes task allocation for a single agent for each cluster, thus the computational cost will be reduced. Here, a k-means method [12] is used to partition the tasks. K-means aims to partition the input data to specific number of clusters. This paper will focus on large-scale assignment problems involving hundreds of robots and tasks. First, the tasks are partitioned to the number of robots, then robots are being assigned to the clusters optimally, and finally a task assignment algorithm is executed individually and in parallel in the clusters.

The paper has been organized as follows. The second section of this paper will describe the formulation of multiple task assignment problem. In Sect. 3, task allocation using clustering method is described. In Sect. 4, a non-clustering genetic algorithm for multiple task assignment problem is proposed. In Sect. 5, an imitation learning algorithm, maximum margin planning (MMP), and clustering MMP for task allocation problem are explained. Sections 6 and 7 provide numerical simulations and a comparison between the GA, MMP, and the proposed algorithms with clustering method. Finally, Sect. 8 concludes the paper.

2 Problem Formulation

In this section, a mathematical formulation of the multiple task allocation problem is presented. Let N_a be the total number of the robots and A be the agents set with known positions that can be defined as follows:

$$A = \left\{A_1, A_2, \ldots, A_{N_a}\right\}. \tag{1}$$

The tasks set T are defined as follows:

$$T = \left\{T_1, T_2, \ldots, T_{N_t}\right\}. \tag{2}$$

where N_t is the number of the tasks. The task allocation formulation can be expressed as follows:

$$\min \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} c_{ij} u_{ij} \right). \tag{3}$$

Subject to:

$$\sum_{i=1}^{N_a} u_{ij} = 1, \quad \forall j \in \left\{1, 2, \ldots, N_t\right\}. \tag{4}$$

$$\sum_{j=1}^{N_t} u_{ij} \leq L_t, \quad \forall i \in \{1, 2, \ldots, N_a\}. \tag{5}$$

where decision variable $u_{ij} = 1$, if agent i is assigned to task j and 0 otherwise, c is the cost (distance) matrix, and L_t is the maximum number of tasks that can be assigned to each agent. The objective of a task allocation is to find the best conflict-free match between N_a agents and N_t tasks. Task allocation is conflict free if each task is assigned to just one agent. The cost function often represents a path dependent reward and the goal is to minimize the total path of robots. In addition, path independent cost functions often minimize a user-defined function by determining the priority of tasks. The task allocation is done when all of the tasks are assigned to agents.

3 Task Allocation Using Clustering Method

3.1 Clustering

The goal of clustering is to divide an input data into a finite discrete set of structures. Clustering algorithms partition the input data into a specific number of clusters. This paper describes a method that reduces the size of the state space explored, by partitioning the tasks to the number of robotic agents. Hence the main problem is divided into some simple problems, such that the problem of task allocation for multiple agents is divides into several task allocation problems, each for a robotic agent in a cluster. Hence, the computational complexity is reduced by decreasing the size of the state space. There are many methods for data clustering, such as hierarchical methods, partitioning methods, density-based methods, model-based clustering methods, grid-based methods, and soft-computing methods [13]. In this paper a partitioning method called K-means is used [12]. K-means clustering is a simple unsupervised learning algorithm for clustering analysis. The algorithm aims to partition N_t points into N_a groups in order to minimize the total distance between the centroids of each cluster and their corresponding points. The objective function defines as follows:

$$\arg\min_{S} \sum_{i=1}^{N_a} \sum_{x \in S_i} \|x - \mu_i\|^2. \tag{6}$$

where $S = \{S_1, S_2, \ldots, S_{N_a}\}$ is the set of clusters, $x = \{x_1, x_2, \ldots, x_{N_t}\}$ is the set of input data and μ_i is the center in cluster S_i. A set of initial cluster centers is chosen randomly, then, in each iteration, each point is assigned to its nearest cluster center, finally, the cluster centers are calculated again.

3.2 Assigning Robots to Clusters

After partitioning the tasks, the robots should be assigned to clusters in an optimum way, based on the distance from robots to clusters. The distance from a robot to a cluster is defined based on the robot and the nearest task of the cluster to it. The distance matrix is a $N_a \times N_a$ matrix, that N_a is the number of robots in the mission. Now, the optimum way to assign robots to the clusters should be computed. This problem is a linear sum assignment problem (LSAP). In LSAP, each row has to be matched to a different column in such a way that sum of the corresponding entries is minimized.

Many approaches has been developed to solve LSAP problem, the algorithms for LSAP are based on different approaches: first class of methods directly solves the primal problem, second one solves the dual, and third one uses an intermediate approach (primal–dual) [14]. Hungarian method [15] is the first polynomial-time primal–dual algorithm for solving the LSAP. The time complexity of the original Hungarian method is $O(n^4)$. Here, we use the Jonker–Volgenant algorithm [16] to solve this problem, which is a shortest path implementation for the Hungarian algorithm with higher performance. This method is a Dijkstra [17]-like algorithm for shortest path augmentation which is done after three levels of preprocessing: column reduction, reduction transfer, and augmenting row reduction. These steps are the main contribution and the most time-consuming part of this method. The Dijkstra-like algorithm for shortest path augmentation grows an alternating tree for an unassigned cluster, to find alternating paths including possibly augmenting path starting from that cluster [14]. A special implementation of Dijkstra and a total time complexity of $O(n^3)$ makes this method suitable for fast assignment of robots to the clusters.

3.3 Clustering Genetic Algorithm

Genetic algorithm (GA) mimics the process of natural selection and an iteration evolutionary process for finding optimal solutions. The set of feasible solutions known as chromosome in the GA represents as follows. Each chromosome with the length of the number of tasks in the cluster expressed as C, and C_i is the ith element of C representing the ith task on the schedule of the agent. To generate a new solution candidate, GA operators such as selection, crossover, mutation, and replacement are executed. The tournament selection method is adopted as selection operator. Two chromosomes from the population are chosen with the selection method as the input for crossover operator. Partially Matched Crossover (PMX) [18] method is adopted as crossover operator. The PMX method is widely used for permutation chromosomes. Randomly two non-equal points between the zero and length of the chromosome are selected, a sub-string between the two random points of the parent chromosome is donated to the offspring at the same position, and then it affects cross by position-by-position exchange operations. The mutation crossover consists of swap, reversion,

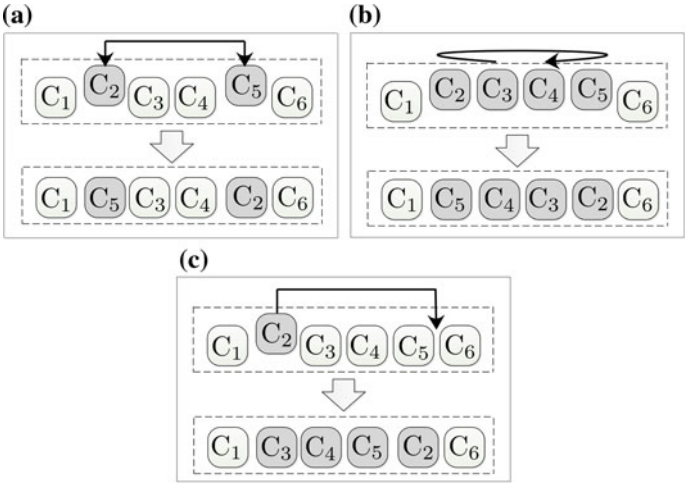

Fig. 1 Mutation operators: **a** swap operator, **b** reversion operator, **c** insertion operator

and insertion. The swap operator exchanges the value of two random points in the chromosome, the reversion operator reverses a swath of the chromosome, and insertion operator transmits the value of a random point to another point in chromosome. Figure 1 explains the mutation operators. The algorithm has been executed several times for different number of tasks to find the iteration number when there is no more change in the cost. A polynomial is fitted to these points to find the termination criteria for further executions of the algorithm with any arbitrary number of tasks.

4 Non-clustering Genetic Algorithm for Multiple Task Allocation Problem

GA is executed to find the suboptimal solution. The set of feasible solutions known as chromosome in the GA represents as follows. Each chromosome expressed as C, which consists of two parts, the first part expressed as CA with the length of N_a, represents the number of tasks for each agent, the ith element of CA expressed as CA_i is the amount of tasks for agent i. The second part expressed as CT with the length of N_t is a permutation of doing tasks for agents. CT is made from concatenation of different sequences, each with length of CA_i, $i = 1, 2, \ldots, N_a$, representing a permutation of allocated tasks to the agent i. Each of these genes has a value representing task number CT_j, $j = 1, 2, \ldots, N_t$. The structure of chromosome is shown in Fig. 2. Two parts of the chromosome are being separated from each other to perform crossover and mutation. Crossover and mutation operators decide to perform action only on one part at each time with a constant probability. GA operators, such as selection, crossover, mutation and replacement are exactly the same with the previous section.

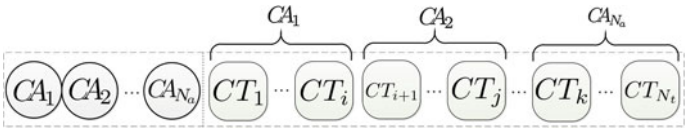

Fig. 2 Structure of the chromosome

Since the sum of elements of the first part is constant and equal to the number of tasks, a little change has been done to the crossover operator to keep the sum of elements constant.

5 Imitation Learning

Learning a mapping between a problem's world states and actions is the most important problem in many robotic applications. This mapping which is also called a policy enables a robot to select an action based upon its current world state. With imitation learning, policy is extracted from a set of demonstrations directly provided by a human or through human guidance [19]. Using imitation learning to extract policies from a set of demonstrated allocations is a powerful approach for reducing the complexity of search spaces for learning by eliminating impossible solutions [20].

In multi-robot task allocation domains, explicitly modeling task features well enough so that they are used in allocation process may be intractable, but a human expert may be able to quickly gain some knowledge about the form of the desired solution [21]. In the next section one of the imitation learning frameworks, Maximum Margin Planning is discussed.

5.1 Maximum Margin Planning for Multiple Task Allocation Problem

Maximum Margin Planning (MMP) is an imitation learning approach for structured prediction over the space of policies [22]. μ represents a particular set of allocations from the space of possible allocations G. $f \in \mathbb{R}^d$ is feature vector for each possible task allocation and F is an accumulation matrix of these features. The product $F\mu$ represents the accumulation of features encountered by following the policy μ [22]. Training data set is specified as $D = \left\{\left(F_i, G_i, \mu_i, l_i\right)\right\}_i^N$. Each training example i consists of a set of tasks, agents, and the expert allocation $\mu_i \in G_i$. Each example includes a loss field l_i, so that $l_i^T \mu$ quantifies how bad a policy μ is compared to the demonstrated policy μ_i. The goal is then to learn a set of weights ω, so that each demonstrated allocation is better than every other possible allocation for the same scenario [21]:

$$\omega^T F_i \mu_i > \omega^T F_i \mu + l_i^T \mu \qquad \forall i, \mu \in G. \tag{7}$$

If constraint (7) holds for all allocations in G, it must hold for the best allocation, thus the only constraint to consider is the tightest one, corresponding [21]:

$$\mu_i^* = \max_{\mu \in G_i} \left[\left(\left(\omega^T F_i + l_i^T \right) \mu \right) \right]. \tag{8}$$

Solving this problem using subgradient method [21]:

$$g_\omega = \lambda\omega + \frac{1}{N} \sum_{i=1}^{N} \left(F_i \mu_i^* - F_i \mu_i \right) = \lambda\omega + \frac{1}{N} \sum_{i=1}^{N} \left(F_i \Delta^\omega \mu_i \right). \tag{9}$$

Subgradient update rule then becomes [21]:

$$\omega_{t+1} = \omega_t - \alpha g_\omega. \tag{10}$$

For learning rate α,intuitively, this gradient update rule increases the reward (in feature space) on the demonstrated allocation and decreases the reward (again in feature space) on the chosen allocation. In order to use demonstrated allocations in task allocation mechanism, a bias term is introduced into task execution order, which uses the learned feature weighting vector w. Therefore the profit used for selecting task for execution is [21]

$$profit = cost + bias. \tag{11}$$

cost is a constant value, added to profit and the bias:

$$bias = w^t f^k. \tag{12}$$

f_k is features vector for task k.

5.2 Clustering Maximum Margin Planning

In order to use clustering advantages in imitation learning for multi-robot task allocation, at first, tasks are clustered with K-means algorithm, and then each robot starts to execute own clusters tasks according to learned policy with maximum margin planning. Tasks execution priority is chosen in order to maximize the total profit, calculated in Eq. (11).

6 Numerical Results for GA

In this section, a comparison including numerical results between clustering GA and the non-clustering GA method for solving multiple task allocation problem is represented. Simulations for both methods are performed in several cases. As explained in Table 1, in each case different number of tasks and agents are included. Coordinates

Table 1 Conditions of numerical solutions

	Case 1	Case 2	Case 3	Case 4	Case 5
Number of tasks	20	50	100	200	200
Number of agents	4	8	10	15	25

Table 2 Numerical results of task assignment for clustering GA

	Case 1	Case 2	Case 3	Case 4	Case 5
Clustering time (s)	0.141	0.156	0.156	0.157	0.157
Assigning time (s)	0.016	0.016	0.016	0.016	0.017
Algorithm time (s)	0.864	2.219	5.365	12.344	8.406
Total cost	1814.5	2414	3951.6	5607	5230.9

Table 3 Numerical results of task assignment for non-clustering GA

	Case 1	Case 2	Case 3	Case 4	Case 5
Algorithm time (s)	1.812	20.531	87.812	226.250	132.078
Total cost	1814.5	2414	3951.6	5607	5230.9

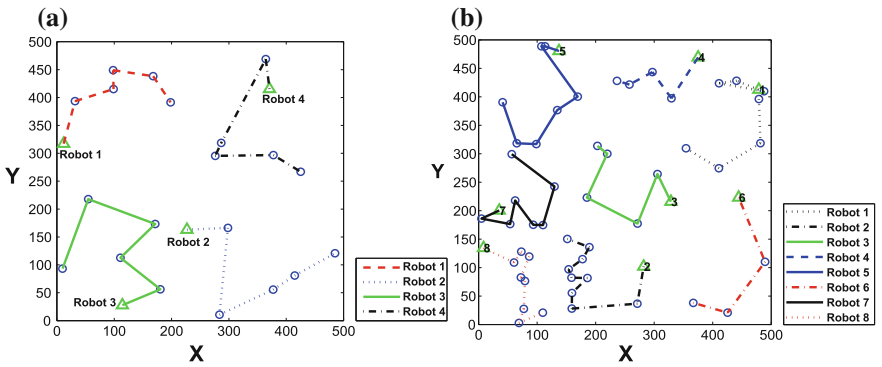

Fig. 3 Final solutions for task allocation with clustering GA for **a** case 1 and **b** case 2; the *circles* indicates the coordinates of tasks and the *triangles* indicates the initial coordinates of robots

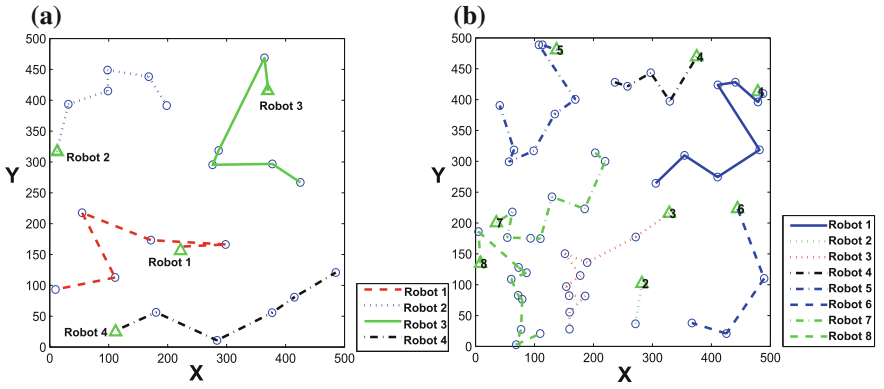

Fig. 4 Final solutions for task allocation with non-clustering GA for **a** case 1 and **b** case 2; the *circles* indicates the coordinates of tasks and the *triangles* indicates the initial coordinates of robots

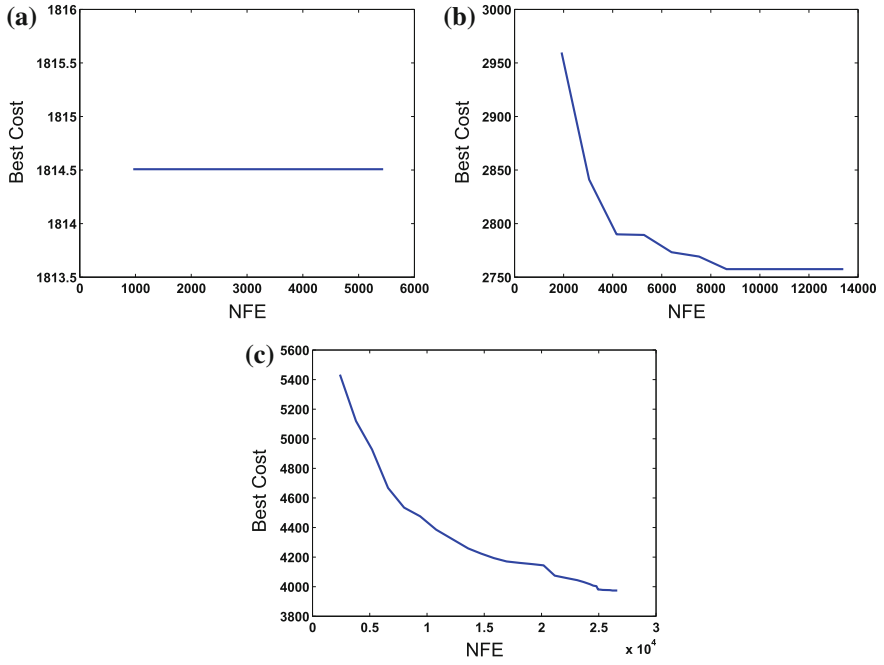

Fig. 5 Total cost of the mission with respect to NFE for clustering GA for **a** case 1, **b** case 2 and **c** case 3

of tasks and agents are generated randomly, the same coordinates are used for both algorithms. The simulations were performed on a laptop computer which has a dual core 2.0 GHz CPU and 6 GB RAM using MATLAB. The MATLAB parallel computing toolbox is used to execute the algorithm for all clusters in parallel. The

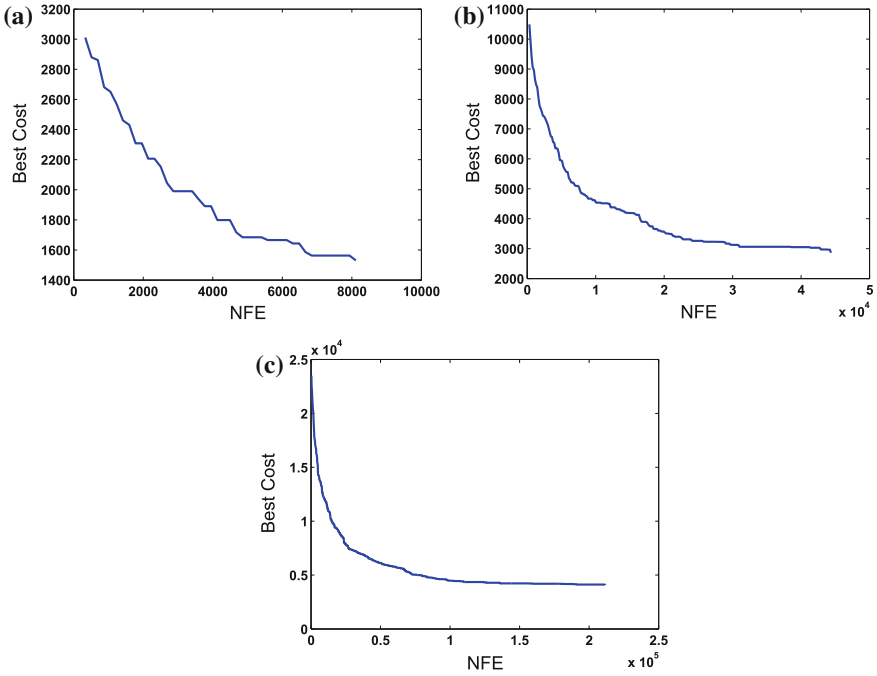

Fig. 6 Total cost of the mission with respect to NFE for non-clustering GA for **a** case 1, **b** case 2 and **c** case 3

termination criterion for the proposed method is explained in Sect. 5. The non-clustering GA method is executed until it reaches the total cost of our method for each case. Total cost is the total distance of the path of all robots. In Table 2, process time of clustering, process time of assigning robots to the clusters, computation time of the algorithm, and the total cost of mission are represented. Table 3 shows the cost and process time of the assignment problem with simple GA. The final solutions of all cases for our proposed method and the GA without clustering are shown in Figs. 3 and 4, also Figs. 5 and 6 represents the total cost of the mission with respect to the number of fitness evaluation (NFE) for all cases. The algorithm process times for clustering GA and non-clustering GA with different number of agents and tasks are illustrated in Figs. 7 and 8. According to the numerical results in all cases, our proposed method produced solutions very fast, whereas the simple GA took a lot of time to reach the same total cost, and this time increases exponentially when the number of tasks increases.

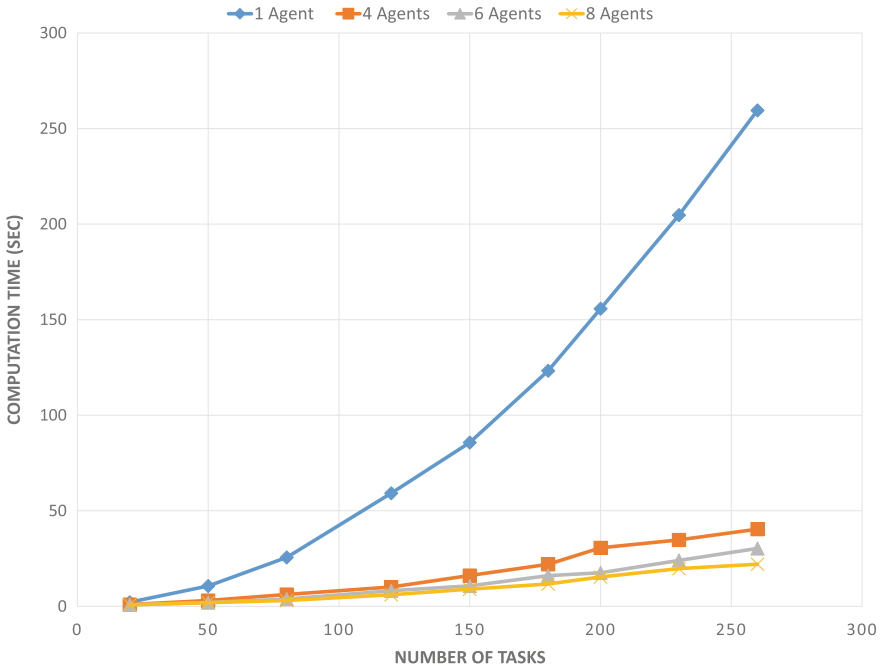

Fig. 7 Computation time for clustering GA with different number of agents and tasks

Fig. 8 Computation time for non-clustering GA with different number of agents and tasks

7 Numerical Results for Imitation Learning

In this section, a comparison between MMP and the MMP with clustering for solving multiple task allocation problem is represented. A scenario for task allocation for buildings in fire [21] is represented and will solved with imitation learning. Each task has a feature vector that express information about each individual task in the environment. Here features that are computed per task, consists of 2 elements: (1) Size: building size represented with size of task square in figures (2) Health: current building health represented in percent in figures. Human experts in example demonstrations, represent optimal task allocations, for example in this scenario tasks closer to robot with bigger building size and healthier buildings are prior to others. Using these training set, policy is extracted and used in future test conditions.

Experimental results for all cases in Table 1 are represented. Tables 4 and 5 show the average profit of all robots profit and total time to finish all robots tasks. The

Table 4 Total time to finish the mission

	Case 1	Case 2	Case 3	Case 4	Case 5
Total time of non-clustering MMP	10.122	15.435	34.1094	43.1214	53.078
Total time of clustering MMP	0.3188	0.4188	0.6188	0.8188	0.9324

Table 5 Average profit of all robots

	Case 1	Case 2	Case 3	Case 4	Case 5
Average profit for non-clustering MMP	348.09	352.22	358	362.07	365.87
Average profit for clustering MMP	348	352.01	358.01	361.07	365.23

Fig. 9 Computation time for clustering imitation algorithm with different number of agents and tasks; because each robot executes each clusters tasks in parallel with other robots, MMP with clustering overcome the algorithm without clustering in total algorithm time in all cases

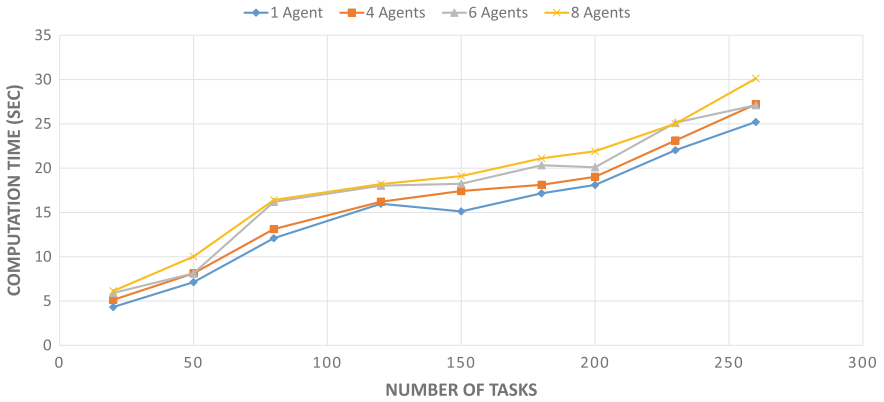

Fig. 10 Computation time for non-clustering imitation algorithm with different number of agents and tasks; approximately in all cases algorithm time increases about 50

algorithm process times for clustering and non-clustering imitation algorithm with different number of agents and tasks are illustrated in Figs. 9 and 10. As the results show, maximum margin planning with clustering overcome the algorithm without clustering in total algorithm time. Total cost and solutions found for both algorithms are nearly the same. Clustering time and assigning time are identical to GA with clustering.

8 Conclusion

In this paper, a task assignment method is presented to deal with large number of tasks and robots. The proposed method is able to assign a large number of tasks to robots in a high efficient time. The effectiveness of our approach is demonstrated by numerical simulations. Especially in large-scaled task allocations, using the simple GA method to reach the final cost similar to our proposed method, takes longer time.

References

1. Wu, F., Zilberstein, S., Chen, X.: Online planning for multi-agent systems with bounded communication. Artif. Intell. **175**(2), 487–511 (2011)
2. Avellar, G.S., Thums, G.D., Lima, R.R., Iscold, P., Torres, L. Pereira, G., et al.: On the development of a small hand-held multi-uav platform for surveillance and monitoring. In: 2013 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 405–412, IEEE (2013)
3. Iscold, P., Pereira, G.A., Torres, L.A.: Development of a hand-launched small uav for ground reconnaissance. IEEE Trans. Aerosp. Electron. Syst. **46**(1), 335–348 (2010)
4. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005)

5. Chandler, P.R., Pachter, M., Rasmussen, S., Schumacher, C.: Multiple task assignment for a uav team. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, p. 4587 (2002)
6. Richards, A., Bellingham, J., Tillerson, M., How, J.: Coordination and control of multiple uavs. In: AIAA Guidance, Navigation, and Control Conference, Monterey, CA (2002)
7. Schumacher, C., Chandler, P., Pachter, M., Pachter, L.: Constrained optimization for uav task assignment. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference, pp. 1–14. American Institute of Aeronautics and Astronautics Inc, Reston, VA, USA (2004)
8. Eun, Y., Bang, H.: Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. J. Aircraft **46**(1), 338–343 (2009)
9. Shima, T., Rasmussen, S.J., Sparks, A.G., Passino, K.M.: Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. Comput. Oper. Res. **33**(11), 3252–3269 (2006)
10. Potvin, J.-Y.: Genetic algorithms for the traveling salesman problem. Ann. Oper. Res. **63**(3), 337–370 (1996)
11. Cruz Jr, J.B., Chen, G., Li, D., Wang, X.: Particle swarm optimization for resource allocation in uav cooperative control. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, pp. 1–11, Providence, USA (2004)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297, Oakland, CA, USA (1967)
13. Rokach, L.: A survey of clustering algorithms. In: Data Mining and Knowledge Discovery Handbook, pp. 269–298. Springer (2010)
14. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment Problems. Revised Reprint, Siam (2009)
15. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Res. Logist. Quart. **2**(1–2), 83–97 (1955)
16. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing **38**(4), 325–340 (1987)
17. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
18. Goldberg, D.E., Lingle, R.: Alleles, loci, and the traveling salesman problem. In: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 154–159. Lawrence Erlbaum Associates, Publishers (1985)
19. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
20. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Springer Handbook of Robotics, pp. 1371–1394. Springer (2008)
21. Duvallet, F., Stentz, A.: Imitation learning for task allocation. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3568–3573. IEEE (2010)
22. Ratliff, N.D., Silver, D., Bagnell, J.A.: Learning to search: Functional gradient techniques for imitation learning. Auton. Robots **27**(1), 25–53 (2009)