Applied Intelligence (2022) 52:2189–2211
https://doi.org/10.1007/s10489-021-02483-3

Multi-robot exploration in task allocation problem

Reza Javanmard Alitappeh[1] · **Kossar Jeddisaravi**[1]

Accepted: 28 April 2021 / Published online: 5 June 2021

Abstract

Task allocation is an important problem in multi-robot system which can be defined with different setup for different application, i.e. coverage, surveillance and mining mission in static or dynamic scenarios. Our focus in this paper is *exploring environment to accomplish tasks distributed over the environment by minimizing overall cost of the system*. This problem is defined as a NP-Hard problem, thus will be more challenging in larger environments containing many robots and tasks. To solve multi-robot task allocation in very large environment we propose a new *deployment-based framework*. Our proposal divided the problem into two sub-problems: *region partitioning* and *routing problem*. This decomposition eases considering our *problem specification* in multi-robot system which are not easily considerable in other approaches, i.e *distribution of the tasks* or *robots' initial position*. Load balancing is done globally by deploying robots in a proper location of the environment and assigning sub-regions among them. Sub-regions contains set of points, where the goal is visiting all the points individually by one of the robots. On the other hand, after deploying the robots, routing techniques can be simply applied to find shortest and safest paths for every robots. To search for solutions in this NP-hard problem, two methods are built on a tailor-made multi-objective scheme of Genetic Algorithm (GA) with a different setup and search operators, and a reinforcement learning approach. Simulation results testify the performance of our methods in comparison to existing ones.

Keywords Task allocation · Multi-robot exploration · Region partitioning · Multi-robot routing problem · Q-learning

1 Introduction

Multi Robot Systems (MRS) have recently received a great deal of attention in exploration and coverage applications. This is due to advantages of using MRS, so that a group of robots can explore the environment faster with more robustness and less redundancy in comparison to single robot system in complex tasks. Moreover, in a MRS, collaboration and coordination between robots balances total load in the system. For instance, in our everyday life delivery service multi vehicle (robots) has become a new trend, where trucks dispatch from warehouses and deliver goods to the markets for daily needs. Similar to delivery, many industrial and service applications such as environment monitoring [37], traffic surveillance, crime monitoring, anti-terrorist mission [76], rescue in disaster mission [16, 28, 46, 68], inspection [41], SCAT (simultaneously coverage and tracking) [29, 55] involve task assignment, exploration, coverage and routing [12, 31, 61, 75]. In most of them, as a basic requirement, teams of robots explore the environment and execute tasks in continuous or discrete ways. A solution is interested if maximizes/minimizes objectives defined under the problem specification, i.e. maximizing covering area and minimizes energy consumption. In this regard, a fair distribution of the tasks among robots yields a lower cost for exploring and servicing tasks.

Among variations of the multi-robot exploration application, this paper presents a solution for Multi-Robot Task Allocation (MRTA) problem, where robots must visit distributed tasks over an environment [5, 11, 24, 25, 67]. However, for simplification we assumed *tasks* are set of *points* in the environment that must be visited by robots. Although what exactly the tasks are and how they will be done are not in the scope of this paper, but these points, for example, could be places to search for an object by spending a fixed time and energy on every point. Accordingly by distributing robots over the environment, each of them will explore and service set of tasks close to its location.

✉ Reza Javanmard Alitappeh
Rezajavanmard64@gmail.com

[1] Department of Electrical and Computer Egineering, University of Science and Technology of Mazandaran, P.O. Box 48518 - 78195, Behshahr, Iran

Many researchers in this area have been trying to find a global solution for variations of task allocation problem, although it is classified as an NP-hard problem. In this paper, task allocation problem is approached by considering many particularities (features) of multi-robot system e.g. robots' location, tasks distribution, shortness and safety of the paths to visit every task point, and so on. To do so, we decompose the entire problem into sub-problems: *region partitioning* and *routing*. Later a tailor-made meta-heuristic (GA) and a Q-learning approach are applied to find a near optimum solution. We also assumed that the environment has a finite size like an university, a governmental department, or a city, and suitable for a centralized system, where infrastructure i.e. communication network is available for all the robots.

An example is shown in Fig. 1, where task points are distributed in the environment. They are assigned to the closer robots, where each robot has its own tasks to visit inside its dedicated region. Thus, the total distance between task points and robots is minimized. Given the above example, to visit every point robots need to move and explore the map. Therefore, first we need to find a suitable robots deployment for starting exploration, which also needs to take map partitioning into account given robots' position. Later, robots traverse routes (preferably the shortest one) for visiting points in their region.

For partitioning, by using Generalized Voronoi Diagram (GVD) we represent the map in form of a graph, which yields a safe route for each robot as well. To deal with collision in robots' motion *potential field* is applied. All these techniques are jointed in our proposed framework which is called *multi-robot exploration for task allocation*.

The main contributions of this work are divided in several folds. Although deployment or graph partitioning, task allocation and routing problems are well-known in the literature and are tackled individually beforehand, in this work we propose an end-to-end multi-objective framework solving all of them at once. Moreover, as a solution for a realistic combinatorial problem, it considers problem' specifications which may not be found in the literature of aforementioned problems. In other words, our framework considers parameters of the input problem, i.e. dynamic of tasks distribution, robots' initial position, shortness of the path etc. Thus, robots will be deployed given the distribution of the tasks, so that they will have a balanced tasks in their region. Moreover, our task allocation framework is built upon a modified GA to search the problem space in a precise manner. In large problems which are the cases in today's big data space, our solution will be comparable with deterministic approaches which struggling to solve NP-Hard problems. This is achieved by our proposal for modeling the entire problem into two sub-problems and solve each separately. As another contribution, we also proposed a Q-learning algorithm designed on the problem specification, to learn finding the solution during searching phase using award mechanism. Finally, we simulate the proposed method in some real scenarios and compare with similar methods, which testify the performance of the proposed method in different experiments conditions.

The paper is organized as follows. Since the proposed framework includes different concepts, in Section 2 a review of the literature in multi-robot deployment, exploration, task allocation and routing problem are explained, at the end exploration using Q-learning is reviewed. In Section 3, we defined the problem formulation. Section 4 is about the proposed methodology, containing objective functions, representation of the problem and proposed algorithms. Finally, in Section 5, simulation results and comparison to other methods show the performance of the proposed framework.

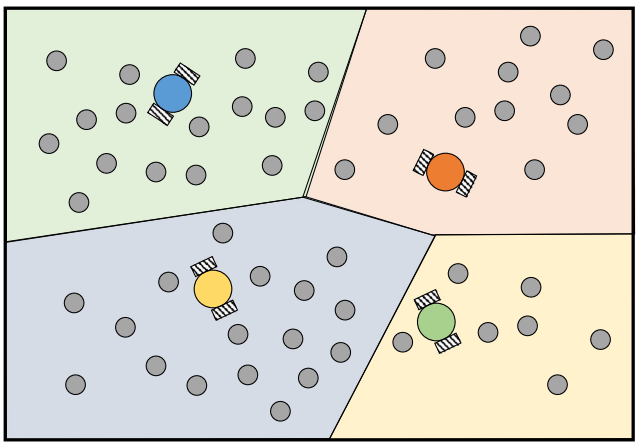


Fig. 1 An example of task allocation with four robots and their dedicated region. Each robot is responsible to the tasks (points) inside its regions

2 Related work

In this paper, we addressed a problem that involves several well-known subjects of classical problems in graph theory and robotics. Thus, we consider three lines of related work: task allocation, exploration and deployment, and routing problem. However, there are some papers for multi-robot task assignment which we first take a look to them.

2.1 Multi-robot task allocation (MRTA)

In general, multi-robot task allocation (MRTA) is how to allocate subtask to each robot, therefore in our problem task points (as subtasks) must be distributed among robots, and each of which will be serviced by a single robot with equal time needed, in which the entire load would be balanced.

Since it is an NP-hard problem, solving problems with large search space makes challenges for deterministic approaches. To deal with that, usually researchers decompose the problem into subproblems, or apply a meta-heuristic technique.

In [24], authors proposed a multi-robot cooperation approach using genetic algorithm to allocate task among robots in a robotcup rescue simulation environment with known robots and tasks. Given their simulation, they claim that the approach has less computation, better real-time and stronger ability to seek for the optimal result. In contrast to our approach, author did not consider distribution of the tasks and also not using a deployment scheme to initial the robots in a proper location, which will make tasks be done faster. In a similar application in [63] a disaster rescue scenario task allocation is done using two multi-objective coalition formation. Although they deal with several challenges i.e. temporal constraint, heterogeneity of robots, dynamic environment etc, they did not consider time and energy consumption for the path that robots must move on them.

Multi-robot task allocation with focus on manufacturing purpose is considered in [52]. Authors developed a path finding technique based on Traveling Salesman Problem (TSP) solution merged by Dijkstra algorithm for calculating bids. They consider cost and time in order to integrate path planning into a robot's bids for tasks. In [59] a concentration for multi-humanoid robots in task allocation problem is addressed. They considered four objectives, namely energy consumption, total tasks' accomplishment time, robot's idle time and fairness. Thus, after partitioning the environment, they applied NSGA III to assign task between humanoid robots given the objectives they defined.

Authors in [2] customized multi-objective optimization model for product task allocation. In [41], authors combined memetic and genetic algorithm for multi-robot task allocation problem where for completing a given task 2 robots are needed to work cooperatively. Researchers in [69] worked on hybrid Multi Objective GA (HMOGA) to minimize a multi-objective problem in assembly line resource assignment and balancing. Two objectives were defined; cycle time and the cost per time unit of a line for a fixed number of stations. With similar purpose but in cloud manufacturing environment, [40] provided a new hybrid GA method to solve the optimal allocation of computing resources problems. This problem can be seen from job shop scheduling problem as well, where a set of jobs have to be transported between machines by several transport robots [48]. Author applied hybrid meta heuristic approach for job shop and robot routing problem. In which, Neighborhood-based Genetic Algorithm (NGA) for a global exploration and tabu search for a precise search. In this problem time for job assignment and completion are the main concern.

In contrast to cited literature, our approach is applicable not only on isolated environment i.e manufacturers etc, it works in dynamic environment where location and distribution of the task points might not be fixed beforehand.

Task allocation in a large number of tasks and robot is proposed in [18], where after clustering the entire task into number of robots, GA and imitation learning algorithm are applied for task allocation individually in each cluster. While they used a simple k-means algorithm to cluster the task at the beginning, in our approach we are using an adopted method upon Voronoi and GVD. In our implementation we compared both approaches. In [67] a solution for Multi-Robot Dynamic Task Allocation problem is proposed. They defined clusters containing one or more robots. Genetic algorithm is used to find a solution of predicated task allocation given objectives authors defined, later the solution is considered as a bid in an auction between the robots. In comparison to the algorithm we proposed in this paper, their approach is more theoretical than practical for real robot scenarios. Unlike this paper, in [62] author proposed a solution to the task allocation problem in a team of UAVs in a decentralized way. Task allocation is decided by robots themselves, however, it was assumed that communication among the UAVs is complete and never fails. In a retirement home application, a deterministic approach upon mixed-integer programming (MIP) and constraint programming (CP) is proposed for planning and scheduling of multiple social robots [5]. In [11], a formal restricted model of MRTA over time for vacancy chain scheduling is presented, which divides global system performance into individual robot contributions. Authors in [25] applied GA for task scheduling between robots, they also used A* algorithm for path planning with collision avoidance. In a different way in [28] author applied cellular automata for allocating spatial–temporal tasks in multi-agent systems. Task allocation for search application is studied in [71], where an auction based strategy is applied on graph representing the environment. Retrieving task after finding the target is one of the concern in this work which makes the paper different in comparison to others. Given the importance of task allocation in many multi-robot systems, a comprehensive review challenging aspects of MRTA problem can be found in [33] and [27].

In this study, for a different purpose objectives such as tasks' priority is not our concern, a practical and generic framework is proposed where take real world problems' parameters into account. Moreover, a deployment scheme is applied for assigning task between the robots, where in contrast to literature considers tasks' dynamic distribution upon a new distance metric.

2.2 Multi-robot exploration and deployment

In multi-robot deployment problem, a team of robots are distributed in an environment in order to explore and execute part of an entire mission. A common utility of deploying scheme is to solve coverage problem, whereas robots cover their dedicated region after the distribution.

Techniques for deployment can be divided in two categories; in the first one, forces in the environment lead the robot to do the desired task. For instance, attractive and repulsive forces were applied in a mobile sensor network to control robots deploying in [56]. A low-level control is developed by authors in [53] based on attractive and repulsive forces as well. In their work, a group of homogeneous robots is distributed in an uncluttered environment to observe multiple moving targets. The second category is based on coverage control approach defined with respect to the centroids of Voronoi cells resulting from the Voronoi tessellation of the domain. Authors in [10], presented a distributed approach for optimally deploying a uniform robotic network in a domain based on an optimized quantization framework derived in [43]. Each robot follows a control law, which is a gradient descent algorithm that minimizes the functional encoding the quality of the deployment. A variation of Cortes' work has been considered, including non-convex environment with heterogeneous robots by [54], discrete partitioning and coverage optimization algorithm in [21], and with short-range communication in [14]. Self-triggered coverage is addressed in [50], a discrete control setup is proposed by [4] and [73]. Authors in [22] presented a multi-objective approach for single robot searching purpose. They focus on path planning by solving Chinese Postman Problem (CPP). The same authors proposed a multi-objective multi-robot deployment approach in [20], where the environment is dynamic and robots need to be redeployed periodically. In part of our proposal, the deployment scheme with a major modification inspired from this work. Although in this work the objective was to move the robot toward the position where the deployment function is minimized, without considering the robots passing routes. However in our proposal the paths of the robots are also our concern, which means robots are forced to move on the paths in our GVD graph representation in such way that the entire graph nodes and edges must be visited at the end. Authors in [12] addressed exploration in unknown environments using a discrete event system with a supervisory control to monitor robots movement toward target points. Exploration for target searching purpose is addressed in [65], where authors applied a hybrid Fruit Fly Optimization Algorithm(FOA) and Particle Swarm Optimization (PSO) algorithm to find best next optimal robot position.

In the current paper different from cited works, for deployment we partition the map by considering the distribution of task points and length of the routes robots need to visit all the task points in the environment.

2.3 Multi-robot routing

When a robot wants to visit a set of task points, it needs to find the shortest route, somehow it can be path planning [15, 34]. A similar subject to our problem can be found in vehicle routing problem. This problem received a great deal of attention due to the applicability in the real world. Several algorithms are available for this problem. As far as this problem is hard combinatorial, solutions based on exact methods have a poor performance in larger scenarios. Thus numerous heuristics algorithm beside many different variations of this problem have been proposed in the literature [39, 52, 70]. Among them, we focus on Multi-Vehicle routing, where vehicles depart from different locations, namely Multi-Depot Vehicle Routing Problem. This problem is addressed in [51], where authors decomposed the problem into subproblems and applied a coevolutionary algorithm in parallel scheme to solve the problem. In a different version author present a meta heuristic approach to solve Multi-depot vehicle routing problem with simultaneous deliveries and pickups (MDVRPSDP) in [38]. Authors in [68] solved multiple-depot multiple traveling salesman problem (MD-MTSP) using fuzzy logic, which works in two phases: assignment and tour construction. An exact method developed for Multi-depot vehicle routing problem (MDVRP) under capacity and route length constraints by Contardo et al. in [9]. Heterogeneity of the vehicles is considered by Salhi et al. in [60], where several local search algorithm are applied to find the solution. A version of this problem is called multi-depot open vehicle routing problem (MDOVRP) [35], where vehicles are not required to return to the departure point after delivering the goods to the customers.

As we represent our map into a graph, in the sense of routing, our problem is similar to MDOVRP, but differently robots must cover all the edges to visit all the task points distributed along the edges. Thus, beside shortness and safety routes contain all the edges in robots' assigned regions (or sub graphs), although we can avoid robots passing unnecessary edges i.e. edges in a route with no task point along and after it in the route, or unvisited edges after visiting all the task points.

To the best knowledge of authors the combinatorial task allocation, exploring and routing problem considering our problem specification using a new end-to-end framework in this paper addressed for the first time as we reviewed in the literature. Although there exist some research in area coverage [1, 72] where the solution can be used in our problem as well. But as we will show in experimental result they do not work well since they do not consider the objectives which are our concern. We propose a multi-objective meta-heuristic framework to find the best robots'

deployment in the environment by partitioning the graph given the distribution of the tasks, minimizing the length of total path traversed by every robot in its sub-region to visit task points and maximizing robot's motion safety with minimizing the probability of robots' collision. It should be noticed that our focus is developing high level strategy for multi-robot motion, instead of localization, mapping, communication or other multi-robot concerns.

2.4 Exploration with Q-learning

As a goal-oriented method, reinforcement learning is a model-free approach, where differ form supervised and unsupervised learning does not need labeled dataset. However, agents in this approach learn to make sequence of decisions given their current state and selected action. To do so, a reward strategy is applied, where agents get either rewards or penalties for the selected actions. As one of the most popular version of reinforcement learning, here we applied Q-learning, where seeks to learn a policy that maximizes the expected value of the total reward over any and all successive steps. There are many applications of reinforcement and Q-learning in robotics and computer games, where robots or agents can learn from scratch i.e. [7].

In several works authors combined meta heuristic algorithm with Q-learning for different purpose [13, 58]. In [13], multi-robots path planning in clutter environment using Q-learning and Particle Swarm Optimization (PSO) is addressed. As their objectives to be minimized, path length, arrival time and turning angle were taken into account. PSO is used for a selecting the next best action in hybridization of Q-learning. In another work, firefly algorithm is joints with Q-learning, where its parameters are learned over learning phase [58]. Similar objectives are considered in [30] for mobile robot path planning. To do so, they extended Q-learning algorithm to minimize traversed time, number of states and robot's turns. Authors in [7], proposed an incremental Q-learning for mobile robot motion planning over a PID controller. Geographic routing in a sensor network is addressed in [26], where routing is needed among number of distributed sensors in an area. In order to increase convergence speed in Q-learning, an approach is developed by authors in [44]. They applied flower pollination algorithm to improve Q-learning initialization.

In [3], author proposed a combiatorial optimization framework based on neural network and reinforcement learning with focus on the TSP problem in 2 dimensional Euclidean space for the city coordinates. To optimize the parameters of the pointer networks which contains two recurrent networks, policy gradient is performed. Although the proposed framework can be generalized to solve other problem than the TSP, e.g. the Knapsack, to apply this framework in our problem we need to modify the approach to consider our problem's specifications. Furthermore, retraining will be a big issue as far as in our application environment might changes frequently. The same issue is applied for the extension of this work in [47], where authors addressed VRP using neural network and RL. They solved the limitation of the previous work by considering system variation on time using an attention mechanism on top of RNN. In a similar scheme author in [32] improved training algorithm by changing the attention layer on the pointer network. All these approaches are designed for a single agent(vehicle) in pure routing problems, i.e. TSP, VRP, which is not the case in our problem.

In a different scheme, given the recent success of deep learning, deep Q-learning (DQN) is proposed upon deep neural network [45]. In this way, deep network approaches improved solutions for complex robotic tasks, for example, in [17], a 7-DOF manipulator is controlled by training images to the network as input and motor velocity as output. Robot navigation using depth data as input of DQN and neural network is proposed in [64] and [42], respectively. Moreover, In [23], authors applied DQL for robot path planning and obstacle avoidance.

In the aforementioned works, researcher either adapted Q-learning in their specific problem, or improved Q-learning algorithm given modifications they proposed. Similarly, in our work, we modified standard Q-learning according to the specification we have in our problem, i.e. graph representation and multiple objectives.

3 Problem formulation

Given a team of mobile robots and a set of task points in an environment, the main goal is to allocate tasks among robots. Robots move in the environment and visit all its corresponding task points for i.e. searching purpose. We assume that the map of the environment and task points are known for all the robots and a centralized system performs task allocation. We decompose the original problem into following:

Problem 1 (Region partitioning (task allocation)) An environment $\Omega \subset \mathbb{R}^2$ contains task points $T = \{t_1, \ldots, t_m\} \in \mathcal{Q}_{free}$ (where $\mathcal{Q}_{free}$ is free configuration space, $\mathcal{Q}_{free} \subset \Omega$), $|T| = m$, and a team of n robots. In this scenario, the objective is to find robots' initial position $P = \{p_1, \ldots, p_n\} \in \mathcal{Q}_{free}$, and partition the environment into n subregions based on a deployment scheme. Task points in each subregion are allocated to one of the robots. Partitioning runs in a way that the load of executing tasks i.e. the time of visiting and performing tasks are balanced among the robots. We represent the environment by a grid graph, so by solving graph partitioning we allocate nodes containing task to the robots.

Problem 2 (Routing) In the environment Ω with presence of obstacles, robots need to find routes to visit task points, which are not only *short* but also *safe*. To solve this problem, we create GVD graph over the environment that covers all the area, and find the route passing all the edges in GVD, thus robots visit task points when moving on GVD graph. Both indicated problems are addressed in the end-to-end proposed framework upon a tailor-made GA.

4 Proposed framework

As shown in Fig. 2, in the proposed method, for simplification we construct two graphs given the input map: grid graph (G_{Grid}) and GVD graph (G_{GVD}). While the earlier is used for partitioning, the later one is for routing. After partitioning the graph, robots follow the routes in their allocated region to visit task points inside the region. In this section we will go through detail of each step.

4.1 Map representation

In the proposed framework, from a continuous space the input map is converted to two discrete graphs. To create a grid graph, we applied a similar approach to [20], where based on an occupancy grid and cell discretization, graph $G_{Grid} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is defined, $\mathcal{V}$ is vertices, $\mathcal{E} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$ is edges and $\mathcal{W}$ is weights of graph which are specified by Euclidean length between cells (See Fig. 3a). As can be seen, the map is discretized into cells so that the number of cells is proportional to a default rate α. (α is defined by the robot's size). It should be noticed that task points are located inside cells. Beside grid graph G_{Grid}, we defined a graph G_{GVD} based on GVD. The Generalized Voronoi diagram (GVD) is one of the most famous roadmaps for path planning. The main advantages of this roadmap is safety, which can be applicable in exploration of cluttered environments. The definition of GVD is given in the next lines.

Let set $\mathcal{Q}_{free}$ in the environment $\Omega \subseteq \mathbb{R}^2$ represent the free configuration space as defined in [8], where the robot can move freely without colliding with obstacles. The Voronoi diagram is partitioning of $\mathcal{Q}_{free}$ into zones, called Voronoi regions. Each region has a specific point that is called site or seed. If robots' position $P = \{p_1, p_2, ..., p_n\}$ denotes the set of seeds, the formal definition of Voronoi region will be given by:

$$v_i = \{q \in \mathcal{Q}_{free} | d(q, p_i) \leq d(q, p_j), \forall i \neq j\}, \tag{1}$$

where $d(q, p_i)$ denotes the distance between a point $q \in \mathcal{Q}_{free}$ and p_i, which is the region site. If we define obstacles as a set of single points, then this set can be considered as a seed. Consequently, the definition of Voronoi region is extended by considering the seeds to be sets instead of single points. GVD is defined as the set of points where the distance to two closest obstacles is the same [8]:

$$\mathcal{T}_{ij} = \{q \in \mathcal{S}_{ij} | d(q, \mathcal{QO}_i) \leq d(q, \mathcal{QO}_h), \forall h\}, \tag{2}$$

where $\mathcal{QO}_i$, $\mathcal{QO}_h$ are two closest obstacles to q and $\mathcal{T}_{ij}$ is termed two-equidistant faces. $\mathcal{S}_{ij}$ is the set defined by:

$$\mathcal{S}_{i,j} = \{q \in \mathcal{Q}_{free} | d(q, \mathcal{QO}_i) = d(q, \mathcal{QO}_j) \text{ and } s\nabla d(q, \mathcal{QO}_i) \neq \nabla d(q, \mathcal{QO}_j)\}, \tag{3}$$

Now the definition of the GVD can be more precise:

$$GVD = \bigcup_i \bigcup_j \mathcal{T}_{i,j}. \tag{4}$$

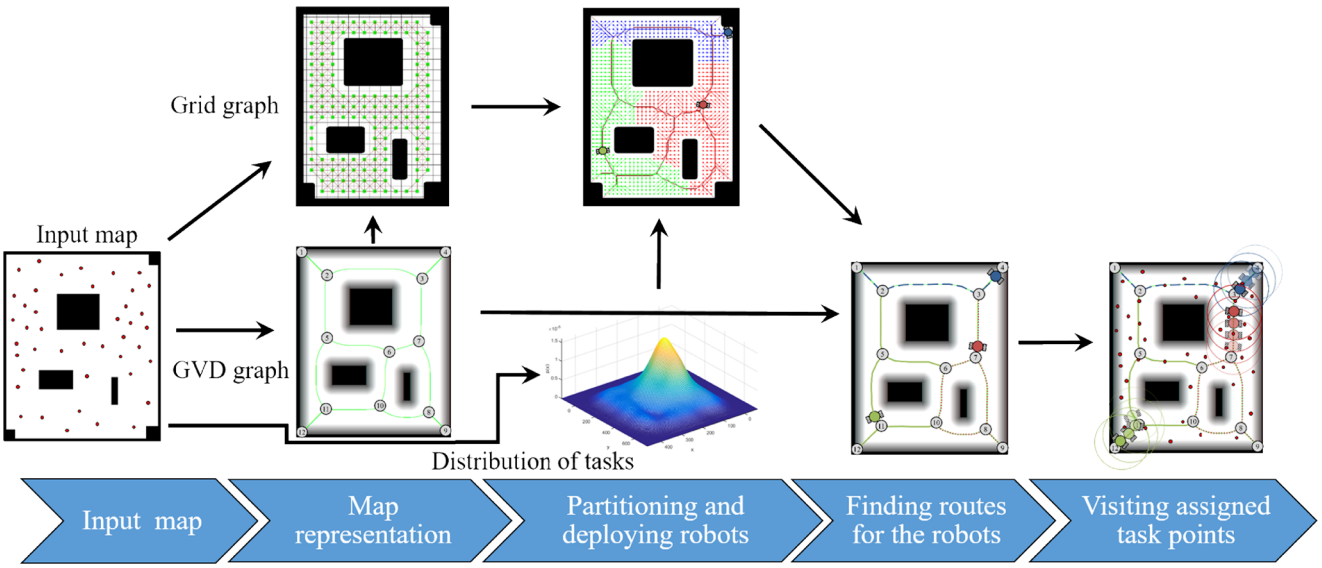


Fig. 2 After creating GVD and grid graph given input map, robots are deployed on GVD. Later robots visits task points based on the route founded on GVD

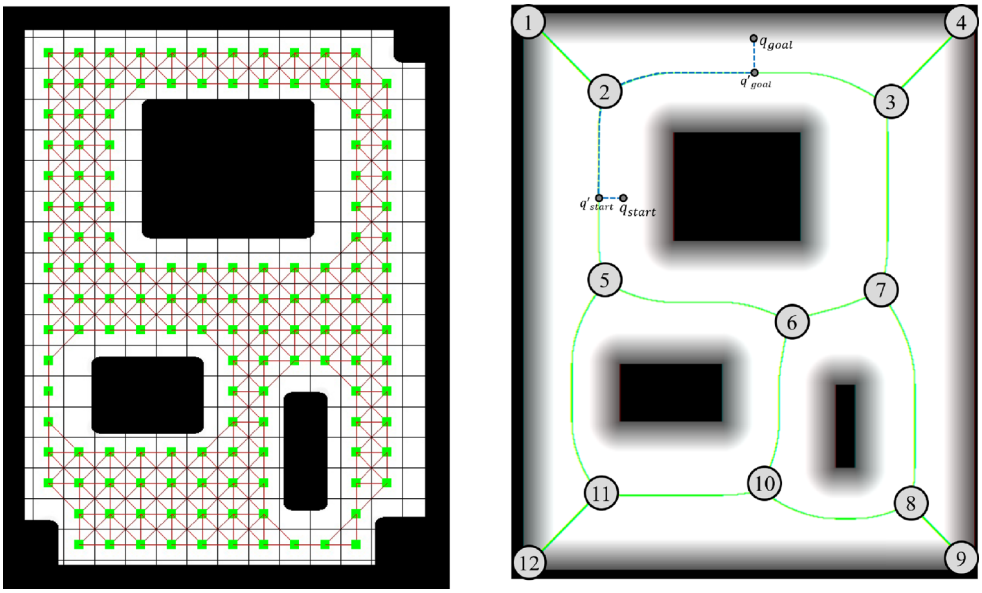


Fig. 3 A representation of a map in grid graph with 8-connectivity links between nodes

(a) Constructing grid graph by discretizing the input map.

(b) Constructing GVD graph based on GVD.

This graph is the safest path that robots can move on it (see Fig. 3b). We merged these graphs and construct a graph G based on a new metric proposed by [19], explained in the following.

Definition 1 (Geodesic GVD (GGVD)) In an environment with GVD as a roadmap, finding a path from an initial point q_{start} to a final point q_{goal}, *Geodesic Distance Based on GVD* (GGVD) is defined as:

$$dg(p_i, p_j) = W_1.||p_i - \Pi_i(GVD)|| + W_2.g(\Pi_i(GVD), \Pi_j(GVD)) + W_1.||p_j - \Pi_j(GVD)||, \quad (5)$$

with the following properties:

$$\begin{cases} dg(x_i, x_i) = 0, \\ dg(x_i, x_j) = dg(x_j, x_i), \\ dg(x_i, x_j) \geq 0, \\ dg(x_i, x_j) \leq dg(x_i, x_k) + dg(x_k, x_j) \end{cases}$$

where $g(x_i, x_j)$ gives the shortest distance between two points x_i and x_j on the GVD, $\Pi_i(GVD)$ represents the projection of the point p_i onto the GVD which corresponds to the closest point on the GVD to p_i, and W_1 and W_2 are the weights of each part of the path.

By assigning a higher value to W_1 the cost of path from q_{start} to GVD or from GVD to q_{goal} will increase. An example of a path from two arbitrary points based on the new metric is shown in Fig. 3 b).

Resulting graph G is created by modifying G_{grid}, such that:

$$G = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}, \text{ where } \mathcal{W}(\mathcal{V}_i, \mathcal{V}_j) = dg(\mathcal{V}_i, \mathcal{V}_j), \quad \forall\, \mathcal{V}_i, \mathcal{V}_j \in \mathcal{V}, \;\; i \neq j`$$

An example of graph G with the new metric is illustrated in Fig. 4, where the distance from 3 robots to all the cells is computed and illustrated by different colors.

4.2 Distribution of task points

In this paper, tasks are defined a set of points, $T = \{t_1, t_2, \cdots, t_m\}$, where $t_i \in \mathcal{Q}_{free}$ that must be visited by robots. Visiting these points could be for searching purpose. Since limited number of robots are available, a good allocation of task points to robots results a better performance. Although this allocation depends on location of robots, the density of task points should be considered as well. In order to find a function $\phi(t_i) \rightarrow \mathbb{R}^+$ showing density of task points we applied *Kernel Density Estimation* (KDE) that estimate a bidimensional function. Usually KDE is used for estimating probability density function (PDF) of the random variables. For a set of observation $x_1, x_2, \cdots, x_n$ it has following formulation:

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{N} K(\frac{y - x_i}{h}), \quad (6)$$

where, $h > 0$ is the so-called bandwidth: the bandwidth here acts as a smoothing parameter, controlling the tradeoff between bias and variance in the result. A larger bandwidth

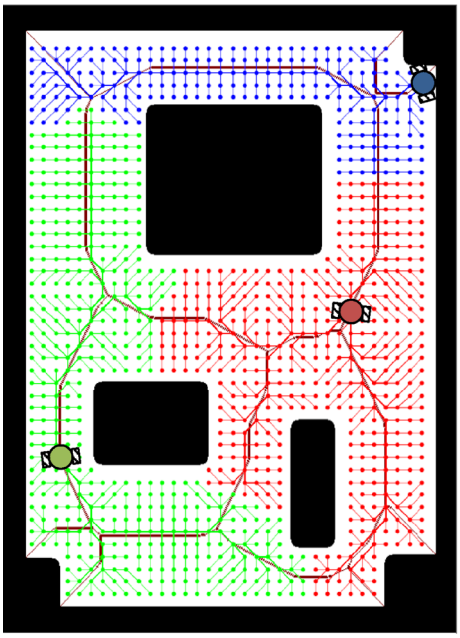


Fig. 4 Constructing graph G based on G_{Grid}, G_{GVD} and the new metric. Obviously robots are forced to move through GVD when they want to reach to a point in the map. Small displacement caused by discretization of the map into grid cells

leads to a smoother density distribution. K is the kernel function which is a symmetric multivariate density. $K()$ can be different functions i.e. exponential, linear, cosine etc. However, in this paper we set the kernel as the "Gaussian" function.

Figure 5a) shows a random distribution of task points. Accordingly, it can be seen in Fig 5b) and c), the corresponding contour and 3D PDF computed based on KDE method.

4.3 Deployment and partitioning

Suppose we have a group of n robots that are available in a bounded environment, $\Omega \subset \mathbb{R}^2$. The deploying position of robots given by $P = \{p_1, \ldots, p_n\}$, where $p_i \in \mathbb{R}^2$. Also, Voronoi partitioning applied on environment given (1), where v_i is Voronoi region for robot i, and $\cup_{i=1}^{n} v_i = \mathcal{Q}_{free}$. In other words, region partitioning is computed on graph G and constructs several sub-graph g_i, where $\cup_{i=1}^{n} g_i = G$. As a result g_i (or sub-region i) contains GVD sub-routes and task points $tp_i = \{t_p, \cdots, t_q, \cdots, t_r\}$, $tp_i \subset v_i$ and $tp_i \subset T$.

In a continuous space, robots drive to centroid of Voronoi regions in Centoidal Voronoi Tessellation (CVT) [10]. This approach is based on classic discrete-time Lloyd's algorithm [43]. In Fig. 6, by using a CVT scheme the environment is partitioned between five robots such that robot i moves to the center of v_i and is responsible to tasks inside this region.

In general the quality of deploying can be measured by using deployment function proposed originally by [10] as follows:

$$\mathcal{H}(P, V) = \sum_{i=1}^{n} \mathcal{H}(p_i, v_i), \tag{7}$$

where, in our discrete space we have:

$$\mathcal{H}(p_i, v_i) = \sum_{t_j \in tp_i} dg(t_j, p_i)^2 \phi(t_j), \tag{8}$$

$dg(t_j, p_i)$ indicates geodesic-GVD distance between robot position and its dedicated task point ($t_j \in tp_i$) in Voronoi region i. As explained before $\phi(t_j)$ is the PDF of task points in the map. Therefore, $\mathcal{H}$ will have a smaller (better) value when robots have minimum distance to the task points.

4.4 Finding routes

In order to find routes for the robots, given subregions including GVD graph inside, we can simply apply single Chinese Postman Problem (CPP) algorithm. Since we already partition environment between robots, single CPP will be used for each robot individually, instead of mixed-CPP. In this way, the complexity will reduce in comparison to mixed-CPP. Solving single CPP, however, will cost expensive in our population based searching algorithm. Therefore, the routes for each robot will be found within our genetic algorithm based on representation of the individual and a DFS-like algorithm, which is explained in Section 4.6.1.

The result of this step will be a set of routes $R = \{r_1, \cdots, r_n\}$, where $r_i = \{\mathcal{V}_p, \cdots, \mathcal{V}_q, \cdots, \mathcal{V}_r\}$ contains nodes and edges of graph G_{GVD}, thus $\cup_{i=1}^{n} r_i = G_{GVD}$. Given the routes we defined a second objective to minimize the cost of visiting task points over the routes in (9).

$$\mathcal{C}(R, T) = \max_{i=1}^{n} \mathcal{C}(r_i, tp_i), \tag{9}$$

where,

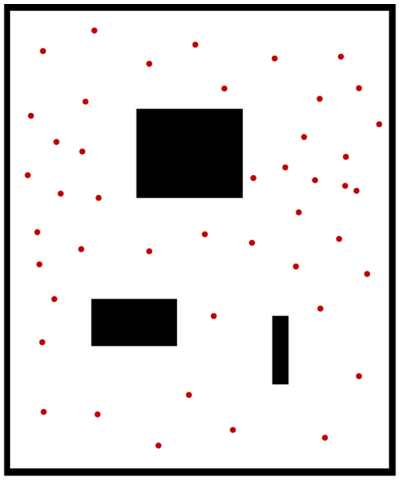
$$\mathcal{C}(r_i, tp_i) = Cost(r_i) + Cost(tp_i), \tag{10}$$

thus, the total cost for finishing the mission for robot i is the summation of time needed to traverse its route plus visit all the task points (performing the desired task i.e. searching on task points) in its dedicated Voronoi region, respectively. Given the length of a route and speed of the robot the time needed to traverse the route can be computed.

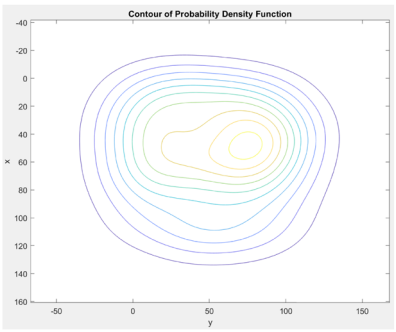
4.5 Visiting task points

Up to this step, we already partition the map and now robots have routes to explore the map and visit task points and perform a task on every point. To do so, robots move on part of GVD graph (G_{GVD}) corresponding to

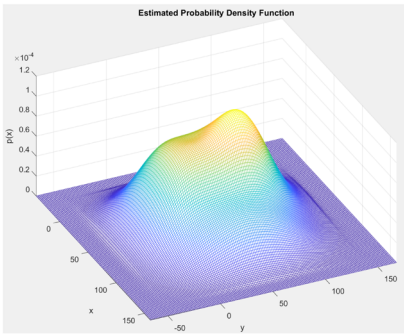
Fig. 5 An example of finding probability density function over a random distribution of the task points by using KDE method



(a) A random distribution of the task points.



(b) Contour of the probabity density function.



(c) 3D view of probability density function by using KDE.

the sub-region in G. Accordingly robot may leave the GVD to visit a task point (this case is shown in Fig. 7) and come back to the GVD to continue exploration. In a simple scenario we can assume that a task will be visited when it is located inside robot sensor range. The focus of this work, however, is not how to execute tasks, instead we are dealing with how to allocate tasks between robots.

4.6 Methodology I

According to the proposed approach, there might be many different partitioning and routes for a team of robot to visit task points. In order to find a near optimal solution we applied a meta heuristic algorithm similar to genetic algorithm (NSGA-II), with different techniques for exploration and exploitation.

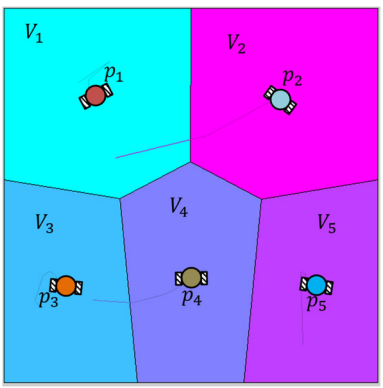


Fig. 6 A CVT based deployment for a group of robots, where each robot is located on centroid of its Voronoi region

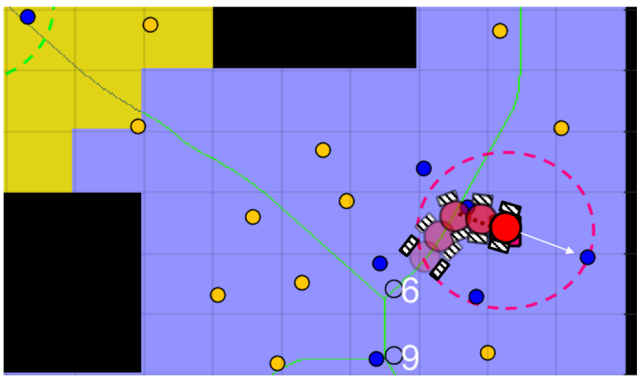


Fig. 7 Robot leaves the GVD in order to visit a task point which is out of its sensor range

Considering the objective functions defined in (7) and (9), Algorithm 1 is used to find a near optimal solution by searching the solution space. In order to create initial population (lines 5-7), a random set of nodes from G_{GVD} is selected as starting points of robots. In line 6 by calling $Partition()$ function, Voronoi region V_{pop}, routes R_{pop} and task points T_{pop} of each robots is computed in Algorithm 2. As we explained the function $FindRoute()$ is implemented by a DFS-like algorithm, however we use CPP algorithm in our simulation as well. In general to explore all the edges of a graph Chinese Postman Problem (CPP) algorithm is used. In lines 8 to 10, three different operators are applied on selected parents: a local search algorithm, mutation and cross-over operators are explained in Section 4.6.2. The evolution happens in lines 8 to 15.

Algorithm 1 $Multi\_Robot\_Exploration()$.

Input: map, T // Input map, Task points
1 $G_{GVD} \leftarrow CreateGVD(map)$ Create GVD graph given the map.
2 $G_{Grid} \leftarrow CreateGrid(map)$ Create grid graph given the map.
3 $\phi \leftarrow FindPDF(map, T)$ Calculate probability density function given T.
4 $G \leftarrow MergeGraph(G_{GVD}, G_{Grid}, \phi)$ Create graph with the new metric G.
Genetic Algorithm:
5 $Pop \leftarrow CreatePop(G_{GVD}, G)$ Create initial population.
6 $(V_{pop}, R_{pop}, T_{pop}) \leftarrow Partition(Pop, G)$ Partition the map.
7 $Pop_{fit} \leftarrow CalculateFitness(Pop, V_{pop}, R_{pop}, T_{pop}, \phi)$ Compute the fitness of each solution (chromosome).
while *Termination_Criteria* **do**
8 $Pop \leftarrow Sort(Pop, Pop_{fit})$ Non-domination sorting.
9 $Pop_{surv} \leftarrow Selection(Pop)$ Select parents for genetic operators.
Searching operators:
10 $Offs1 \leftarrow LocalSearch(Pop_{surv})$ A local search technique.
11 $Offs2 \leftarrow Cross(Pop_{surv} \cup Offs1)$ Changing permutation of robots' path.
12 $Offs3 \leftarrow Mut(Pop_{surv} \cup Offs1 \cup Offs2)$ Changing robots' location.
13 $Pop \leftarrow Offs1 \cup Offs2 \cup offs3 \cup Pop_{surv}$ Concatenating three offsprings.
14 $(V_{pop}, R_{pop}, T_{pop}) \leftarrow Partition(Pop, G)$ Partition the map.
15 $Pop_{fit} \leftarrow CalculateFitness(Pop, V_{pop}, R_{pop}, T_{pop}, \phi)$ Compute the fitness of each solution (chromosome).

Algorithm 2 $Partition()$.

Input: Pop, G // Inputs are Pop and G
Output: V_{pop}, T_{pop}, R_{pop} // Population and Voronoi region
1 $V_{pop} \leftarrow VoronoiD(Pop, G)$ Compute Voronoi Diagram.
2 $T_{pop} \leftarrow FindTask(V_{pop})$ Find tasks in Voronoi regions.
3 $R_{pop} \leftarrow FindRoute(V_{pop})$ Find routes in Voronoi regions.

4.6.1 Representation of the problem

We defined a specific structure of nodes to represent the problem. In Fig. 9 we show the representation in the GA corresponding to Fig. 8. For example, robot 3 is located on node 18 and has 7 nodes in its Voronoi region (1, 2, 13, 14, 17, 18, 19).

For the routing purpose, we use the nodes in each chromosome and try to find a shorter route which pass all the edges connecting nodes in each row (See Fig. 9). As far as the objective is to explore edges between nodes, we guarantee that all the edges will be explored by considering all the links in adjacency subgraph matrix for every robot. Although we apply a DFS-like algorithm to find the routes, during evolution we expect that the nodes and routes will evolve given their fitness values (objective functions). In our DFS-like algorithm, to find a route for robot i, if there is a link which is not a part of its Voronoi but makes the entire route shorter, it may be in the route of robot i. The other challenge is assigning common edges between two robots which will be explained in Section 4.6.3.

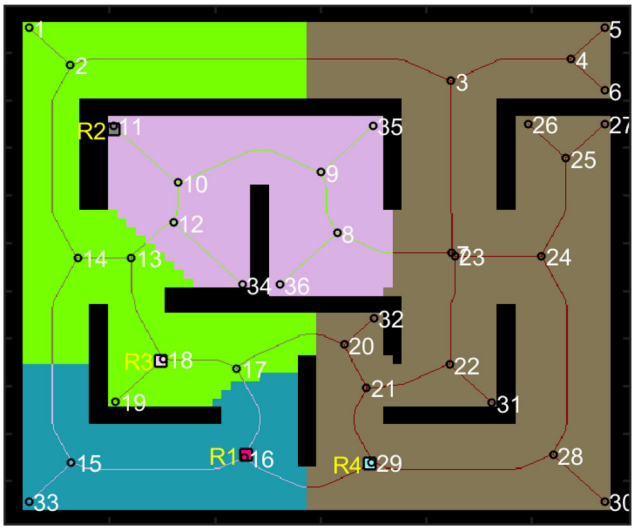


Fig. 8 An input map is represented in a graph with nodes on GVD's meet points and end points. After computing Voronoi region (by the new metric) for 4 robots, each of which has its own region (subgraph) with nodes to travel

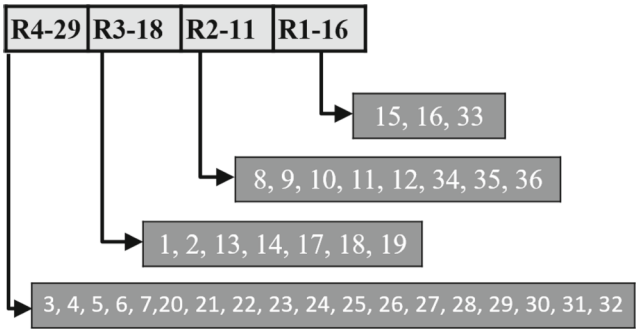


Fig. 9 A chromosome contains 4 robots' positions and nodes according to Fig. 8

4.6.2 Local search, mutation and crossover

To have a trade off between exploration and exploitation in searching phase, we defined three different operators. In one hand, for the *local exploration* purpose we change the location of some robots to nearby nodes, which causes recomputing Voronoi diagram etc. An example is shown in Fig. 10a), where a chromosome among the population is selected randomly. Later a randomly selected gen will be replaced by a node (which also is randomly selected) through its neighbor nodes in the graph i.e. in our example nodes 11 and 29 are replaced by 10 and 21 respectively (Fig. 8 illustrates the neighbor nodes). Beside this operator, we applied a *mutation operator* which selects a random gene from the chromosome and changes the node to another arbitrary node located in its Voronoi region (Fig. 10b)); in contrast to local search where the new node must be the neighbor node in the graph. This operator promotes variety in the population by applying randomness. And finally, exploitation is done by *order cross-over* operator (see Fig. 10 b)) which produces a permutation of the parents, thus nodes will not be repeated.

4.6.3 Modifying Voronoi region by considering common edges

As we mentioned to consider sub GVD graph of each Voronoi region, we have some edges which are common between two neighbor robots. So:

$$\{\forall i, j \; \exists \; \mathcal{E}(\mathcal{V}_i, \mathcal{V}_j), \; where \; \mathcal{V}_i \in v_i \; and \; \mathcal{V}_j \in v_j\}.$$

To handle these edges, our solution is to compute the load of assigned task for the neighbor robots and assign this specific edge to the robot with lighter mission. We applied this modification by computing the following term for neighbor robots:

$$L_i = \frac{\sum_{t_j \in tp_i} dg(p_i, t_j)}{\sum_{t_j \in tp_i} \phi(t_j)}, \tag{11}$$

where, $dg(p_i, t_j)$ is the distance between robot i and its task points, and $\phi(t_j)$ shows the PDF of task points in v_i. In

Parent	Node	16	11	18	29
	Neighbor node	15 17 29	10	13 17 19	16 21 28
New individual		16	10	18	21

(a) A local search operator for changing genes of an individual by using neighbor nodes.

Parent	Node	16	11	18	29
New individual		16	8	14	29

(b) An operator for random exploration.

Parent	98	109	15	110	76	12	33	66
	95	115	79	4	102	112	15	91
New individual	115	79	15	110	76	112	91	95
	98	109	79	4	102	12	33	66

(c) An operator that exchanges parent information for exploitation purpose.

Fig. 10 Two operators are applied to do exploration and exploitation in problem space

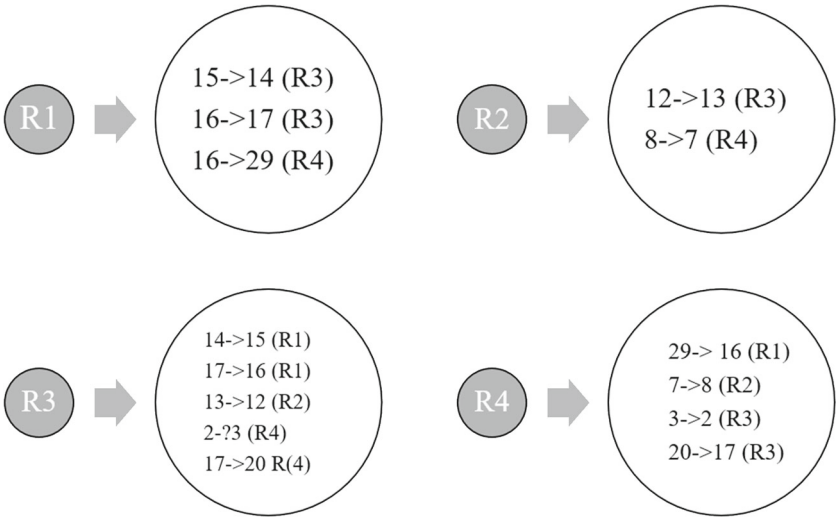


Fig. 11 Common edges between 4 robots in example given in Fig. 8

this way, L_i computes the ratio of the area and PDF of task assigned to robot i. For a big area with few task points, L_i is bigger value than with more tasks and vice versa. Thus, we are interested to give the common edge (task) to the robot which has lower L_i. Accordingly, the Voronoi region of two neighbor robots will be modified, thus one will receive more task to do while the other one less.

To know which edges are common between two robots (or two sub-graph g_i and g_j) we find the links (edges) that connect two subgraphs. Given example in Fig. 8, can be seen common edges in Fig. 11, where robot 1 has 3 edges ($15 \rightarrow 14$, $16 \rightarrow 17$ *and* $16 \rightarrow 29$) in common with robots 3 and 4.

After finding common edges, they must be added in the corresponding chromosome. The modified chromosome is shown in Fig. 12. Common edges are added to the robots with less load (according to (11)). In this specific example robot 4 does not receive new nodes.

4.7 Methodology II

In our second methodology we propose a Q-learning algorithm to solve the problem. As one of the traditional RL algorithm, Q-learning computes estimation of the state-action pair value function $Q(s, a)$. In the learning process, sequence of state-action value tried to find optimal value function $Q^*(s, a)$. In iteration t, agent observe state s, select an action a, and receive immediate reward r. The most common action-value function in reinforcement learning task is Bellman function, which is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \, max \, Q(s',a)] \tag{12}$$

where, α is learning rate, γ is discount factor for balancing immediate and future reward. In Q-learning agents try different actions given the different states, to learn the policy which maximize the Q-value. In action selection agents need to keep a trade-off between exploration to explore the environment and exploitation to learn the most rewarding actions. To do so, we applied ϵ-greedy policy, where a parameter $\epsilon \in [0, 1]$ controls exploration and exploitation rate, after generating a random number $b \sim \mathcal{N}(0, 1)$, if b is larger than ϵ, the greedy action is selected, otherwise a random action among the possible action list is selected:

$$\pi(a|s) = \begin{cases} \arg\max \; Q(s,a) & if \;\; b \geq \epsilon \\ \text{a} \in [\mathcal{L}_a \setminus \arg\max \; Q(s,a)] & \text{otherwise} \end{cases}, \tag{13}$$

Given the graph G, action set $a = a_1, a_2, \cdots, a_n$, where $a_i = n_1, n_2, \cdots, n_k$, is the neighbor nodes of a_i, indicating the possible action that the agent can apply in its current position (node a_i). In our scenario to list potential solution for robots' next position, all combinations of the next nodes connected to the robots' current position are considered. As in Fig. 13 suppose that we have 3 robots, reaching to nodes 3, 12, 20, which has 2 options nodes each. In this way, potential action list includes all combination of neighbor nodes i.e. $\mathcal{L}_a = \{(2, 13, 18), (2, 13, 22), (26, 13, 18), (26, 13, 22), \cdots\}$, where $|\mathcal{L}| = 8$ in this example.

In a state, to find the best action which yields maximum value, in a multi-objective Q-learning scheme, we applied

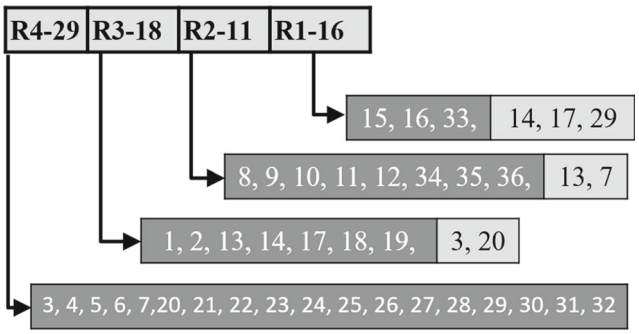


Fig. 12 Modified chromosome by considering common edges in Fig. 8. New nodes are added at the end

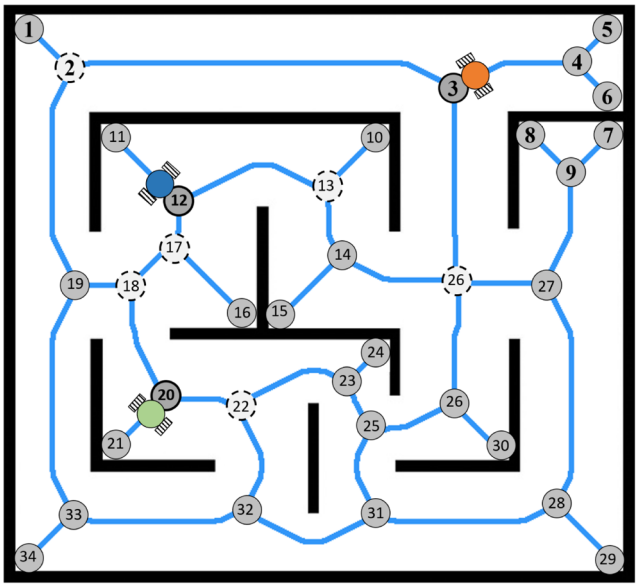


Fig. 13 Robots are located on nodes 3, 12 and 20, where they have two potential nodes for their next position

a non-dominated sorting approach. Therefore, in our graph representation to select next best node, considering path between current and next node plus the time needed to service task point in the route the same objective functions in (8) and (9) are computed. In general, robots start from a random position, and move to the position where minimize the objective functions. However, there might be a case that two robots want to move to the same node at the same time, where the robot with less cost can be the winner. Algorithm 3 shows the details in the proposed approach.

Algorithm 3 Q-learning algorithm.

Input: $Q(s, a)$
Output: R_{pop} Final rout for each robot
$Q(s, a) \leftarrow 0$ Initialized Q
for *episod* $i \leftarrow 0$ **do**
  S select a random initial state
  while s *is not terminal* **do**
    Create a list of potential actions $\mathcal{L}_a$ given state S
    Calculate objectives (F_1, F_2 (8),(9)) for all the potential actions.
    Select action $a \in \mathcal{L}_a$ given policy derived in (13)
    $v_i^* \leftarrow$ corresponding node to the action a
    apply action a, receive reward, and move to next state s'
    update the rule
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma\ max Qs(s', a)]$
    $s \leftarrow s'$ update current state to s'

In this algorithm Q-table with size $N \times N$ (where N is number of nodes in the graph) is initialized. In the inner loop after creating a list of possible action for the next state, we compute our objective functions defined in (8), (9), and select one of them given the ϵ-greedy policy (13). In the greedy scheme we apply non-dominated sorting approach to consider both objectives and select one with smaller F_2 value among non-dominated solutions. Later the node is selected and the reward must be given to the action. To do so, we applied the objective function values to reward the action, and finally change the state to the current one.

4.8 Robot control

Given a nonholonomic robot in Fig. 14, to move on GVD, we consider waypoints (wp_i) among that. Therefor, from current position, robots move by following a vector aligned to next waypoint **d**. In this work, although we attempted to maximize safety, however, a moving unanticipated obstacle i.e. other robots might appear and hinder robots to move along the routes. There are many researches in obstacle avoidance [36, 57, 66, 74], since the focus of this paper is not in obstacle avoidance, authors are referring the reader to related cited papers. Nevertheless, we address obstacle avoidance problem by simply re-planning. Thus, after detecting an unexpected obstacle with an equipped sensor i.e. laser range finder, robot moves along the new local path to detour this unexpected obstacle. To do so, in a vector filed scheme, we consider attractive and repulsive force from goal (next waypoint to go) and obstacle, respectively as following:

$$\mathbf{U} = \mathbf{U_{att}} + \mathbf{U_{rep}}, \tag{14}$$

where,

$$\mathbf{U} = \frac{1}{2}\eta d(p_i, wp_k)^2 + \frac{1}{2}\eta\left(\frac{1}{d(p_i, uo_k)} - \frac{1}{q^*}\right) \tag{15}$$

In (15), η is a scaling parameter for distance $d(p_i, wp_k)$ of robot to goal (waypoint), $d(p_i, uo_k)$ indicates the

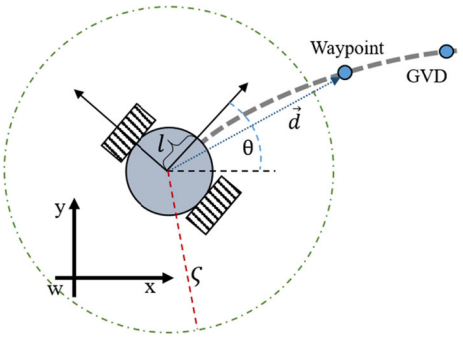


Fig. 14 A model of nonholonomic robot in working space with state vector $[x, y, \theta]^T$. Robot moves toward waypoints located on GVD given the vector **d**

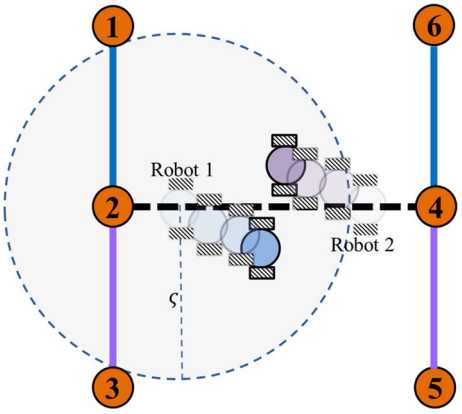


Fig. 15 Addressing unforeseen obstacle by applying potential vector filed. Two robots pass each other by following the gradient vector F

distance of robot's current position to the unseen obstacle, and $q^* < \varsigma$ is the effective distance from obstacle, thus if the robot is far enough($d(p_i, uo_k) > q^*$) from obstacle $\mathbf{U_{rep}} = 0$.

Given above explanation, to move robots toward the next waypoint, the gradient of vector U is needed, hence we define function $F = -\nabla U$. By feeding the gradient vector ($[F_x \;\; F_y]^T$) into feedback linerarization controller [49], we achieve:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta)/l & cos(\theta)/l \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}, \tag{16}$$

where, robot inputs v and ω are linear and angular velocity. l is the distance from the center of the robot to its control point and θ is the angle between the robot and the reference frame W (See Fig. 14). An example is shown in Fig. 15, where robot confronts with a moving obstacle. With the controlling rule in (15) each robot can pass each other. Another uncertainty could be *robots failure*, since the system is centralized after determining a failure, task assignment algorithm will be re-executed for the remaining tasks, hence available robots will be part of the new mission.

5 Simulation results

We tried to implement several methods to compare with our proposal, however some of them didn't fit on our problem, i.e one couldn't work in our representation, the other didn't consider our continuous density function, etc. However, we selected most similar approaches which we could apply in our problem with minor modification. Thus, in some of them we consider our formulation and parameters to have a fair comparison. The most similar solutions to the problem (without considering distribution of task points) addressed in this paper is the family of Chinese Postman Problem (CPP) and Vehicle Routing Problem (VRP), which are NP-hard problems. In other word, it is very difficult to find optimum solution by applying centralized solvers. For instance, in one of our try, for a graph with 22 nodes and 3 robots after 10 hours running no result was found for m-CPP problem. In this way, we could not compare the proposed method with centralized solvers. However, we compare our algorithm with a similar approaches proposed in [1, 18, 24, 59].

The main concern in exploration problem is the length of the final route that robots must explore. In the case of multi-robot system, depends on the number of task points in each region and the time taken to perform tasks, the cost of the robot that finish its mission lastly, is considered as total cost.

In our simulation, for instance, the average speed of robots is $s = 0.05$ m/sec, and 1 second is considered to visit a single task point when it is inside robots laser range (ς). Otherwise, robot moves toward the task point until it lies inside robot's sensor range. Therefore, to compute the cost of routes we rewrite (10) as following:

$$\mathcal{C}(r_i, tp_i) = Len(r_i) \cdot s + \sum_{j \in tp_i} CE_j^{tp}, \tag{17}$$

where, $Len(r_i)$ is the length of the route i, and the cost of visiting task points CE_j^{tp} is defined in two cases:

$$CE_j^{tp} = \begin{cases} 1\,\text{sec} & if \;\; t_j \in \varsigma \\ 1\,\text{sec} + \text{Len}(r_i^j) \cdot \text{s} & \text{Otherwise} \end{cases},$$

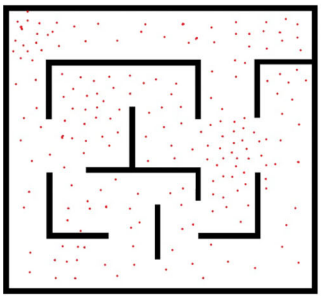
in the second case, if robot i needs to move toward task point t_j, sub-route r_i^j can be defined as a line in the following:

$$r_i^j = \lambda \cdot p_i + (1 - \lambda)(|t_j - \varsigma|), \forall \lambda \in [0, 1].$$

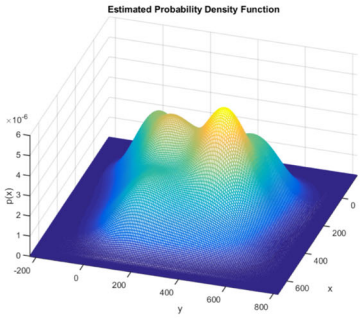
It should be noticed that the map is an images, so the length can be considered the number of pixels. Simulations are done in 5 different scenarios which are mentioned in Table 1, including number of task points, node and time needed to search each task point. While 3 first maps can be considered as a real map with present of obstacle (See Fig. 16), 2 last maps are graphs without obstacle. To show the consistency and convergence of the proposed algorithm Monte-Carlo

Table 1 The time (in second) needed to execute algorithm to find a solution

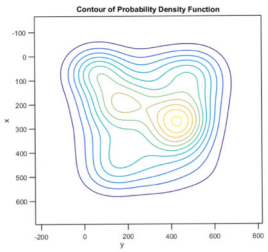
Map num.	Name	task point num.	Node num.	task duration	Obstacle
1	office1	352	34	1	No
2	midmap [1]	110	22	1	Yes
3	office2	294	120	1	Yes
4	graph1	2000	500	1	No
5	graph2	4000	1000	1	No



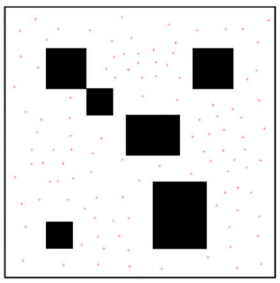
(a) Office 1: The first map with the task points.



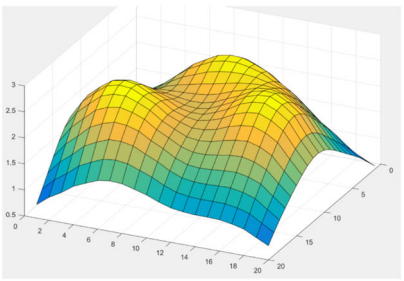
(b) 3D view of probability density function by using KDE.



(c) Contour of the probabity density function.

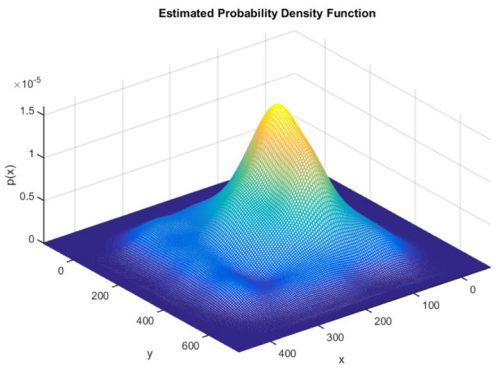


(d) Midmap: The second map with the task points

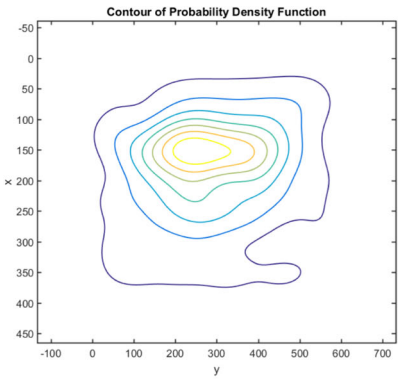


(e) The function of distribution of task points.

(f) Office 2: The third map with the task points distributed in the map.



(g) 3D view of probability density function by using KDE.



(h) Contour of the probabity density function.

Fig. 16 3 main maps where applied in our simulation with their distribution of tasks. Probability density function is computed by KDE method according to the distribution of the task

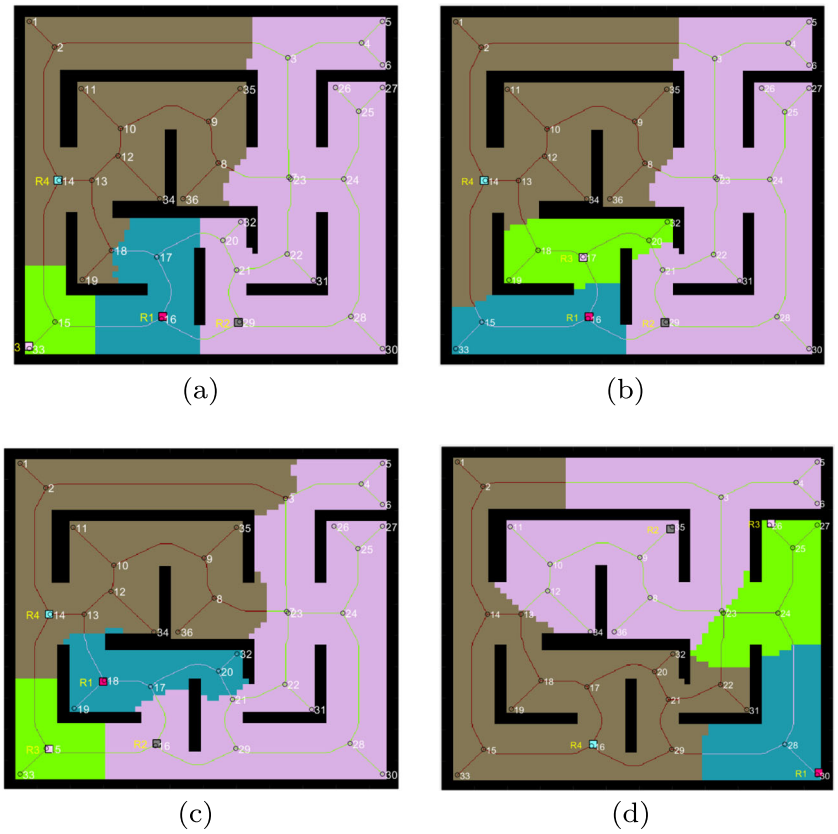


Fig. 17 The estimated Pareto set with 4 robots. Different partitions are shown with different colors. Squares indicate the robots' positions

simulations are performed in our experiments. Experiments are implemented in Matlab on a computer with processor Intel (R) Core (TM) i7-3520M 2.90 GHz with 12 GB RAM.

5.1 Partitioning and routing

In the first simulation the proposed approach is executed on the map *office1* with 34 nodes. Our objective is to evaluate our *routing* and *partitioning* approach in our specific problem in comparison to well-known CPP and K-means algorithms, respectively. Given a set of solutions in the estimated Pareto front we select the one which has a better routing (smaller $\mathcal{C}(R, T)$). For instance, some of the solutions in the estimated Pareto front are shown in Fig. 17 a-d). Each partition assigned to a robot, where is located inside that partition (Square).

After 30 Monte-Carlo simulations with varying robot's initial location we achieved following results in Table 2. In this table the cost given (9) and (17) is computed and its *min, max, mean* and *standard deviation* of runs is shown. In each run the maximum cost is considered.

Since the location of robots changes in different runs, the standard deviation of maximum cost varies as well. Both approaches find the near optimum solution, however, we preferred to apply our DFS-Like instead of single CPP; while the former is embedded in the searching algorithm, the latter run separately. In searching algorithm we penalize the longer routes, simultaneously, with finding the route, therefor in the sense of computational time this method works in an efficient way in comparison to CPP.

We also show the algorithm execution time to find a solution in Table 3. As it can be seen among two applied methods for routing: our DFS-like and CPP, the proposed method runs faster. However both can find a solution in

Table 2 The result of applying proposed algorithm with two routing methods

Method/$\mathcal{C}$	Min	Max	STD	Mean
Single_CPP	98.30	172.91	24.06	124.90
Proposed Method I	73.64	165.39	21.09	120.25

$\mathcal{C}$ is defined in (17)

Table 3 The time (in second) needed to execute algorithm to find a solution

Method/sec	Min	Max	STD	Mean
Single_CPP	515	1229	194.52	814.67
Proposed Method I	329	845	138.89	552.93

Table 4 The result of applying our proposed algorithm with two clustering methods

Method/$\mathcal{C}$	Min	Max	STD	Mean
K-means Clustering	101.96	227.46	39.02	163.46
Proposed Method I	72.94	166.79	29.81	117.88

$\mathcal{C}$ is defined in (17)

an acceptable time, while using m-CPP solvers will not be practical.

In another simulation, to compare the performance of our clustering algorithm, we applied k-means clustering instead of our Voronoi GVD-based method, and as it shown in Table 4, the result is better in our method. Beside the weakness of k-means in none hyper-ellipsoids space, which might be the case in our application, it brings an extra burden in our framework which does the partitioning together with searching.

5.2 Comparison with similar methods

In order to show the performance of the proposed algorithms, we compare our methods with several most similar approaches in [1, 18, 24, 59]. For example, in [1], author used a swap algorithm [6] for partitioning the area, and in the iterative approach genetic algorithm is applied for Single Traveling Salesman Problem to find the shortest tour in each sub region. We also applied the approach proposed by Chen et al. in [24], where genetic algorithm is used to solve our MRTA problem. We feed the parameter in a way that is compatible to their approach. Moreover, works proposed in [18] and [59] are considered in our comparison as well. The result of our experiment is shown in Table 5, where *minimum*, *maximum*, *standard deviation* and *mean* of 5 scenarios are listed in this table. Number of robot in each scenario varies given the size and task points number. It can be seen that our proposed algorithms could find a better solution in most of the scenarios with a smaller mean value. However, in the first 3 maps our algorithms work much better than others in comparison to the two last maps. Seems that, other approaches work well in obstacle free environment. It should be mentioned that in contrast to our proposed method, the compared approaches do not consider the distribution of the task points. Therefore, we achieved a better performance in the proposed method since we partition and deploy the robots based on distribution of

Table 5 The cost of running the exploration algorithms on 5 different scenarios for 30 runs. Minimum average cost is bolded in each scenario

Scenario		GA_Clustering [18]	GA ([24])	Swap_GA [1]	Hyb_MH [59]	Proposed Meth1	Proposed Meth2
office 1	Min	75.14	80.61	85.06	79.23	68.11	72.36
5 rob	Max	115.65	126.03	108.70	149.72	129.95	121.56
	STD	10.99	11.57	6.19	16.39	15.17	9.12
	Mean	84.72	90.01	91.83	87.39	79.88	**77.63**
midmap	Min	25.002	24.06	28.45	26.34	23.17	23.13
4 rob	Max	39.18	41.82	45.66	50.68	36.31	40.68
	STD	3.83	4.59	4.61	5.31	3.29	3.23
	Mean	30.06	28.39	32.01	31.88	26.05	**25.79**
office 2	Min	58.05	61.01	62.45	65.49	55.03	57.11
7 rob	Max	90.56	89.70	84.89	78.77	75.59	81.90
	STD	7.62	8.28	4.2	3.67	3.63	4.43
	Mean	68.80	67.29	69.06	69.50	**57.78**	58.84
graph 1	Min	439.54	445.23	464.30	449.72	462.19	436.50
10 rob	Max	945.63	939.78	970.18	944.34	907.20	929.53
	STD	139.61	129.34	125.45	128.90	101.90	107.09
	Mean	**552.61**	576.76	594.12	624.79	563.38	556.28
graph 2	Min	739.20	854.36	760.98	762.94	763.41	722.55
20 rob	Max	1416.71	1497.27	1490.50	1224.63	1357.77	1423.37
	STD	174.07	180.16	183.53	117.72	111.52	156.37
	Mean	877.008	975.84	878.74	859.49	**835.2**	853.17

$\mathcal{C}$ is defined in (17)

Table 6 The duration of running the exploration algorithms on 5 different scenarios for 30 runs (in second). Shortest duration among the scenarios are bolded in each row

Scenario	GA_Clustering [18]	GA [24]	Swap_GA [1]	Hyb_MH [59]	Proposed Meth1	Proposed Meth2
office 1	65.03	69.02	72.04	67.99	63.80	**60.74**
midmap	21.84	20.93	23.90	22.42	19.98	**18.43**
office 2	49.55	47.52	50.85,	53.72	**43.33**	45.99
graph 1	**140.05**	157.16	170.19	173.15	153.85	144.53
graph 2	277.83	297.87	282.26	283.99	**268.10**	286.13

the tasks. Thus, locations with bigger peak of task have more robot to be deployed, which yields a better load balancing in task allocation.

Table 6 shows the time traveled by the robots in the 5 scenarios. More precisely, the duration to visit the last task point with one of the robots in the map. It can be seen that, in general, the proposed techniques overcome the state of the art. In smaller map, the range of the robots' sensor is big enough to visit the task points while passing over the GVD road map, in contrast the bigger map is, the more task points to visit by leaving the road map occurs, meaning that task completion time increases. Although in our experiments we consider more task point with more graph nodes in in different maps to testify the performance of the proposed approaches considering scaling factor.

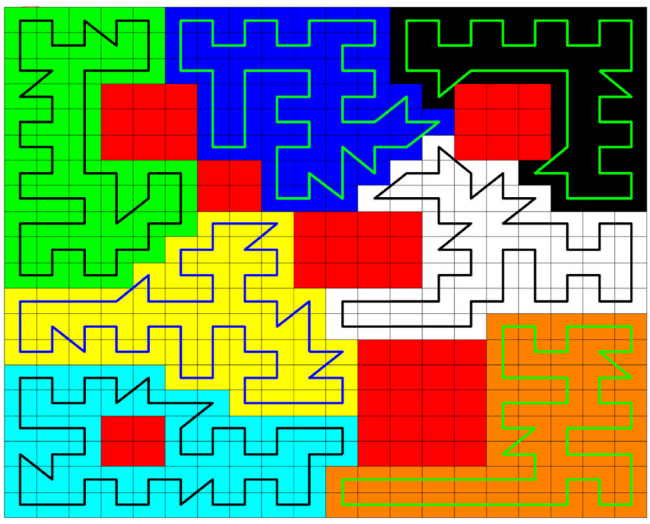
For better understanding the final route for exploring map "midmap" is shown in Fig. 18, where two different approaches created two different rout for the robots. To have a fair comparison, in computing $\mathcal{C}$ for the proposed method in [1], we find the shortest path from robot's position to the task point given the route yield by their approach.

5.3 Impact of number of task points

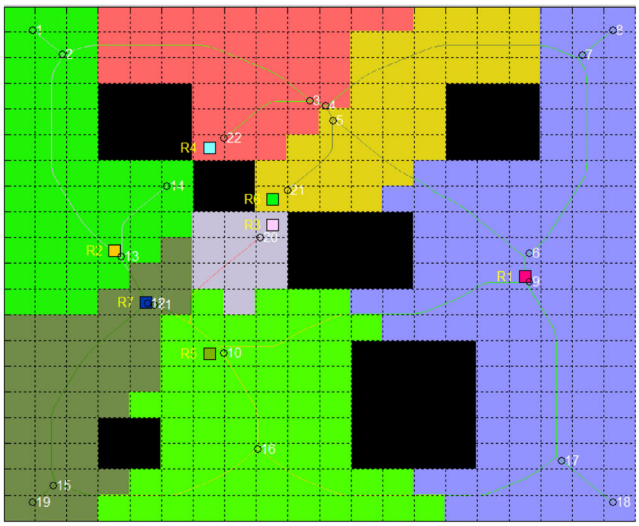
In another comparison, we defined a range of the task points for 7 robots in *offcie2* map and compute the exploration cost for different approaches. It can be seen in Fig. 19 that, our methods outperforms the other approaches. This is due to the GVD graph we used, it means robots will be always on the roadmap which has minimum distance to task points, therefore by increasing the number of task points route cost ($\mathcal{C}$) increases smoothly.

5.4 Impact of number of robots

In this simulation, we change the number of robots in *office2* map. The result of *deployment* (8) and *route cost* (17) functions for 2-10 robots is shown in Fig. 20. Obviously, by increasing the number of robots functions are decreasing. In this algorithm, if we only consider maximum rout cost the performance of individual robots becomes more important than the whole system. Therefore, our first objective *deployment function* play a complementary role to take the



(a) The result of applying [1] method for task allocation.



(b) The result of partioning by the proposed algorithm.

Fig. 18 The map of second scenario with the distribution of the tasks and results of applying two methods

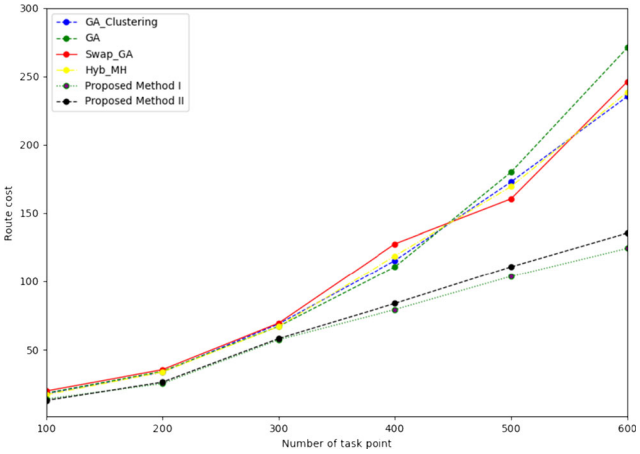


Fig. 19 Comparison between 6 strategies on office2 map by considering different number of task points

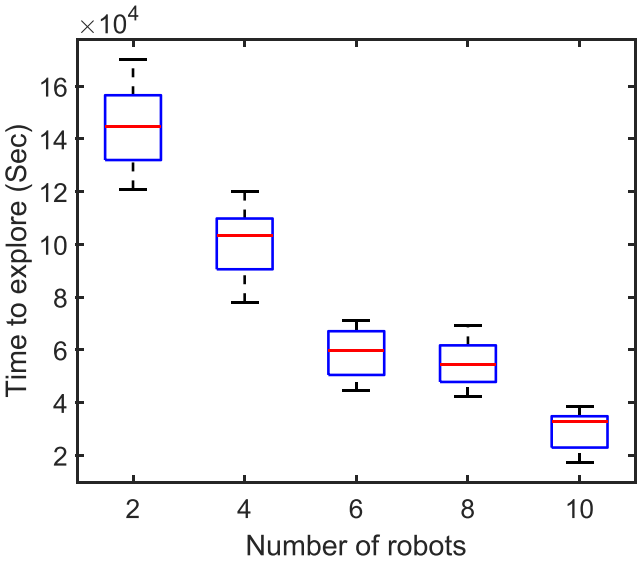


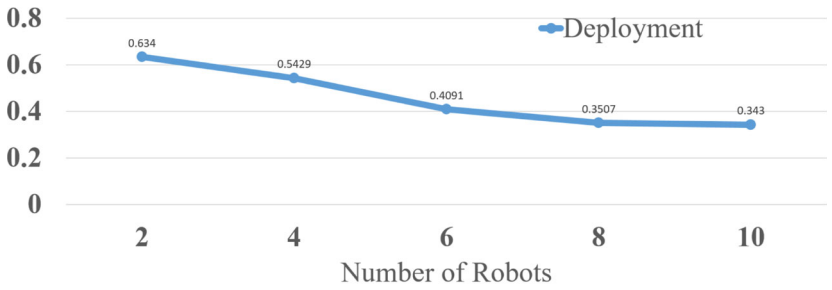
Fig. 21 Time needed to explore longest route with different number of robots in office2 map

performance of the whole system into consideration, by optimizing the deploying location of the robots given the distribution of the task points. From this figure, we can depict that by increasing the number of robots cost functions will not decrease drastically in some points, which means no need to increase the number of robots after a given number.

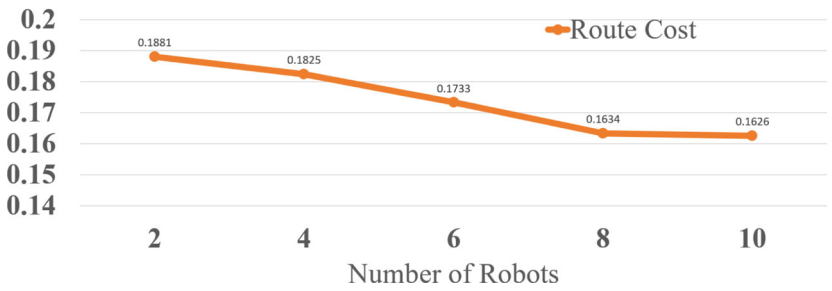
For better understanding, in Fig. 21, the exploration time for 30 Monte-Carlo runs for different number of robots is shown. It can be seen that the exploring time decreases drastically when the number of robots varies from 2 to 6. However, we do not see a considerable decrease for number of robots between 6 to 10. This result shows that, in general, multiple robots solves the problem faster.

We visualized one of the solutions in Fig. 22, where 8 robots are distributed over the map and each one has its own region and edges to visit the tasks within their allocated regions.

Fig. 20 The behavior of deployment and route cost function with increasing number of robots. Both function values are normalized



(a) The *deployment function* $\mathcal{H}$ is decreasing by increasing the number of robots.



(b) The *route cost* $\mathcal{C}$ is also decreasing by number of robots.

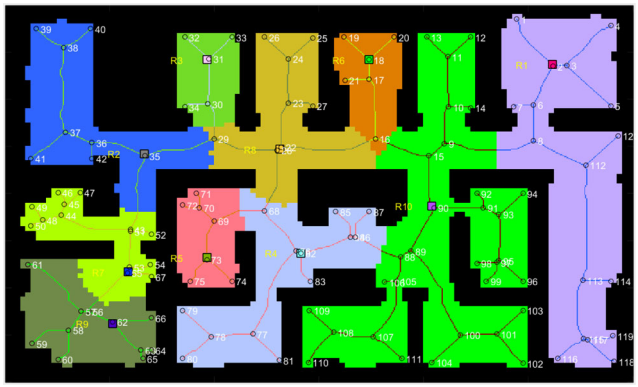


Fig. 22 A visualization of the best solution in office-like map for 8 robots

5.5 Impact of task distribution

To testify the *convergence* of the algorithm given different distribution, with the same number of task points, 294, we created 5 different distributions on *office2* map, and run the algorithm 30 times with fixed number of robots 10. Cost values in Table 7 shows that with different distribution our proposed approach solve the problem with a stable behavior. Standard deviation in all 5 scenarios indicates algorithm convergence.

5.6 Videos of multi-robot exploration

To realize how robots will execute the exploration task given their assigned routes we upload two videos on youtube. In this simulation we simplified the robot as a a holonomic one, so it can move to any direction instantaneously. The first video can be found in https://youtu.be/toKfazC-PR0 which is for the first scenario (Office1 map) with four robots. In the second video https://youtu.be/eteE33wuYOw, we show the performance of our proposed method on the midmap given from [1].

Table 7 The cost for 5 distributions and 10 robots after 30 runs

$\mathcal{C}$	Dist1	Dist2	Dist3	Dist4	Dist5
Min	37.62	35.003	30.02	28.10	30.36
Max	60.71	65.11	48.97	73.78	60.75
STD	5.35	5.41	3.56	10.72	5.46
Mean	43.24	39.11	34.34	36.66	34.25

$\mathcal{C}$ is defined in (17)

6 Conclusion

In this work, we addressed multi-robot task allocation problem. Robots must explore the environment in order to execute tasks on their assigned region. Thus, by interpreting the original problem as a task allocation problem, we proposed a new end-to-end multi-objective framework based on multi-robot deployment scheme. The map will be divided into regions based on the distribution of the task points, and each robot will be allocated to one of the regions given its location. For the simplification we represent the input map into a GVD graph (based on a traditional roadmap GVD), and also a grid graph. Later, two graphs merged into a single one given a new metric, which forces the robots to move toward GVD (as the safest path) from the shortest path. In this problem we defined two objectives for deployment/partitioning and routing, respectively. By applying a meta-heuristic and Q-learning approach we tried to minimize the objective functions. Simulation results prove the applicability of the proposed framework to solve our NP-hard problem in comparison to exact methods. Moreover comparing our method to the similar approaches shows the performance of our method in *compatibility* and *consistency*. Some shortage must be done in future work, i.e. the proposed approach does not work efficiently in a filed where there is no obstacle i.e. a farm, so using other type of meshing techniques for graph creation is necessary instead of using GVD. Also it needs an infrastructure which provides minimum requirement for communication between robots and monitoring task points. We should note that our main goal was to develop an end-to-end framework for addressing task allocation, exploration and routing simultaneously, however checking performance of other substitution remains open for future work as well.

References

1. Ann S, Kim Y, Ahn J (2015) Area allocation algorithm for multiple uavs area coverage based on clustering and graph method. IFAC PapersOnLine 48(9):204–209
2. Bao B, Yang Y, Chen Q, Liu A, Zhao J (2014) Task allocation optimization in collaborative customized product development based on double-population adaptive genetic algorithm. Journal of Intelligent Manufacturing
3. Pham H, Le QV, Norouzi M, Bengio S (2017) Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940
4. Bhattacharya S, Ghrist R, Kumar V (2013) Multi-robot coverage and exploration in non-euclidean metric spaces. Algo Foundat Robot X Springer Tracts in Adv Robot 86:245–262
5. Booth KEC, Nejat G, Beck JC (2016) A constraint programming approach to multi-robot task allocation and scheduling in retirement homes. In: Rueher M (ed) Principles and practice of constraint programming, springer international publishing, pp 539–555

6. Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Machine Intell 23(11):1222–1239
7. Carlucho I, Paula MD, Villar SA, Acosta GG (2017) Incremental q-learning strategy for adaptive pid control of mobile robots. Expert Syst Appl 80:183–199. https://doi.org/10.1016/j.eswa.2017.03.002. http://www.sciencedirect.com/science/article/pii/S0957417417301513
8. Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion: Theory, algorithms and implementation. MIT Press, Boston
9. Contardo C, Martinelli R (2014) A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discret Optim 12:129–146
10. Cortes J, Martinez S, Karatas T, Bullo F (2004) Coverage control for mobile sensing networks. IEEE Tranact Robot Autom 20(2):243–255
11. Dahl TS, Matarić M, Sukhatme GS (2009) Multi-robot task allocation through vacancy chain scheduling. Robot Auton Syst 57(6):674–687
12. Dai X, Laihao J, Zhao Y (2016) Cooperative exploration based on supervisory control of multi-robot systems. Appl Intell 45
13. Das P, Behera H, Panigrahi B (2016) Intelligent-based multi-robot path planning inspired by improved classical q-learning and improved particle swarm optimization with perturbed velocity. Eng Sci Technol Int J 19(1):651–669. https://doi.org/10.1016/j.jestch.2015.09.009, http://www.sciencedirect.com/science/article/pii/S2215098615001548
14. Durham J, Carli R (2012) Discrete partitioning and coverage control for gossiping robots. IEEE Trans Robot 28(2):364–378
15. Hidalgo-Paniagua A, Vega-Rodriguez MA, Ferruz J, Pavon N (2015) Mosfla-mrpp: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning. Eng Appl Artif Intell 44:123–136
16. Hussein A, Adel M, Bakr M, Shehata OM, Khamis A (2014) Multi-robot task allocation for search and rescue missions. J Phys Conf Series 570(5):052006
17. James S, Johns E (2016) 3d simulation for robot arm control with deep q-learning. arXiv:160903759
18. Janati F, Abdollahi F, Ghidary SS, Jannatifar M, Baltes J, Sadeghnejad S (2017) Multi-robot task allocation using clustering method. In: Robot intelligence technology and applications 4: Results from the 4th International Conference on Robot Intelligence Technology and Applications, Springer International Publishing, pp 233–247
19. Javanmard AR, Pimenta CAL (2014) Distributed safe deployment of networked robots. In: Proc. of the 12th international symposium on distributed autonomous robotic systems (DARS), pp 452–464
20. Javanmard Alitappeh R, Jeddisaravi K, Guimaraes FG (2016) Multi-objective multi-robot deployment in a dynamic environment. Soft Comput 1–17
21. Javanmard Alitappeh R, Pereira ASG, Araújo RA, Pimenta CAL (2017) Multi-robot deployment using topological maps. J Intell Robot Syst 86(3):641–661
22. Jeddisaravi K, Alitappeh RJ, Pimenta ALC, Guimarães FG (2016) Multi-objective approach for robot motion planning in search tasks. Appl Intell 45(2):305–321
23. Jiang L, Huang H, Ding Z (2019) Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. IEEE/CAA Journal of Automatica Sinica 1–11
24. Jianping C, Yimin Y, Yunbiao W (2009) Multi-robot task allocation based on robotic utility value and genetic algorithm. In: IEEE International conference on intelligent computing and intelligent systems, vol 2, pp 256–260
25. Jose K, Pratihar DK (2016) Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. Robot Auton Syst 80:34–42
26. Jung W, Yim J, Ko Y (2017) Qgeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic networks. IEEE Commun Lett 21(10):2258–2261
27. Khamis A, Hussein A, Elmogy A (2015) Multi-robot Task allocation: A Review of the State-of-the-Art. Springer International Publishing 31–51
28. Khani M, Ahmadi A, Hajary H (2019) Distributed task allocation in multi-agent environments using cellular learning automata. Soft Comput 23(4):1199–1218
29. Kim YG, Kwak JH, Hong DH, Ahn JH, Wee SG, An J (2013) Localization strategy based on multi-robot collaboration for indoor service robot applications. In: 2013 10th international conference on ubiquitous robots and ambient intelligence (URAI), pp 225–226
30. Konar A, Goswami Chakraborty I, Singh SJ, Jain LC, Nagar AK (2013) A deterministic improved q-learning for path planning of a mobile robot. IEEE Trans Syst Man Cybern Syst 43(5):1141–1153
31. Kong CS, Peng NA, Rekleitis I (2006) Distributed coverage with multi-robot system. In: Proceedings 2006 IEEE international conference on robotics and automation, 2006. ICRA 2006, pp 2423–2429
32. Kool W, van HoofH, Welling M (2019) Attention, learn to solve routing problems! arXiv:1803.08475
33. Korsah GA, Stentz A, Dias MB (2013) A comprehensive taxonomy for multi-robot task allocation. Int J Robot Res 32(12):1495–1512
34. Lacomme P, Moukrim A, Quilliot A, Vinot M (2017) A new shortest path algorithm to solve the resource-constrained project scheduling problem with routing from a flow solution. Eng Appl Artif Intell 66:75–86
35. Lalla-Ruiz E, Exposito-Izquierdo C, Taheripour S, Voss S (2016) An improved formulation for the multi-depot open vehicle routing problem. OR Spectrum 38(1):175–187
36. Lam CP, Chou CT, Chiang KH, Fu LC (2011) Human-centered robot navigation towards a harmoniously human-robot coexisting environment. IEEE Trans Robot 27(1):99–112
37. Lemaire T, Alami R, Lacroix S (2004) A distributed tasks allocation scheme in multi-uav context. In: Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, vol 4, pp 3622–3627
38. Li J, Pardalos PM, Sun H, Pei J, Zhang Y (2015) Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. Expert Syst Appl 42(7):3551–3561
39. Li J, Zhou M, Sun Q, Dai X, Yu X (2015) Colored traveling salesman problem. IEEE Trans Cybern 45(11):2390–2401
40. Lin YK, Chong CS (2015) Fast ga-based project scheduling for computing resources allocation in a cloud manufacturing system. Journal of Intelligent Manufacturing
41. Liu C, Kroll A (2015) Memetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks. Soft Comput 19(3):567–584
42. Liu Y, Liu H, Wang B (2017) Autonomous exploration for mobile robot using q-learning. In: 2017 2nd international conference on advanced robotics and mechatronics (ICARM), pp 614–619
43. Lloyd S (1982) Least squares quantization in pcm. IEEE Trans Inf Theory 28(2):129–137
44. Low ES, Ong P, Cheah KC (2019) Solving the optimal path planning of a mobile robot using improved q-learning. Robot Auton Syst 115:143–161. https://doi.org/10.1016/j.robot.2019.02.

013. http://www.sciencedirect.com/science/article/pii/S0921889018308285
45. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning arXiv:1312.5602. Comment: NIPS Deep Learning Workshop
46. Nair R, Ito T, Tambe M, Marsella S (2002) Task Allocation in the RoboCup Rescue Simulation Domain: A Short Note. Springer, Berlin, pp 751–754
47. Nazari M, Oroojlooy A, Snyder LV, Takáč M (2018) Reinforcement learning for solving the vehicle routing problem. arXiv:1802.04240
48. Nouri HE, Belkahla Driss O, Ghédira K (2016) Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. Appl Intell 45:808–828
49. dAndrea Novel B, Campion G, Bastin G (1995) Control of nonholonomic wheeled mobile robots by state feedback linearization. Int J Robot Res 14(6):543–559
50. Nowzari C, Cortes J (2012) Self-triggered coordination of robotic networks for optimal deployment. Automatica 48(6):1077–1087
51. de Oliveira FB, Enayatifar R, Sadaei HJ, Guimaraes FG, Potvin JY (2016) A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem. Expert Syst Appl 43:117–130
52. Ozturk S, Kuzucuoglu A (2015) Optimal bid valuation using path finding for multi-robot task allocation. J Intell Manuf 26
53. Parker LE (2002) Distributed algorithms for multi-robot observation of multiple moving targets. Autonomous Robots 12(3):231–255
54. Pimenta LCA, Kumar V, Mesquita RC, Pereira GAS (2008) Sensing and coverage for a network of heterogeneous robots. In: Proc. of IEEE conference on decision and control (CDC), vol 2, pp 3947–3952
55. Pimenta LCA, Schwager M, Lindsey Q, Kumar V, Rus D, Mesquita RC, Pereira GAS (2010) Simultaneous coverage and tracking (SCAT) of moving targets with robot networks. Springer, Berlin, pp 85–99
56. Poduri S, Sukhatme GS (2004) Constrained coverage for mobile sensor networks. In: Proc. of IEEE international conference on robotics and automation (ICRA). IEEE, pp 165–171
57. Pozna C, Precup RE, Koczy LT, Ballagi A (2002) Potential field-based approach for obstacle avoidance trajectories. IPSI BgD Trans Internet Res 8(2):40–45
58. Sadhu AK, Konar A, Bhattacharjee T, Das S (2018) Synergism of firefly algorithm and q-learning for robot arm path planning. Swarm and Evolution Comput 43:50–68. https://doi.org/10.1016/j.swevo.2018.03.014. http://www.sciencedirect.com/science/article/pii/S2210650217306776
59. Saeedvand S, Aghdasi H, Baltes J (2019) Robust multi-objective multi-humanoid robots task allocation based on novel hybrid metaheuristic algorithm. Appl Intell 49
60. Salhi S, Imran A, Wassan NA (2014) The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. Comput Oper Res 52:315–325
61. Sariel S, Balch T (2005) Integrating planning into scheduling. American Association for Artificial Intelligence (AAAI)
62. Schwarzrock J, Zacarias I, Bazzan ALC, de Araujo Fernandes RQ, Moreira LH, de Freitas EP (2018) Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence. Eng Appl Artif Intell 72:10–20
63. Su X, Wang Y, Jia X, Guo L, Ding Z (2018) Two innovative coalition formation models for dynamic task allocation in disaster rescues. J Syst Sci Syst Eng 27
64. Tai L, Liu M (2016) A robot exploration strategy based on q-learning network. In: 2016 IEEE international conference on real-time computing and robotics (RCAR), pp 57–62
65. Tang H, Sun W, Yu H, Lin A, Xue M, Song Y (2019) A novel hybrid algorithm based on pso and foa for target searching in unknown environments. Appl Intell 49
66. Tang L, Dian S, Gu G, Zhou K, Wang S, Feng X (2010) A novel potential field method for obstacle avoidance and path planning of mobile robot. In: 2010 3rd international conference on computer science and information technology, vol 9, pp 633–637
67. Tolmidis AT, Petrou L (2013) Multi-objective optimization for dynamic task allocation in a multi-robot system. Eng Appl Artif Intell 26(5):1458–1468
68. Trigui S, Cheikhrouhou O, Koubaa A, Baroudi U, Youssef H (2017) Fl-mtsp: A fuzzy logic approach to solve the multi-objective multiple traveling salesman problem for multi-robot systems. Soft Comput 21(24):7351–7362
69. Triki H, Mellouli A, Masmoudi F (2014) A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (ALRABP-2). J Intell Manuf 2
70. Wang J, Zhou Y, Wang Y, Zhang J, Chen CLP, Zheng Z (2016) Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms. IEEE Trans Cybern 46(3):582–594
71. Wei C, Hindriks K, Jonker C (2016) Dynamic task allocation for multi-robot search and retrieval tasks. Appl Intell 45:383–401
72. Xu L, Stentz A (2011) An efficient algorithm for environmental coverage with multiple robots. In: Robotics and automation (ICRA), 2011 IEEE International Conference on. IEEE, pp 4950–4955
73. Sk Yun, Rus D (2013) Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. Robotica 32(02):257–277
74. Zhang Q, Sg Yue, Qj Yin, Yb Zha (2013) Dynamic obstacle-avoiding path planning for robots based on modified potential field method. In: Intelligent computing theories and technology: 9th international conference, ICIC 2013, nanning, china, july 28-31 2013 Proceedings. Springer, Berlin, pp 332–342
75. Zhang Y, Dw Gong, Zhang JH (2012) Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 103:172–185
76. Zhao S, Chen BM, Lee TH (2013) Optimal sensor placement for target localisation and tracking in 2d and 3d. Int J Control 86(10):1687–1704

Reza Javanmard Alitappeh is currently an Assistant Professor at the University of Science and Technology of Mazandaran, Iran. In 2019 he worked as a research fellow at Visual Artificial Intelligence Laboratory in Oxford Brookes University (UK) with Prof Fabio Cuzzolin and Dr Bradley. He completed his PhD in 2016 in artificial intelligence and robotics under the supervision of Prof Pimenta and Chaimowicz at the Federal University of Minas Gerais, Brazil. He is a reviewer for various journals and conferences.

Kossar Jeddisaravi received her MSc degree in Computer Systems and Robotics from the Federal University of Minas Gerias, Brazil, in 2014, with a dissertation on multi-objective robot exploration. She obtained a PhD from the same university in 2017, under the supervision of Prof Frederico Guimaraes. Since 2017 she is a lecturer in programming language and image processing at the University of Science and Technology of Mazandaran. Her research interests include multi-agent systems and machine vision.