A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario

Wanqing Zhao, *Member, IEEE*, Qinggang Meng, *Member, IEEE*, and Paul W. H. Chung

***Abstract*—Using distributed task allocation methods for cooperating multivehicle systems is becoming increasingly attractive. However, most effort is placed on various specific experimental work and little has been done to systematically analyze the problem of interest and the existing methods. In this paper, a general scenario description and a system configuration are first presented according to search and rescue scenario. The objective of the problem is then analyzed together with its mathematical formulation extracted from the scenario. Considering the requirement of distributed computing, this paper then proposes a novel heuristic distributed task allocation method for multivehicle multitask assignment problems. The proposed method is simple and effective. It directly aims at optimizing the mathematical objective defined for the problem. A new concept of significance is defined for every task and is measured by the contribution to the local cost generated by a vehicle, which underlies the key idea of the algorithm. The whole algorithm iterates between a task inclusion phase, and a consensus and task removal phase, running concurrently on all the vehicles where local communication exists between them. The former phase is used to include tasks into a vehicle's task list for optimizing the overall objective, while the latter is to reach consensus on the significance value of tasks for each vehicle and to remove the tasks that have been assigned to other vehicles. Numerical simulations demonstrate that the proposed method is able to provide a conflict-free solution and can achieve outstanding performance in comparison with the consensus-based bundle algorithm.**

***Index Terms*—Distributed task allocation, multitask, multivehicle, overall objective, search and rescue.**

I. Introduction

RESEARCH into multivehicle systems has been increasingly drawing a wide interest over a single vehicle as it is more capable of accomplishing difficult and complex missions in an effective and efficient manner and is more fault tolerant [1]–[6]. In a multivehicle system, a mission can be divided into different tasks and a number of specialized vehicles are then introduced to solve each task concurrently. These tasks may be known by the vehicles before task execution stage or may be dynamically appeared during task execution [7]. For example, a team of autonomous vehicles are executing an exploration mission [8]. The aim is thus to locate and visit a number of predetermined targets in a partially unknown terrain. The challenge is then how to assign these targets to the vehicles in order to optimize an overall system objective required by the mission.

The multivehicle task allocation problem (where multiple vehicles are employed for task allocation) has been shown to be NP-hard and, therefore, obtaining a globally optimal solution is computationally intensive [9], [10]. Most algorithms thus provide suboptimal solutions. Early research used centralized approaches to generate a plan for cooperating all the vehicles by using a central server who is able to gather the whole system information (situational awareness) [11], [12]. In such approaches, an overall objective function for the problem of interest can explicitly be minimized or maximized based on the full collection of every vehicle's information on the central server. The main advantage thus lies on their ability of the optimization of the overall objective function (thus for obtaining a solution which is optimal or very close to it). However, they suffer from several weaknesses especially for a large-scale system [13]. First, they require consistent and complete communication between the central server and each vehicle, which places a heavy communication burden on the server and also reduces the mission coverage area. Second, the computational demand is high as the computational operations are all placed on the central server which also has to keep track of changes in substantially internal knowledge of each vehicle as the mission progresses. Third, centralized approaches are vulnerable to the single point of failure.

Therefore, multivehicle task allocation problems are often solved in a distributed manner using market-based mechanisms, while auction approaches are among the most popular since they are efficient in terms of both computation and communication. The information of the vehicles and tasks can be compressed into numerical bids and computed in parallel by each vehicle. For single-assignment problems where each vehicle can handle at most one task, the single-item auctions [7], [14] can be used where vehicles bid on tasks that are auctioned off individually. The vehicle with the highest bid wins the task and then has to finish it. However, for multiassignment problems where each vehicle is able to handle several tasks, they are not so easy to be solved as essentially they belong to the class of combinatorial optimization problem.

Manuscript received October 20, 2014; revised January 20, 2015; accepted March 24, 2015. Date of publication April 13, 2015; date of current version March 15, 2016. This work was supported by the U.K. Engineering and Physical Sciences Research Council Autonomous and Intelligent Systems Programme under Grant EP/J011525/1 with BAE Systems as the leading industrial partner. This paper was recommended by Associate Editor J. Su.

W. Zhao is with the Cardiff School of Engineering, Cardiff University, Cardiff CF24 3AA, U.K. (e-mail: W.Q.Zhao@ieee.org).

Q. Meng and P. W. H. Chung are with the Department of Computer Science, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: Q.Meng@lboro.ac.uk; P.W.H.Chung@lboro.ac.uk).

Digital Object Identifier 10.1109/TCYB.2015.2418052

Generally speaking, strong synergies exist between the tasks for bidders. It is considered that a set of tasks have a positive synergy for a vehicle if their combined cost for executing them together is less than the sum of their individual cost incurred by doing them separately, and conversely for a negative synergy. On the one hand, one can achieve a near-optimal allocation of a set of tasks to vehicles by using single-round combinatorial auctions [15], [16]. It works by calculating the bid for every vehicle, based on bundles of tasks rather than individual tasks. The bid for each vehicle to hold a bundle is computed through the smallest path cost that needs to visit all tasks in the bundle from the vehicle's current location. However, it is computationally inefficient to bid for all possible bundles of tasks as the number exponentially increases with the number of tasks. On the other hand, parallel single-item auctions treat each task independent of other tasks and every vehicle bids for each task in parallel. Such mechanism has its computation and communication efficiencies while it inevitably lead to highly suboptimal solutions since it does not account for any synergies between tasks. On balance, the sequential (multiround) single-item auctions [17], [18] provide the advantages of solution quality from single-round combinatorial auctions, and computation and communication efficiencies from parallel single-item auctions. It works in a multiround manner and in each round every vehicle places bid on the unallocated tasks. The bid is computed as the smallest cost increase resulted from winning the task and the vehicle with the overall smallest bid is allocated the corresponding task. The process is repeated until all the tasks have been allocated.

Other distributed auction-based algorithms have been developed to remove the auctioneer and place the algorithms on each vehicle. In such cases, consensus can be used to handle those identical variables in different vehicles (as they may be assigned with different values), thus resulting a conflict-free solution amongst all vehicles. The consensus-based bundle algorithm (CBBA) proposed by Choi *et al.* [19] has attracted a lot of research interests. It iterates between a bundle construction phase where a single bundle is constructed and continuously updated as the auction proceeds, and a conflict resolution phase which is used to reach consensus on the winning bids and to release tasks that are outbid. Recently, a number of modifications have been suggested to extend its functions and application areas [20]–[22]. Basically, they all rely on the market-based auction principles for task allocation followed by a consensus process on the bidding values. In this paper, a novel distributed task allocation method for solving the multivehicle multitask problem is proposed. Unlike the auction-based method, the proposed method directly aims at optimizing the overall objective mathematically formulated under the problem of interest. The key concept is called significance. The value of significance is calculated as the local contribution of a task to a vehicle, which reflects the importance of a task to the local cost generated by the vehicle. In this way, the overall cost of the objective function can be decreased if certain criterion is satisfied on these significance values when switching tasks between different vehicles. The proposed method iterates between a task inclusion phase, and a consensus and task removal phase. The former phase is used to include tasks into a vehicle's task list for decreasing the overall cost, while the latter is to reach consensus on the significance value of tasks for each vehicle and to remove the tasks that have been included into other vehicles' task list. The two phases run concurrently on all the vehicles where limited local communication exists between them. Various numerical simulations are finally presented to demonstrate the effectiveness and outstanding performance of the proposed method in comparison with the CBBA [19].

In addition, regarding the difference between the proposed method and CBBA, it mainly lies in their different working principles. CBBA is based on auction principles where generally each vehicle is selfish on reducing the local cost generated by it, while the proposed method aims directly at optimizing the overall objective for the problem of interest. In detail, in CBBA, the largest bid is utilized by the vehicle to include a new task in the task list at each step. Instead, our method works heuristically on optimization principles and the largest difference of the significance is pursued at each step for the purpose of decreasing the overall cost for the problem. The bid and significance also have obviously different meanings. A bid in CBBA is calculated according to an extra bundle list for recording which tasks were first added into the task list of a vehicle. Thus, if the inclusion sequence of tasks has been changed the bids for the corresponding tasks are no longer valid. The bids for all the tasks cannot be traced if only a solution is given for the whole problem of interest. This also explains why the CBBA method needs to remove all the tasks after the outbidded task in the bundle list. However, the significance concept in the proposed method is directly assessed based only on the task list and is not affected by different task inclusion sequences. In other words, it has no relationship with the intermediate process of an algorithm and can be determined as long as a task list is given for a vehicle. Therefore, our method can easily be applied to the results produced by other methods by taking their results as the input in order to further improve the overall system objective. In particular, apart from the different working principles, it is also worth mentioning that the basic difference between the proposed method and sequential single-item auctions also lies in two other aspects. On the one hand, the proposed method is fully distributed with a planner placed on every vehicle and cooperating with the linked local neighbors, while the sequential auctions usually have an auctioneer to receive all the bids from the vehicles which places limitations on the network topologies. On the other hand, the proposed method runs concurrently on every vehicle by allocating tasks in parallel, while the sequential auctions simply allocate tasks in sequence.

This paper is organized as follows. Section II presents the scenario description and system configuration. The objective for the problem of interest is then given in Section III together with its mathematical formulation. Section IV proposes the distributed task allocation method for multivehicle multitask problems. The effectiveness of the proposed method is then demonstrated through several simulated scenarios in Section V. Finally, Section VI concludes this paper.

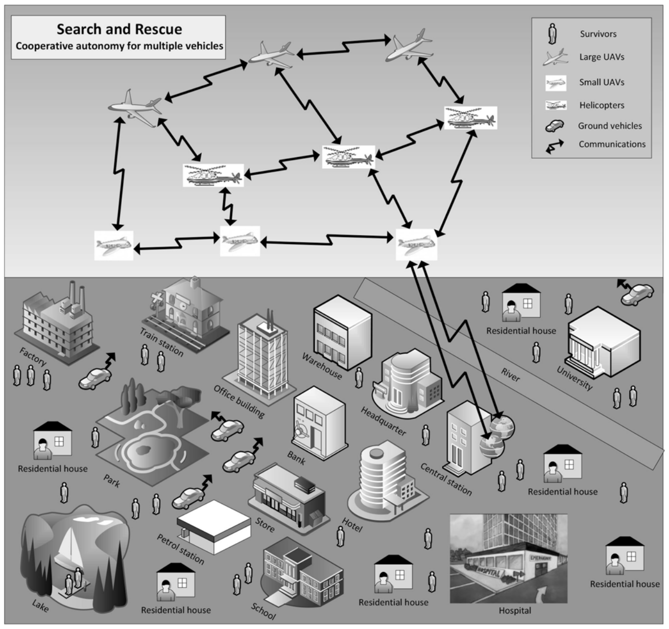


Fig. 1. Search and rescue scenario with heterogeneous unmanned vehicles: the aerial vehicles fly over together with the ground vehicles cruising in an area where the survivors are believed to be located, gathering information of potential survivors and cooperating between one another to rescue the survivors found.

II. General Scenario Description

Consider a variety of heterogeneous unmanned vehicles (such as large unmanned aerial vehicles (UAVs), small UAVs, helicopters, ground vehicles) performing a search and rescue mission on a bounded terrain, the purpose is to locate and rescue a number of survivors whose positions are partially unknown. The vehicles always know their own current location and are capable of providing critical support to the search and rescue mission. As shown in Fig. 1, heterogeneous vehicles are deployed in a complex area of interest, performing sensory operations to detect location and capture other relevant information of the survivors, and meanwhile cooperating between one another to rescue the survivors discovered. The cooperative system for this search and rescue mission is thus studied here, where each survivor has to be visited by one vehicle to get rescued. The vehicles do not need to return to their initial location after finishing each assigned task and every vehicle always follows a minimum time cost path to visit all of the unvisited targets allocated to it.

According to the nature of the underlying problem, the heterogeneous unmanned vehicles can be generally divided into a search and rescue teams as shown in Fig. 2. The former is used to find the survivors together with their location and other relevant health condition information. The latter is responsible for providing rescue operation. The general working mechanism of the cooperative systems is illustrated in Fig. 2. A reasoning system which at the beginning uses search team situation including all the relevant vehicles' information together with the terrain information as the input is applied to generate search plan for the search team to find potential survivors. Whenever a vehicle gains more information about the terrain, it shares this information with other vehicles.

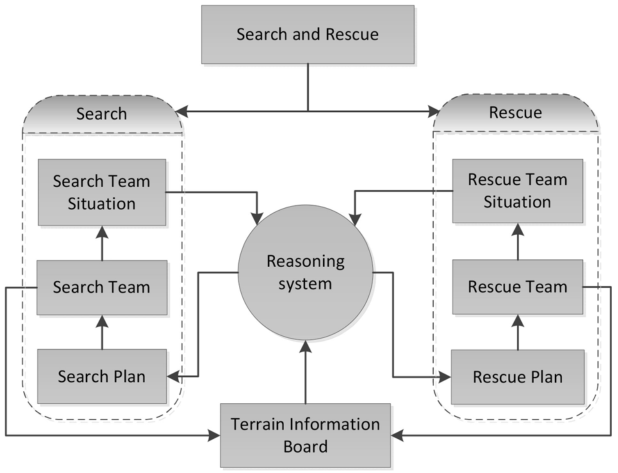


Fig. 2. General system architecture for search and rescue.

As long as some survivor(s) are discovered, the reasoning system in the meantime is capable of making appropriate decisions on the rescue team for rescuing the discovered survivors by also considering the input information from the current rescue team situation and the terrain information. Given this is a dynamic system, the search and rescue operations work in parallel and interact with each other. The key problem here is to design an effective reasoning system to meet all these requirements and the major challenges then arise in how to assign these different tasks amongst all vehicles in order to satisfy the system objective. If the remaining path to the unvisited tasks of at least one vehicle is blocked or the terrain information is updated, then all vehicles put up their unvisited tasks for reallocation.

Given the key features of this reasoning system, potentially there are several ways to realize it from adopting a centralized planner to several distributed planners, which can be generally categorized as follows.

1) *One Planner:* There is a centralized planner located at a ground workstation or an extra server, being responsible for producing a whole plan for all vehicles included in the scenario.
2) *Two Planners:* There are two centralized planners located at a ground workstation or an extra server, being used to generate the search and rescue plans separately for the search and rescue teams. The two planners can interchange the information they have with each other during their reasoning process.
3) *Multiple Planners:* There are several planners located at the different distributed vehicles, each being adopted to produce a plan for a vehicle or a local region of vehicles. The planners at different location may communicate with one another to exchange the information and to negotiate their designed solutions.
4) *Mixed Planners:* A mixed realization of the above two types of planners, in which one of the team (search or rescue) shares a common centralized planner and the other one uses multiple planners.

It is worth mentioning that the actual system working architecture can vary from the different types of planners adopted for the scenario, and it depends on whether the planners are centralized or distributed. Given the overall picture of the whole problem, in this paper, we are only interested in using multiple planners for the rescue team, where each rescue vehicle (with limited local communication to others) is equipped with a planner to cooperate with others to produce and execute the final plan to support the survivors. The search team, however, can use a similar mechanism to find the survivors and notify the rescue team. Without loss of generality, providing a specific rescue support to a survivor is simply regarded as a task in the following distributed task allocation method to be presented.

III. Objective and Mathematical Formulation

Generally speaking, two categories of optimization criteria can be used in task allocation to achieve some stated system objective, i.e., maximization (reward based) or minimization (cost based) [23]. The first category is adopted when there is a reward in assigning a task to some vehicle, while the second can practically be more applicable when a cost exists for performing a task and it can be further divided by the following.

1) *MiniSum:* Minimizing the sum of the path cost over all vehicles.
2) *MiniMax:* Minimizing the maximum path cost over all vehicles.

Here, the path cost of a vehicle is the sum of the costs generated from performing each of the ordered tasks along this vehicle's task list. The MiniSum criterion can be used to minimize the total usage of resources consumed by all vehicles or the average cost associated with performing each single task. For example, if the cost is fuel consumption, then MiniSum minimizes the total fuel consumed by all vehicles. Whilst the cost is travel time, MiniSum can minimize how long it takes on average for a task to be completed (average task completion time). The MiniMax is to minimize the cost of the worst performing vehicle. If the cost is travel time, then MiniMax minimizes the whole time that all tasks have been executed (mission completion time).

For the search and rescue mission considered in this paper, the time consumed on rescuing a survivor is the key factor concerned as a survivor's health condition deteriorates until a vehicle comes to rescue it. This conducts the utilization of the MiniSum criterion together with time costs for the problem of interest. Consider, for example, as shown in Fig. 3, v_1 denotes the location of a vehicle and t_1, t_2, t_3, and t_4 denote the locations of four survivors. Apparently, there are strong synergies between these tasks, where the positive synergies are existed between t_2, t_3, and t_4 for the vehicle and a negative synergy is existed between t_1 and t_2. The vehicle has a travel distance/time of three units to t_1, four units to t_2, five units to t_3, and six units to t_4, from its current location. Here, assume that every survivor is associated with a different deadline for rescuing it (based on the survivor's health condition, which is formulated as a constraint to be presented later) and all the survivors get rescued before their deadlines in the two assignments 1 and 2 as shown in Fig. 3. By ignoring the task execution duration, the actual times for rescuing the survivors are shown in both assignments. In the assignment 1 (resulted from the MiniSum criterion with fuel costs, or the MiniMax criterion with either time costs or fuel costs), rescuing the survivor t_1 first and then the survivors t_2, t_3, and t_4 can attain both least fuel consumption and earliest mission completion time. However, the latter three survivors t_2, t_3, and t_4 are visited very late and the average rescuing time is 36/4. On the contrary, in the assignment 2 (resulted from the MiniSum criterion with time costs), rescuing the latter three first and then the survivor t_1 attains the least average amount of rescuing time (30/4) and thus obtains an overall better average health condition of the survivors.

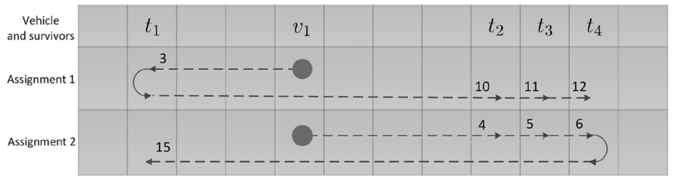


Fig. 3. Illustration of the assignments given by different optimization criteria (v_1 represents the vehicle; t_1, t_2, t_3, and t_4 represent the four survivors; assignment 1: start with rescuing t_1; assignment 2: start with rescuing t_2; the numbers are the time for rescuing each survivor under each assignment).

To formulate the problem of interest mathematically, $\boldsymbol{V} = [v_1, \ldots, v_n]^{\mathrm{T}}$ is defined as the set of n heterogeneous unmanned vehicles and $\boldsymbol{T} = [t_1, \ldots, t_m]^{\mathrm{T}}$ as the set of m survivors to be rescued. An allocation $\boldsymbol{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]^{\mathrm{T}}$ is a partition of $\boldsymbol{T}$ where $\mathbf{a}_i, i = 1, \ldots, n$, is an ordered list of survivors assigned to be rescued by vehicle v_i. The objective for the scenario can be expressed as

$$J = \min\left\{\frac{1}{m}\sum_{i=1}^{n}\sum_{\kappa=1}^{\alpha_i} c_{i,\kappa}(\mathbf{a}_i)\right\} \tag{1}$$

where $c_i(\mathbf{a}_i)$ denotes the total time cost incurred by vehicle v_i rescuing all the ordered survivors in $\mathbf{a}_i$, $c_{i,\kappa}(\mathbf{a}_i)$ is the time cost incurred by vehicle v_i rescuing the κth survivor in $\mathbf{a}_i$, and α_i denotes the number of survivors assigned to v_i.

For the convenience of formulating the constrains for the underlying problem, the vehicle and survivor properties are first described as follows. The vehicle property consists of vehicle ID number (v_i), vehicle type (e.g., large UAVs, small UAVs, helicopters, ground vehicles, etc.), vehicle spatial location ($x_i^{v)}$, $y_i^{v)}$, $z_i^{v)}$), and vehicle cruising speed (ω_i, m/s). On the other hand, the survivor property involves survivor ID number (t_k), survivor type (e.g., needing water, food, medicine, paramedic, doctor, etc.), deadline for starting to rescue a survivor (s_k), execution duration (p_k) for rescuing a survivor, survivor spatial location ($x_k^{t)}$, $y_k^{t)}$, $z_k^{t)}$). Moreover, a symmetric communication matrix $\mathbf{G}(t)$ is also defined to describe the communication network amongst all the vehicles at time instant t, where the entry $g_{i,j}(t) = 1$ represents that an instant connection exists between vehicles v_i and v_j at time t and 0 otherwise. Communication takes place between two vehicles only when there exists a connection link between them. It should be noted that each type of vehicle has a speciality in performing only the same type of tasks, like dropping bottled water or food.

With the definitions of the survivor list $\mathbf{a}_i$, the time cost $c_{i,\kappa}(\mathbf{a}_i)$, and the vehicle and task properties, the constrains involved in the optimization of the aforementioned objective (1) can now be formulated as

$$|\mathbf{a}_i| \leq m_i \tag{2}$$

$$\bigcup_{i=1}^{n} \mathbf{a}_i = \boldsymbol{T}, \quad \mathbf{a}_i \bigcap \mathbf{a}_j = \emptyset \text{ with } i \neq j \tag{3}$$

$$\text{Compatibility matrix } \mathbf{H} \text{ with entries } h_{i,k} = 0, 1 \tag{4}$$

$$c_{i,\kappa}(\mathbf{a}_i) \leq s_{a_{i,\kappa}}. \tag{5}$$

Here, formula (2) represents the capability of each vehicle, where $|\cdot|$ denotes the cardinality of the list, and the vehicle v_i can only be assigned a maximum of m_i survivors. In (3), it is used to guarantee that the allocation $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ is a valid partition of the whole set of survivors. As the vehicles are heterogeneous, it is necessary to define a compatibility matrix in (4), in which $h_{i,k} = 1$ if a vehicle of type i is able to perform a task of type k and 0 otherwise. Formula (5) is just a simple constraint on the start time for rescuing the κth survivor $(a_{i,\kappa})$ in $\mathbf{a}_i$ by vehicle v_i. This is because each survivor has to be getting rescued before the deadline due to its deteriorating health condition. Thus, it would be failed if a survivor assigned to a vehicle is rescued after its associated deadline. Since the fact that a series of survivors can be included in a task list to a vehicle, the value of $c_{i,\kappa}(\mathbf{a}_i)$ is usually confined with the time costs for rescuing other survivors in the task list.

IV. Proposed Distributed Task Allocation Method

This section describes the proposed distributed task allocation method. A novel concept of significance is first defined, followed by the basic idea of the proposed method. To better facilitate the distributed computing for the task allocation as discussed in Sections I and II and to avoid local optima caused by local communication, the whole distributed task allocation method is then fully presented. It will be shown that the proposed method aims directly at optimizing the defined objective while meeting the constraints as presented in Section III for the problem of interest.

A. Basic Idea

Suppose, we have an allocation for all the vehicles at some point during the task allocation process, the significance of task t_k with regard to its assigned vehicle v_i is first defined as

$$w_{k,i}(\mathbf{a}_i \ominus t_k) = c_i(\mathbf{a}_i) - c_i(\mathbf{a}_i \ominus t_k) \tag{6}$$

where $\mathbf{a}_i \ominus t_k$ denotes removing task t_k from the corresponding position of the ordered task list $\mathbf{a}_i$. In the search and rescue scenario, apart from the disappearance of the time cost related with t_k in $c_i(\mathbf{a}_i \ominus t_k)$, it is worth pointing out that the removal of the task t_k may also cause the decrease to other time costs $c_{i,\kappa}(\mathbf{a}_i)$ $(\kappa = 1, \ldots, \alpha_i;\ a_{i,\kappa} \neq t_k)$ from performing other tasks in the reduced list due to the inside synergies. On the contrary, if the removed task is again inserted at the same position in the task list from which it was removed, it will lead to the same value of significance including the cost of itself and the variation of costs from other tasks. Thus, the value of significance is bidirectional, which truly reflects the contribution of a task to the local cost generated by a vehicle.

The basic idea of the proposed method is now explained as follows. First, the significance of task t_k computed from vehicle v_i as described above is transmitted to another neighbor vehicle v_j where a communication exists between them, i.e., $g_{i,j}(t) = 1$. Second, the marginal significance $w^{\star}_{k,j}(\mathbf{a}_j \oplus t_k)$ by adding t_k into the currently ordered task list $\mathbf{a}_j$ of vehicle v_j is calculated by

$$w^{\star}_{k,j}(\mathbf{a}_j \oplus t_k) = \min_{l=1}^{|\mathbf{a}_j|+1} \{c_j(\mathbf{a}_j \oplus_l t_k) - c_j(\mathbf{a}_j)\} \tag{7}$$

where $\mathbf{a}_j \oplus_l t_k$ denotes adding task t_k at the lth position in the task list $\mathbf{a}_j$. The operator min is used to find the minimal local cost increment by inserting t_k at a proper position in $\mathbf{a}_j$. Finally, by comparing the changes in the two local costs incurred by v_i and v_j as a result of the movement of t_k, it is now obvious that the overall cost can be decreased if the criterion

$$w_{k,i}(\mathbf{a}_i \ominus t_k) > w^{\star}_{k,j}\big(\mathbf{a}_j \oplus t_k\big) \tag{8}$$

is satisfied and the improvement to the overall cost is given by the difference between the above two values. Then, the task t_k is removed from the task list of vehicle v_i and included into vehicle v_j's, thus giving the overall cost reduction. In this way, the significance of task t_k is further transmitted to other vehicles and continuously updated, subject to the decrease of the overall cost. The same process is repeated until no reduction on the cost can be found by exchanging all the tasks between the vehicles. The convergence is guaranteed as whenever a change is made to the task allocation will decrease the overall cost. It should also be noted that when $i = j$ in (8), the equality $w^{\star}_{k,j}(\mathbf{a}_j \oplus t_k) = w_{k,j}(\mathbf{a}_j \ominus t_k)$ should be followed. The inclusion of task t_k could thus cause the increase to the time cost $c_{j,\kappa}(\mathbf{a}_j)$ $(\kappa = 1, \ldots, \alpha_j)$ of the tasks in $\mathbf{a}_j$. Thus far, the basic idea is based on the global optimization of the original objective defined in (1). It is not like other distributed methods which do not explicitly go for the global objective, where they are often selfish on optimizing the local cost defined for each vehicle as global information for all the vehicles and tasks is not available in the centralized manner. In the proposed method, through the transmission of the significance value of the tasks amongst every vehicle via local communication, the overall cost can successively be decreased by performing appropriate operations locally on each task list. As a result, the global information imposed in a centralized optimization approach is relaxed by the idea presented here.

However, since the nature of distributed computing, it is not difficult to find out that the basic idea of the proposed method can easily get stuck in local optima, especially for the task allocation problem under employing heterogeneous vehicles. For example, suppose we have an intermediate allocation at some point during the algorithm running process as shown in Fig. 4. A row communication network is applied here where all the vehicles are widely distributed and there is a relatively low communication between them. There are a total of six survivors who need to be provided with food and another six

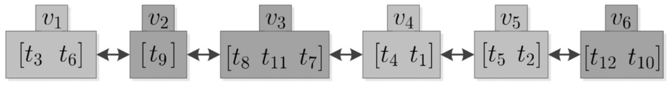


Fig. 4. Illustration of local optima found under a row communication network in the basic idea of the proposed method.

survivors with medicine, which can be solved by the corresponding two types of heterogeneous vehicles. Specifically, the first six survivors (i.e., $t_1, \dots, t_6$) can be taken care of by vehicles v_1, v_4, v_5, while the rest by vehicles v_2, v_3, and v_6. According to the previous discussions, we now assume that the overall cost can be reduced if task t_7 is going to be removed from the task list of v_3 and to be added into the task list of v_6. But this could not happen since vehicles v_4 and v_5 are not able to include t_7 into their task list due to heterogeneity, which in turn means the only channel for gradually transmitting and updating the significance of task t_7 from v_3 toward v_6 has been blocked. Thus, the local optimum has reached at this stage. This phenomenon could also occur in the case of using homogeneous vehicles when some local optima have been reached between part of the communicated vehicles during the algorithm running process, as these local optima could affect the transmission of task significance to other vehicles who may be able to perform the corresponding task with reduced overall cost.

B. Task Inclusion Phase

Given the above consideration, it is natural to consider circulating the significance of every task amongst all vehicles. In this way, every vehicle is able to know the significance of all the tasks and thus to compare them with the corresponding marginal significance calculated under their current task list. Thus, the significance of a task can be updated in the global manner, resulting in better objective values for the problem of interest. The proposed algorithm is capable of running on each vehicle concurrently. It is now described as follows by first assuming that a significance list is defined by $\boldsymbol{\gamma}_i = [\gamma_{i,1}, \dots, \gamma_{i,m}]^{\mathrm{T}}$ on vehicle v_i for recording all the tasks' significance. This significance list is stored at every vehicle ($i = 1, \dots, n$) and is able to be updated through communicating with other vehicles. The ideal case is that each vehicle has an identical copy of the significance list, which means that all the vehicles have reached the global consensus between one another, to be discussed in more detail in Section IV-C.

Given the significance list and the current ordered task list of a vehicle, the next step is to compute the marginal significance according to (7) for all the tasks not included in the current task list. In particular, since the start rescuing time is considered as the cost for the search and rescue scenario with the objective defined in (1), the marginal significance of task t_k with regard to vehicle v_i can be calculated as

$$w^{\star}_{k,i}(\mathbf{a}_i \oplus t_k) = \min_{l=1}^{|\mathbf{a}_i|+1} \left\{ c_{i,l}(\mathbf{a}_i \oplus_l t_k) + \sum_{r=l}^{|\mathbf{a}_i|} c_{i,r+1}(\mathbf{a}_i \oplus_l t_k) - \sum_{r=l}^{|\mathbf{a}_i|} c_{i,r}(\mathbf{a}_i) \right\}, \quad k = 1, \dots, m. \tag{9}$$

The time cost for the tasks after the new inserting task in the task list could be increased as the start rescuing time for these tasks may have been changed. As a result, a list for storing all the task marginal significance on vehicle v_i ($i = 1, \dots, n$) can thus be formulated by $\boldsymbol{\gamma}^{\star}_i = [w^{\star}_{1,i}, \dots, w^{\star}_{m,i}]^{\mathrm{T}}$, where $w^{\star}_{k,i}$ ($k = 1, \dots, m$) is used for representing $w^{\star}_{k,i}(\mathbf{a}_i \oplus t_k)$ for simplicity purpose. One can use an infinity value to denote those marginal significance related to the tasks that have been already included in the task list or that do not satisfy the constraints in (4) and (5), in order to exclude them from the current task inclusion phase.

It is now ready to decide whether a new task should be inserted into the current ordered task list $\mathbf{a}_i$ on vehicle v_i ($i = 1, \dots, n$) according to the following criterion:

$$\max_{k=1}^{m} \{\gamma_{i,k} - \gamma^{\star}_{i,k}\} > 0, \quad k = 1, \dots, m \tag{10}$$

by comparing the global task significance list $\boldsymbol{\gamma}_i$ with the task marginal significance list $\boldsymbol{\gamma}^{\star}_i$, where $\gamma_{i,k}$ and $\gamma^{\star}_{i,k}$ are the corresponding entries from them, respectively. If the criterion is met, the task corresponding to the largest objective reduction, i.e., $t^{i)}_{\ell} = \arg\max_{k=1}^{m}\{\gamma_{i,k} - \gamma^{\star}_{i,k}\}$ is added into the ordered task list $\mathbf{a}_i$ at the position $l^{i)}_{\ell} = \arg w^{\star}_{\ell,i}(\mathbf{a}_i \oplus t^{i)}_{\ell})$. The significance of the task $t^{i)}_{\ell}$ in the significance list is updated as $\gamma_{i,\ell} = \gamma^{\star}_{i,\ell}$. The corresponding information about the tasks in the task list of v_i, such as the time cost, is thus updated. To facilitate the consensus of the significance list across all the vehicles to be discussed later, another vehicle list $\boldsymbol{\beta}_i = [\beta_{i,1}, \dots, \beta_{i,m}]^{\mathrm{T}} \in \Re^m$ noting the assigned vehicle $\beta_{i,k}$ to each task t_k ($k = 1, \dots, m$) known by vehicle v_i will also be defined. Thus, the ℓth element of the vehicle list $\boldsymbol{\beta}_i$ for vehicle v_i is updated with $\beta_{i,\ell} = v_i$. By again computing the task marginal significance (9) and satisfying the criterion (10), the above process continues. The process stops when the criterion is no longer met or the maximum number of m_i tasks has been included in the task list as defined in (2). The significance list $\boldsymbol{\gamma}_i$ needs to be updated according to (6) when this process is terminated as the inclusion of new tasks into the vehicle's task list may change the significance of other tasks in the list $\mathbf{a}_i$. Similar as in (9), the significance of the tasks included in the task list $\mathbf{a}_i$ for the search and rescue scenario can be calculated as

$$w_{k,i}(\mathbf{a}_i \ominus t_k) = c_{i,b}(\mathbf{a}_i) + \sum_{r=b+1}^{|\mathbf{a}_i|} c_{i,r}(\mathbf{a}_i) - \sum_{r=b+1}^{|\mathbf{a}_i|} c_{i,r-1}(\mathbf{a}_i \ominus t_k), \quad k \in \mathbf{a}^{\mathrm{T}}_i \tag{11}$$

where b denotes the position of the task t_k in the task list, i.e., $a_{i,b} = t_k$.

C. Consensus and Task Removal Phase

As described by the aforementioned basic idea of the proposed method, the significance of the tasks in the task list of a vehicle should be transmitted to other vehicles in order to notify them of removing the corresponding tasks from their task list and to get the significance values further decreased.

If a vehicle receives a significance value smaller than the one it has produced for a particular task already in its task list, this task is then required to be removed from the task list. The significance of the tasks assigned to every vehicle can be broadcast to other vehicles through local communications between one another. Since the algorithm is running on all vehicles concurrently, different significance values for a task can occur for different vehicles. A consensus stage for reaching the global significance value for every task is thus required. This global significance value of a task can only originally be provided by one vehicle with the task contained in its task list and is transmitted to the other vehicles via local communications. Apart from determining whether to remove a task from the current task list, the task significance after consensus on every vehicle is also used for determining whether to add new tasks into the current task list as presented in Section IV-B. In this manner, several iterations of attempting to reach consensus of task significance will lead to a conflict-free assignment of all the tasks. The consensus and task removal phase is now discussed in detail as follows.

Although the significance value of a task generally becomes smaller during the algorithm running process, possible larger value still occurs given that the strong synergies are existed between the tasks. (The significance of a task is dependent on other tasks in the task list and thus their changes.) A direct broadcast of the significance list of each vehicle to the other vehicles through a series of local communications to find the smallest significance value for every task in order to obtain the convergence of the global significance list across all the vehicles is improper since some significance value for a task might converge to a value nonexisted (in the sense of not being provided by any vehicle under its task list). Due to the significance value of every task can vary in both directions, i.e., becoming either smaller or larger, the consensus of it to the smallest one which is also valid in the sense of being provided under some vehicle's task list, is needed. The heuristic rules presented in CBBA for the consensus of the so-called winning bids are employed here to reach the consensus of the significance list through enabling the consensus value generally moving downward instead of upward. Similar as in CBBA, a vehicle list $\boldsymbol{\beta}_i$ and a time stamp list $\boldsymbol{\delta}_i$ are also utilized in assistance with updating a proper global significance list.

After a local copy of significance list $\boldsymbol{\gamma}_i$ on vehicle v_i has been updated by communicating with the neighbors, it is now ready to determine if any tasks in the task list $\mathbf{a}_i$ should be removed. First of all, a set of tasks intended to be removed from the task list are found by $\mathbf{d}_i = \mathbf{a}_i[\boldsymbol{\beta}_i[\mathbf{a}_i] \neq v_i]$. The criterion for removing a task is given by

$$\max_{k=1}^{|\mathbf{d}_i|}\left\{\boldsymbol{\gamma}_i^{\diamond}[\mathbf{d}_i] - \boldsymbol{\gamma}_i[\mathbf{d}_i]\right\} > 0 \tag{12}$$

where $\boldsymbol{\gamma}_i^{\diamond}$ denotes the significance vector of the corresponding tasks, computed from the current task list of vehicle v_i. Then, the task associated with the largest objective reduction is removed from the list $\mathbf{a}_i$ and the set $\mathbf{d}_i$. The time cost of the remaining tasks in $\mathbf{a}_i$ and the significance of the remaining tasks in $\mathbf{d}_i$ are then updated due to the removal of a task. Continuing this removal process until the criterion is no longer satisfied or the set $\mathbf{d}_i$ becomes empty. Finally, the remaining tasks in the set $\mathbf{d}_i$ (if they do exist) have smaller significance value after the task removal process and they are retained in the task list $\mathbf{a}_i$. The present vehicle ID is, therefore, needed to be put back to the vehicle list, i.e., $\boldsymbol{\beta}_i(\mathbf{d}_i) = v_i$. The algorithm then moves to the task inclusion phase as described previously. The whole algorithm stops when no changes can be made in the task inclusion and removal phases for a period of time. Although the proposed method is devised according to the search and rescue scenario, it also applies to other scenarios with different types of objectives as the task significance can be easily obtained. For example, in the case of optimizing a distance objective, the calculation of the significance of a task with regard to a vehicle is only related to the locations of this task and its two neighbor tasks in the ordered task list of the vehicle. Similarly, the marginal significance can only be related to the locations of the new inserting task and its two neighbor tasks in the task list.

D. Convergency and the Algorithm

By assuming that the cost function is diminishing marginal gain (DMG) based on the bundle list, the solutions generated in CBBA can gradually be converged to the one given by centralized sequential greedy algorithm (SGA). In this way, CBBA convergency is guaranteed according to the solution of SGA although it is not optimal even under the DMG assumption. This is different for the distributed task allocation algorithm proposed in this paper. It is a heuristic distributed algorithm essentially working on the iterative optimization principle (as also used by other centralized optimization methods, e.g., genetic algorithms, cellular automata, particle swarm optimization, differential evolution, harmony search, etc.) with each vehicle aiming to decrease the overall cost at each iteration, which hereby underlies its convergency property. In detail, each vehicle receives the significance of all the tasks and based on that, try to reduce the overall cost as much as possible by recursively adding/removing the corresponding tasks into/from the proper position of its task list. The significance value should be kept updated to reflect the current allocation as this is a combinatorial optimization problem and the significance of a task is highly correlated with other tasks in a vehicle's task list. The algorithm is converged when no changes can be made in the significance values and in both task inclusion and removal phases.

To run the proposed method, the significance values of all the tasks can be initially assigned as infinite (this can be imagined by assuming that huge costs are required to perform the tasks allocated to some artificial vehicles) to allow the tasks to be included in the vehicles' task list. The two phases of the proposed method running on vehicle v_i is summarized in Algorithms 1 and 2 and is now detailed as follows.

1) Sending the significance list $\boldsymbol{\gamma}_i$ together with the vehicle list $\boldsymbol{\beta}_i$ stored on vehicle v_i to its communicated vehicle v_j where $g_{i,j}(t) = 1$.
2) Receiving the significance list $\boldsymbol{\gamma}_j$ together with the vehicle list $\boldsymbol{\beta}_j$ from vehicle v_j who has a communication link to vehicle v_i where $g_{j,i}(t) = 1$.

Algorithm 1 Consensus and Task Removal Phase Running on Vehicle v_i

```
1:  Send the significance list $\boldsymbol{\gamma}_i$ and vehicle list $\boldsymbol{\beta}_i$ to vehicle $v_j$, where $g_{i,j}(t) = 1; j = 1, \dots, n; j \neq i$.
2:  Receive the significance list $\boldsymbol{\gamma}_j$ and vehicle list $\boldsymbol{\beta}_j$ from vehicle $v_j$, where $g_{j,i}(t) = 1; j = 1, \dots, n; j \neq i$.
3:  Perform the consensus procedure to update $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$.
4:  Find tasks intended to be removed $\mathbf{d}_i \leftarrow \mathbf{a}_i[\boldsymbol{\beta}_i[\mathbf{a}_i] \neq v_i]$.
5:  while $\max_{k=1}^{|\mathbf{d}_i|}\{\boldsymbol{\gamma}_i^{\diamond}[\mathbf{d}_i] - \boldsymbol{\gamma}_i[\mathbf{d}_i]\} > 0$ and $\mathbf{d}_i \neq \emptyset$ do
6:      Remove the corresponding task given by $t_r^{i)} \leftarrow \arg\max_{k=1}^{|\mathbf{d}_i|}\{\boldsymbol{\gamma}_i^{\diamond}[\mathbf{d}_i] - \boldsymbol{\gamma}_i[\mathbf{d}_i]\}$ from task list $\mathbf{a}_i$.
7:      Remove task $t_r^{i)}$ from $\mathbf{d}_i$.
8:      Update time costs $c_{i,\kappa}(\mathbf{a}_i)$ for the tasks followed after $t_r^{i)}$ in $\mathbf{a}_i$.
9:      Update significance $\boldsymbol{\gamma}_i^{\diamond}$ for the rest of tasks in $\mathbf{d}_i$.
10: end while
11: Update vehicle list $\boldsymbol{\beta}_i(\mathbf{d}_i) \leftarrow v_i$.
```

Algorithm 2 Task Inclusion Phase Running on Vehicle v_i

```
1:  while $|\mathbf{a}_i| \leq m_i$ do
2:      Compute the marginal significance list $\boldsymbol{\gamma}_i^{\star} \leftarrow [w_{1,i}^{\star}, \dots, w_{m,i}^{\star}]^{\mathrm{T}}$.
3:      if $\max_{k=1}^{m}\{\gamma_{i,k} - \gamma_{i,k}^{\star}\} > 0$ then
4:          $t_\ell^{i)} \leftarrow \arg\max_{k=1}^{m}\{\gamma_{i,k} - \gamma_{i,k}^{\star}\}$.
5:          $l_\ell^{i)} \leftarrow \arg w_{\ell,i}^{\star}(\mathbf{a}_i \oplus t_\ell^{i)})$.
6:          Add task $t_\ell^{i)}$ into $\mathbf{a}_i$ at position $l_\ell^{i)}$.
7:          Update significance $\gamma_{i,\ell} \leftarrow \gamma_{i,\ell}^{\star}$.
8:          Update vehicle list's entry $\beta_{i,\ell} \leftarrow v_i$.
9:          Update time cost $c_{i,\kappa}(\mathbf{a}_i)$ for the tasks followed after $t_\ell^{i)}$ in $\mathbf{a}_i$.
10:     else
11:         break.
12:     end if
13: end while
14: Update significance list $\boldsymbol{\gamma}_i \leftarrow [\gamma_{i,1}, \dots, \gamma_{i,m}]^{\mathrm{T}}$.
```

3) Carrying out the consensus procedure to update the significance list $\boldsymbol{\gamma}_i$ and the vehicle list $\boldsymbol{\beta}_i$ according to all the received lists $\boldsymbol{\gamma}_j$ and $\boldsymbol{\beta}_j$ where $j = 1, \dots, n$ and $g_{j,i}(t) = 1$.
4) Checking the current task list $\mathbf{a}_i$ with the vehicle list $\boldsymbol{\beta}_i$ to see if any of the included tasks should be removed according to the significance vector $\boldsymbol{\gamma}_i^{\diamond}$ corresponded to the current task list and the significance list $\boldsymbol{\gamma}_i$ (after consensus) obtained at the last step. The tasks are removed from $\mathbf{a}_i$ and $\mathbf{d}_i$ one after another by each time satisfying the criterion (12) and leading to the largest objective reduction $\max_{k=1}^{|\mathbf{d}_i|}\{\boldsymbol{\gamma}_i^{\diamond}[\mathbf{d}_i] - \boldsymbol{\gamma}_i[\mathbf{d}_i]\}$, given that $\mathbf{d}_i$ is not an empty set. The time cost $c_{i,\kappa}(\mathbf{a}_i)$ and the significance vector $\boldsymbol{\gamma}_i^{\diamond}$ for the corresponding tasks are continuously updated during the task removal process. The remaining tasks in $\mathbf{d}_i$ are retained in the task list $\mathbf{a}_i$ and the vehicle list is revised as $\boldsymbol{\beta}_i(\mathbf{d}_i) = v_i$.
5) Computing the marginal significance list for all the tasks $\boldsymbol{\gamma}_i^{\star} = [w_{1,i}^{\star}, \dots, w_{m,i}^{\star}]^{\mathrm{T}}$ according to (9). The tasks are then continuously added into the task list $\mathbf{a}_i$ one after another by each time satisfying the criteria (10) and (2) and leading to the largest objective reduction with the new task $t_\ell^{i)} = \arg\max_{k=1}^{m}\{\gamma_{i,k} - \gamma_{i,k}^{\star}\}$ at position $l_\ell^{i)} = \arg w_{\ell,i}^{\star}(\mathbf{a}_i \oplus t_\ell^{i)})$ in the task list. During this task inclusion phase, the significance of the task $t_\ell^{i)}$ and the ℓth element of the vehicle list $\boldsymbol{\beta}_i$ are set as $\gamma_{i,\ell} = \gamma_{i,\ell}^{\star}$ and $\beta_{i,\ell} = v_i$, respectively. The time cost $c_{i,\kappa}(\mathbf{a}_i)$ for other tasks in the task list is also continuously updated. Finally, the significance list $\boldsymbol{\gamma}_i$ on the vehicle needs to be updated according to (11) before this phase is completed.
6) Steps 1) and 5) are repeated until no actions have been performed in steps 3)–5) for a period of time.

E. Computational Complexity

Given the cardinality $|\mathbf{a}_i|$ of the task list $\mathbf{a}_i$ and the capability m_i of vehicle v_i, the computational complexity of the distributed method running on every vehicle is now analyzed according to the aforementioned two phases described in Sections IV-B and IV-C. First of all, there are relatively few computations required for the consensus of the significance and vehicle lists on each vehicle as most operations are simple rule-based logical judgements and assignments. Also, the simple computations involved in knowing what tasks are intended to be removed from the current task list and in determining which tasks are eventually removed afterwards at each iteration can be omitted. Therefore, the major computational complexity involved in the consensus and task removal phase arises from the update of the significance value of the intended removal tasks and the update of the time cost for the remaining tasks in the task list. Assuming a total of ϑ_x tasks are found potentially removed from the task list, this requires a computational complexity of $|\mathbf{a}_i|\vartheta_x\sigma - \vartheta_x(\vartheta_x + 1)\sigma/2 + (|\mathbf{a}_i| - 1)\sigma$ maximally, where σ denotes the complexity of computing the time cost of a task. The complexity is based on that the time costs of all the tasks after a specific task in the task list need to be updated when computing the significance of this task or when removing it from the task list. Therefore, the actual computational burden can be significantly smaller as practically only the time costs of several immediately followed tasks in the task list are changed.

Regarding the task inclusion phase, similarly, there is a maximum computational complexity of $(|\mathbf{a}_i|+1)(|\mathbf{a}_i|+2)\vartheta_y\sigma/2 + |\mathbf{a}_i|\sigma$ including the computation of task marginal significance and the update of task time costs in order to add a new task into the current task list $\mathbf{a}_i$, where ϑ_y denotes the number of tasks that are not yet involved in the task list and also meet the constraint (4). It is now obvious that the major computation of the proposed method comes from the task inclusion phase. As a result, the computational complexity for the proposed distributed task allocation method is polynomial and dominated by the task inclusion phase and it is $O((m_i - |\mathbf{a}_i|)|\mathbf{a}_i|^2\vartheta_y\sigma)$ (considering a maximum number of $m_i - |\mathbf{a}_i|$ tasks that can be added into the task list) at each iteration of the algorithm running on each vehicle. It should be pointed out that the proposed approach will find good solutions efficiently rather than globally optimal solution by an exact algorithm which is NP-hard for the multivehicle task allocation problems.

Given the distributed computing nature of the problem, it is worth mentioning that the variables $|\mathbf{a}_i|$, ϑ_x, and ϑ_y vary at each iteration of the algorithm but are also bounded by the specific problem of interest and the associated constraints. For example, the various communication limits placed on each vehicle will then affect the consensus results, which in turn affect the value of ϑ_x, $|\mathbf{a}_i|$, and ϑ_y but all are bounded by the capability of the vehicle and the number of homogeneous tasks being able to be performed by the vehicle. Noting that if the distance objective is considered, the main computational complexity for removing a task reduces to $\vartheta_x\sigma + \sigma$ as the task significance then is only related to its two neighbor tasks, whereas σ denotes the complexity of computing the travel cost for performing a task. Similarly, adding a new task mainly needs $2(|\mathbf{a}_i|+1)\vartheta_y\sigma + \sigma$, resulting in an overall computational complexity of $O((m_i - |\mathbf{a}_i|)|\mathbf{a}_i|\vartheta_y\sigma)$ at each iteration of the algorithm running on each vehicle.

V. Simulation Results

This section presents the simulation results for the proposed distributed task allocation method to demonstrate its effectiveness and outstanding performance. The scenario description is introduced first in Section V-A, followed by the various detailed simulation results in Section V-B obtained by the proposed method from scenarios with different random initial settings and different number of vehicles and tasks. The obtained results are also compared with results from CBBA given the same objective of minimizing the average time cost for rescuing a survivor.

A. Example Scenario and System Configuration

We now examine the proposed method for a multivehicle multitask allocation problem based on the search and rescue scenario, where the distributed framework and heterogeneous vehicles are considered. As mentioned before, suppose that there are a number of survivors discovered after a disaster happened in an area, who need to be rescued by a team of heterogeneous rescue vehicles. The goal is to provide emergent rescue support to these survivors as quickly as possible by using the heterogeneous vehicles since the health condition of the survivors is getting worse over time. For example, there is a total of 12 survivors found at different locations in a local area (may be dynamically discovered in reality); six of these survivors are injured and need medicine; the other six need food. A heterogeneous rescue team of six members with two types of specialized vehicles is able to cruise over the area, where three of them can provide medicine to the survivors and the other three can provide food. The mission is, therefore, to send out these specialized rescue vehicles to achieve the goal with limited local communications available on the vehicles.

In this simulation, the vehicles and survivors were randomly generated in a $10\,000 \times 10\,000 \times 1000$ m area. At the beginning, the vehicles were uniformly parked in the whole area of the corresponding 2-D ground plane, while the survivors' location was uniformly distributed in the whole area of the 3-D spatial plane. The speed of each vehicle was assumed constant and set to 30 m/s for the vehicles delivering medicine and 50 m/s for the vehicles delivering food. All the survivors appeared in the time range [0, 2000 s] and the particular deadline for each survivor to get the corresponding support was randomly placed in this time range. Given the heterogeneous vehicles, the task execution duration for the vehicles providing medicine and for the vehicles sending food were 300 and 350 s, respectively.

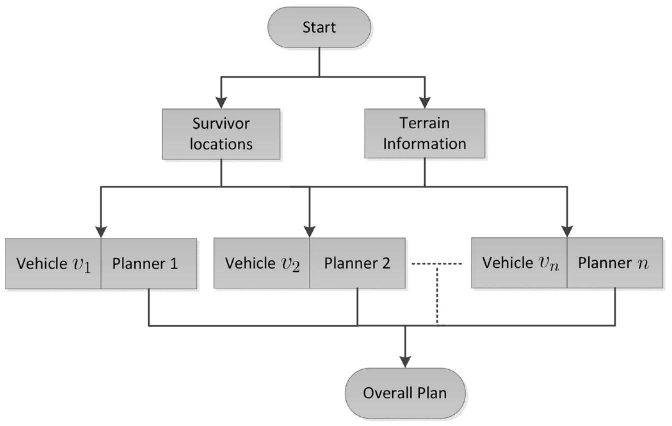


Fig. 5. System working architecture for the distributed task allocation algorithm.

A total of ten simulations were conducted in the following section for each method for various conditions according to the different scenarios randomly generated in the design region. To get all these survivors successfully rescued, the constraints listed in (2)–(5) ($m_i = m$) are required to be met to provide conflict-free solutions for the underlying problem. Specifically, except the constraint $\bigcup_{i=1}^{n} \mathbf{a}_i = \boldsymbol{T}$, the rest constraints are generally satisfied to provide valid allocation of survivors to each vehicle. For the constraint $\bigcup_{i=1}^{n} \mathbf{a}_i = \boldsymbol{T}$ indicating that all the survivors are rescued, it may not always be the case due to the random generation of scenarios (where theoretically it is not possible to rescue all survivors) and also the local optima occurred in an algorithm, since essentially there is always a deadline imposed on rescuing a survivor. This could, therefore, result in some survivors failed to be rescued. The distributed task allocation algorithm runs on each vehicle concurrently with the system working architecture as shown in Fig. 5. The survivor locations obtained from the search team together with the terrain information reflecting the path connectivity are fed as the input to the algorithm running on each vehicle. Each planner equipped with a copy of the proposed method is placed on the corresponding vehicle and communicates with other planners where communications exist between them to negotiate an overall plan for the whole system.

Given the nature of the distributed task allocation, four basic communication topologies [13] were tested for the proposed method. Among which, the row communication shown in Fig. 6(a) connects the vehicle one after another which has a very low degree of communication between the vehicles, while Fig. 6(b) generally depicts a row communication but with one of the vehicle being able to correspond with several other vehicles. Fig. 6(c) describes a circular communication

TABLE I
RESULTS OF THE CBBA AND PROPOSED METHODS FOR PROVIDING EMERGENCY SUPPORT TO THE SURVIVORS

Scenarios			# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
(6, 12)	CBBA	# Failed	2	1	1	2	2	1	2	2	1	0
		Ave. time	1/1	1/0	0/1	1/1	1/1	0/1	1/1	1/1	1/0	307.6106
	Proposed	# Failed	2	0	0	1	1	0	1	1	1	0
		Ave. time	1/1	285.9589	270.8845	1/0	1/0	297.9673	1/0	0/1	1/0	303.0081
(8, 16)	CBBA	# Failed	3	3	0	3	3	2	2	4	4	1
		Ave. time	1/2	2/1	274.0099	2/1	1/2	1/1	2/0	3/1	2/2	0/1
	Proposed	# Failed	2	0	0	1	1	0	1	1	1	0
		Ave. time	0/2	328.7850	270.3316	1/0	1/0	266.5068	1/0	1/0	1/0	288.8882
(10, 20)	CBBA	# Failed	4	3	2	2	4	2	2	2	3	1
		Ave. time	1/3	2/1	1/1	1/1	3/1	1/1	1/1	1/1	1/2	0/1
	Proposed	# Failed	4	0	1	1	1	0	0	2	2	0
		Ave. time	1/3	312.0478	1/0	1/0	1/0	261.7693	271.7408	1/1	1/1	276.1955
(12, 24)	CBBA	# Failed	2	2	2	3	6	1	1	4	3	1
		Ave. time	1/1	1/1	1/1	2/1	3/3	1/0	1/0	1/3	0/3	0/1
	Proposed	# Failed	2	0	0	1	1	0	0	2	0	0
		Ave. time	1/1	268.9474	274.5524	1/0	0/1	255.6307	248.0726	1/1	265.4530	272.9357
(14, 28)	CBBA	# Failed	4	1	2	3	6	3	2	4	5	3
		Ave. time	2/2	0/1	1/1	2/1	2/4	2/1	2/0	2/2	2/3	2/1
	Proposed	# Failed	2	0	0	1	2	0	0	2	2	0
		Ave. time	2/0	264.7861	269.4521	1/0	1/1	263.4354	235.2826	1/1	1/1	272.0922
(16, 32)	CBBA	# Failed	6	3	3	3	5	3	4	5	7	2
		Ave. time	3/3	2/1	2/1	2/1	2/3	1/2	1/3	3/2	4/3	2/0
	Proposed	# Failed	3	1	0	1	2	0	0	2	2	0
		Ave. time	3/0	1/0	265.4293	1/0	1/1	241.5782	262.4904	1/1	1/1	238.49

* Here, 'Ave. time' represents the average time cost (seconds) obtained from the corresponding method if all the survivors are rescued; otherwise, it is used to denote the number of failed rescued survivors in each type respectively. Similarly afterwards in Tables II and III.

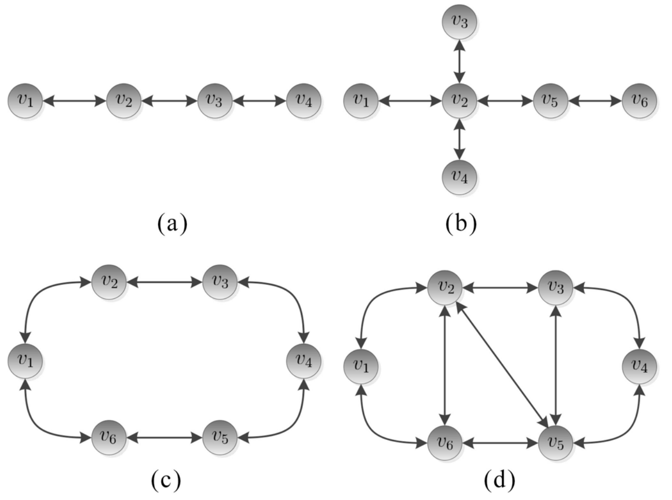


Fig. 6. Four basic communication topologies used between the vehicles. (a) Row communication. (b) Star row communication. (c) Circular communication. (d) Mesh communication.

where each vehicle can communicate with two other vehicles. A mesh communication is represented in Fig. 6(d), where a vehicle is able to correspond with several vehicles within its communication range. All the simulations were conducted on an Intel Core 2 Duo Processor E8135 2.40 GHz, with programs compiled by MATLAB under Windows 7 operating system.

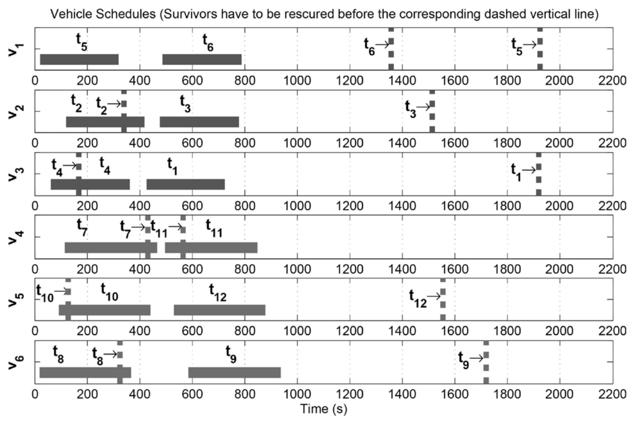
B. Simulation Results

The proposed method was first examined on the aforementioned scenarios with a number of different vehicles and survivors involved, where the number of survivors being doubled the number of vehicles was adopted here. For each specific number of vehicles and survivors, a total of ten scenarios established on random locations of the vehicles and survivors and deadlines for rescuing the survivors were also evaluated. The low communication topology, i.e., row communication, was first adopted for the algorithm to conduct the simulation. Given the distributed computing nature and that the two phases involved in our method are iterative and only the final results are guaranteed conflict-free, the results presented in this section are thus the final output of the both phases. The difference of the two phases in our method lies in that the former phase is used to add more tasks into a vehicle's task list, while the latter is to reach consensus on the significance value of tasks and to remove tasks from the vehicles' task list. As these scenarios were randomly generated given this heterogeneous problem together with a limited number of vehicles being used, theoretically it might not be possible to find an overall solution in order to get all the survivors rescued in time. This is practically reasonable since there are normally limited and fixed resources for a sudden disaster just happened whilst the survivors' health condition in the meantime is getting worse over time. As a result, two kinds of scenarios, i.e., one with all survivors rescued and the other with some survivors missed, can be seen from the conducted simulations.

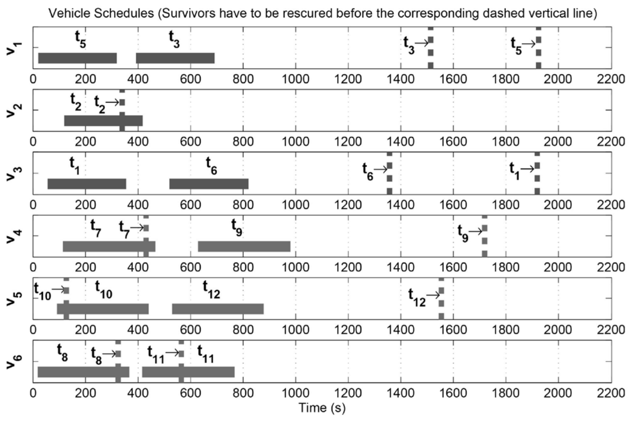
As shown in Table I, the number of failed rescued survivors is presented in the first row for each method. Then, the average time cost for rescuing a survivor is shown in the second row if all the survivors were successfully rescued; otherwise, the number of failed rescued survivors in each type (needing to be provided medicine or food) is, respectively, listed there. Generally speaking, our method found more scenarios with all the survivors rescued across different numbers of vehicles and survivors. For the first two cases, i.e., (6, 12) and (8, 16), in which the values in the brackets denote the number of vehicles and survivors, respectively, the CBBA method

TABLE II
RESULTS OF THE CBBA AND PROPOSED METHODS ACROSS DIFFERENT COMMUNICATION TOPOLOGIES

Scenarios			# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
Star	CBBA	# Failed	2	1	1	2	2	1	2	2	1	0
		Ave. time	1/1	1/0	0/1	1/1	1/1	0/1	1/1	1/1	1/0	307.6106
	Proposed	# Failed	2	0	0	1	1	0	1	1	1	0
		Ave. time	1/1	285.9589	270.8845	1/0	1/0	297.9673	1/0	0/1	1/0	303.0081
Circular	CBBA	# Failed	2	1	1	2	2	1	2	2	1	0
		Ave. time	1/1	1/0	0/1	1/1	1/1	0/1	1/1	1/1	1/0	307.6106
	Proposed	# Failed	2	0	0	1	1	0	1	1	1	0
		Ave. time	1/1	285.9589	270.8845	1/0	1/0	297.9673	1/0	0/1	1/0	303.0081
Mesh	CBBA	# Failed	2	1	1	2	2	1	2	2	1	0
		Ave. time	1/1	1/0	0/1	1/1	1/1	0/1	1/1	1/1	1/0	307.6106
	Proposed	# Failed	2	0	0	1	1	0	1	1	1	0
		Ave. time	1/1	282.8492	270.8845	1/0	1/0	296.3573	1/0	0/1	1/0	301.9922



(a)



(b)

Fig. 7. Every vehicle plan of both our and CBBA methods where all the survivors were rescued by our method. (The horizontal line denotes the time period for rescuing a survivor and the vertical line is the deadline for starting to rescue a survivor. Similarly afterwards in Figs. 8 and 9.) (a) Results from our method (all survivors rescued). (b) Results from CBBA (survivor t_4 with deadline at 168.4163 s missed).

was only able to find one scenario with all the survivors rescued (scenarios 10 and 3, respectively). The situation became even more worse for the last four cases when larger numbers of vehicles and survivors involved, i.e., (10, 20), (12, 24), (14, 28), and (16, 32), where none of the scenarios has been found with all the survivors rescued. Moreover, regarding the only two scenarios in which all the survivors rescued by using CBBA, more average rescuing times were consumed compared to that from ours. Apart from some scenarios where theoretically only part of the survivors can be rescued, our method was still able to reach feasible solutions for all the survivors in other scenarios to which the CBBA failed. On the other hand, for the scenarios where both methods did not provide emergency support to all the survivors, the advantage of our method can also be observed in terms of missing less number of survivors. In conclusion, it has been demonstrated that the proposed method achieved outstanding performance in providing emergency support to the survivors.

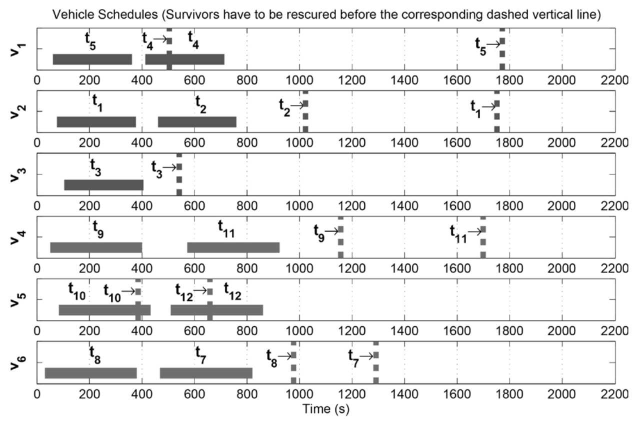
Specifically, one scenario with all the survivors rescued and another part of the survivors rescued are, respectively, shown in Figs. 7 and 8, where the time period between the actual start and completion times for rescuing a survivor is plotted as the horizontal line and the deadline for starting to rescue a survivor is denoted by the vertical line. Fig. 7(a) gives each vehicle's plan for rescuing the survivors assigned to it from our method, resulting in all the 12 survivors successfully rescued. However, for the CBBA results shown in Fig. 7(b), the last six survivors needing food supply were successfully rescued, while the survivor t_4 with a deadline at 68.4163 s had been failed in getting medicine. Regarding Fig. 8, both methods had missed to rescue survivor t_6, which is caused by its too early deadline (69.5223 s). According to the locations of the survivor and the three vehicles capable of providing medicine to t_6, the earliest arrival times of these right vehicles to reach the location of the survivor are 180.8837, 226.1896, and 213.1589 s, respectively, which by then would all be too late. Therefore, theoretically it is unable to get the survivor t_6 rescued given the current resources. Besides t_6, our method had rescued all the other survivors.

Since the proposed algorithm is distributed and running on each vehicle concurrently, it is interesting to see the effectiveness of the algorithm by running the simulation under different communication topologies presented in Fig. 6. As described previously, the same ten scenarios were employed to proceed the simulation given 6 vehicles and 12 survivors. Apart from the results obtained from row communication which were already shown in Table I, the results from star, circular, and mesh communications are hereby, respectively, listed in Table II. It is also confirmed that the proposed method was generally able to produce promising results for all the different communication topologies.

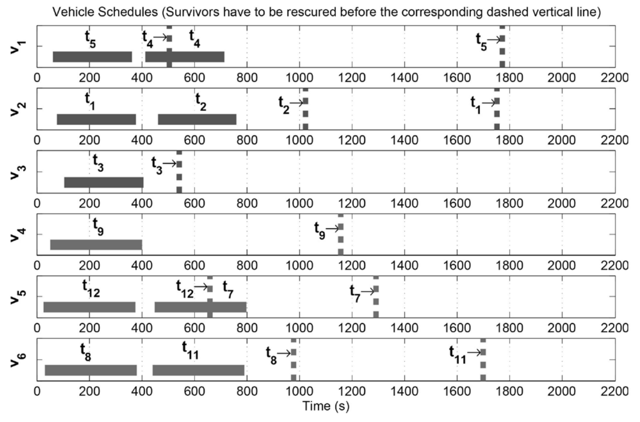
Further simulations were also conducted to investigate the performance of the proposed method under dynamic and

TABLE III
RESULTS OF THE PROPOSED METHOD FROM ADDING TWO NEW DIFFERENT TYPE SURVIVORS AFTER THE COMPLETION OF ALLOCATIONS

Scenarios		# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
(6, 12)	# Failed	2	0	0	1	1	0	1	1	1	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	1/1	370.6796	352.8350	1/0	1/0	375.8090	1/0	0/1	1/0	363.0227
(8, 16)	# Failed	2	0	0	1	1	0	1	1	1	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	0/2	386.1152	327.7432	1/0	1/0	332.8359	1/0	1/0	1/0	339.0993
(10, 20)	# Failed	4	0	1	1	1	0	0	2	2	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	1/3	353.5860	1/0	1/0	1/0	309.3205	310.4857	1/1	1/1	323.2525
(12, 24)	# Failed	2	0	0	1	1	0	0	2	0	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	1/1	304.9477	312.3289	1/0	0/1	300.0281	286.8617	1/1	306.0724	305.8445
(14, 28)	# Failed	2	0	0	1	2	0	0	2	2	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	2/0	299.7221	298.9202	1/0	1/1	297.3298	267.0040	1/1	1/1	300.2812
(16, 32)	# Failed	3	1	0	1	2	0	0	2	2	0
	# Newly failed	0	0	0	0	0	0	0	0	0	0
	Ave. time	3/0	1/0	294.0000	1/0	1/1	274.4337	282.8865	1/1	1/1	267.8115

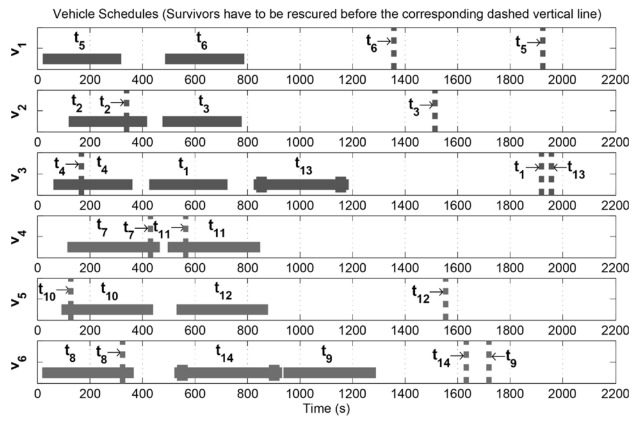


(a)

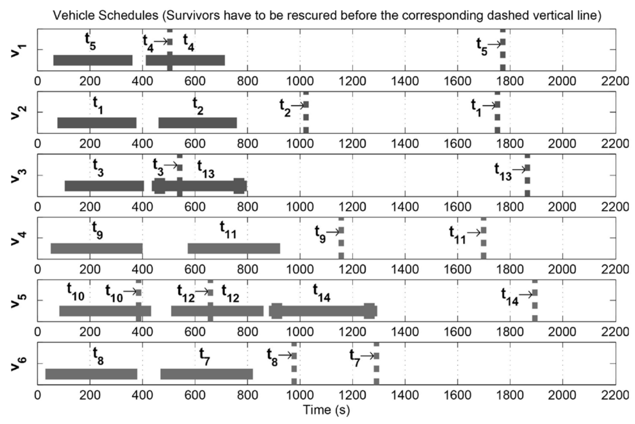


(b)

Fig. 8. Every vehicle plan of both our and CBBA methods where part of the survivors were missed by both CBBA and our methods. (a) Results from our method (survivor t_6 with deadline at 69.5223 s missed). (b) Results from CBBA (survivor t_6 with deadline at 69.5223 s missed and survivor t_{10} with deadline at 385.5396 s missed).



(a)



(b)

Fig. 9. Obtained vehicle plan by adding two new survivors to the completed allocations corresponding to Figs. 7 and 8 in the proposed method. (a) All original survivors rescued, corresponding to Fig. 7: the new survivor t_{13} is allocated to v_3 and the new survivor t_{14} is allocated to v_6. (b) Some original survivors missed, corresponding to Fig. 8: the new survivor t_{13} is allocated to v_3 and the new survivor t_{14} is allocated to v_5.

unexpected situations when new survivors appeared after the allocation of the previous survivors had been finished. Here, one survivor who needs medicine support together with another needing food supply was randomly generated with the deadline in the time range of [1500, 2000 s]. The number of missed survivors from all the survivors and from the newly added survivors, together with the average rescuing time or the number of missed survivors from each type is listed in Table III for different scenarios and different numbers of vehicles and tasks. It can be seen that both newly added two type survivors had been successfully allocated to the corresponding vehicles based on the previous allocation as shown in the second row of each bundle of vehicles and survivors. The proposed method was capable of dealing with the dynamic changes in the appearance of new tasks. Particularly, for the number of 6 vehicles and 12 survivors, as compared to the previous allocations as shown in Figs. 7 and 8, the new allocations after adding survivors t_{13} and t_{14} are given in Fig. 9. The two survivors were appeared in the task list of the corresponding heterogeneous vehicles [v_3 and v_6 in Fig. 9(a), v_3 and v_5 in Fig. 9(b)].

VI. Conclusion

This paper has proposed a novel distributed method to solve the multivehicle and multitask allocation problems. Given the search and rescue scenario, the objective and its mathematical formulation was first extracted from the general scenario description. Considering the requirement of distributed computing for solving this NP-hard problem, a heuristic and effective distributed task allocation method was then proposed aiming directly at optimizing the mathematical objective defined for the problem of interest. A new concept called significance was devised to measure the contribution of a task to the local cost generated by each vehicle. The overall cost of the objective can thus be decreased by switching tasks amongst different vehicles to satisfy certain criteria. The proposed method iterates between a task inclusion phase, and a consensus and task removal phase, the first phase being used to include tasks into a vehicle's task list while the latter being responsible for the consensus on significance values of tasks for each vehicle and for removing the tasks that have been taken over by other vehicles. Finally, a series of simulations were conducted under various conditions for the proposed method according to the different scenarios randomly generated in a local design region. The results have demonstrated the effectiveness of the proposed method and its outstanding performance in comparison with the CBBA.

Acknowledgment

The authors would like to thank the editors and the reviewers for their most constructive comments to improve the quality of this paper.

References

[1] H. I. Son *et al.*, "Human-centered design and evaluation of haptic cueing for teleoperation of multiple mobile robots," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 597–609, Apr. 2013.

[2] H. Li and F. Nashashibi, "Cooperative multi-vehicle localization using split covariance intersection filter," *IEEE Intell. Transp. Syst. Mag.*, vol. 5, no. 2, pp. 33–44, Apr. 2013.

[3] J. Qin, W.-X. Zheng, and H. Gao, "Coordination of multiple agents with double-integrator dynamics under generalized interaction topologies," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 44–57, Feb. 2012.

[4] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[5] D. Zhu, H. Huang, and S. X. Yang, "Dynamic task assignment and path planning of multi-AUV system based on an improved self-organizing map and velocity synthesis method in three-dimensional underwater workspace," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 504–514, Apr. 2013.

[6] J. Su and W. Xie, "Motion planning and coordination for robot systems based on representation space," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 248–259, Feb. 2011.

[7] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 1. Sendai, Japan, Sep. 2004, pp. 698–705.

[8] J. Hawley and Z. Butler, "Hierarchical distributed task allocation for multi-robot exploration," in *Distributed Autonomous Robotic Systems* (Springer Tracts in Advanced Robotics), vol. 83. Berlin, Germany: Springer, 2013, pp. 445–458.

[9] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multi-robot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.

[10] S. Sariel and T. Balch, "Efficient bids on task allocation for multi-robot exploration," in *Proc. 19th Int. Florida Artif. Intell. Res. Soc. Conf. (FLAIRS)*, Melbourne Beach, FL, USA, 2006, pp. 116–121.

[11] D. Turra, L. Pollini, and M. Innocenti, "Fast unmanned vehicles task allocation with moving targets," in *Proc. 43rd IEEE Conf. Decis. Control (CDC)*, vol. 4. Nassau, Bahamas, 2004, pp. 4280–4285.

[12] J. Chen and D. Sun, "Coalition-based approach to task allocation of multiple robots with resource constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 3, pp. 516–528, Jul. 2012.

[13] K. Zhang, E. G. Collins, Jr., and D. Shi, "Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 2, pp. 21:1–21:22, Jul. 2012.

[14] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems," in *New Trends in Systems Theory* (Progress in Systems and Control Theory), vol. 7, Boston, MA, USA: Birkhäuser, 1991, pp. 105–112.

[15] M. Berhault *et al.*, "Robot exploration with combinatorial auctions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 2. Las Vegas, NV, USA, 2003, pp. 1957–1962.

[16] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 129–141, Jul./Dec. 2013.

[17] S. Koenig *et al.*, "The power of sequential single-item auctions for agent coordination," in *Proc. Nat. Conf. Artif. Intell.*, Palo Alto, CA, USA, 2006, pp. 1625–1629.

[18] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," *Robot. Auton. Syst.*, vol. 58, no. 7, pp. 900–909, Jul. 2010.

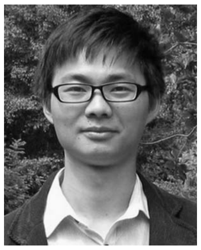
[19] H.-L. Choi, L. Brunet, and J. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[20] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A fast distributed auction and consensus process using parallel task allocation and execution," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, San Francisco, CA, USA, 2011, pp. 4716–4721.

[21] H.-L. Choi, A. K. Whitten, and J. P. How, "Decentralized task allocation for heterogeneous teams with cooperation constraints," in *Proc. Amer. Control Conf. (ACC)*, Baltimore, MD, USA, Jul. 2010, pp. 3057–3062.

[22] L. B. Johnson, S. S. Ponda, H.-L. Choi, and J. P. How, "Asynchronous decentralized task allocation for dynamic environments," in *Proc. AIAA Infotech Aerosp. Conf.*, St. Louis, MO, USA, Mar. 2011, pp. 1–12.

[23] C. Tovey, M. G. Lagoudakis, S. Jain, and S. Koenig, "The generation of bidding rules for auction-based robot coordination," in *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, L. E. Parker, F. E. Schneider, and A. C. Schultz, Eds. Dordrecht, The Netherlands: Springer, 2005, pp. 3–14.



Wanqing Zhao (M'13) received the B.Eng. degree in automation from Anhui Polytechnic University, Wuhu, China, in 2006, the M.Eng. degree in control theory and control engineering from Shanghai University, Shanghai, China, in 2009, and the Ph.D. degree from Intelligent Systems and Control Group, Queen's University Belfast, Belfast, U.K., in 2012.

He was a Research Associate with the Department of Computer Science, Loughborough University, Loughborough, U.K. He is currently a Research Fellow with the School of Engineering, Cardiff University, Cardiff, U.K. His current research interests include system identification, fuzzy regression, neural networks, machine learning, heuristic optimization methods, autonomous systems, and water resource management.

Qinggang Meng (M'06) received the B.Sc. and M.Sc. degrees in electronic engineering from Tianjin University, Tianjin, China, and the Ph.D. degree in computer science from Aberystwyth University, Aberystwyth, U.K.

He is currently a Senior Lecturer with the Department of Computer Science, Loughborough University, Loughborough, U.K. His current research interests include biologically and psychologically inspired learning algorithms and developmental robotics, service robotics, robot learning and adaptation, multi-UAV cooperation, driver's distraction detection, human motion analysis and activity recognition, activity pattern detection, pattern recognition, artificial intelligence and computer vision.

Dr. Meng is a fellow of the Higher Education Academy of the U.K.

Paul W. H. Chung received the B.Sc. degree in computing science from Imperial College London, London, U.K., in 1981, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 1986.

From 1984 to 1991, he was with the Artificial Intelligence Applications Institute, University of Edinburgh. He joined Loughborough University, Loughborough, U.K., in 1991, where he has been a Professor of Computer Science since 1999, the Head of the Department of Computer Science from 2004 to 2008, and the Dean of the School of Science from 2011 to 2014. He was a Visiting Professor with the Beijing University of Posts and Telecommunications, Beijing, China, and the Institute Technology of Brunei, Gadong, Brunei. He was an Invitation Fellow with Okayama University, Okayama, Japan. His current research interests include applying advanced computing techniques to solve novel complex problems. He has successfully supervised 30 doctorate students and published over 200 papers.