Distributed Matching-By-Clone Hungarian-Based Algorithm for Task Allocation of Multiagent Systems

Arezoo Samiei, *Student Member, IEEE*, and Liang Sun, *Member, IEEE*

***Abstract*—In this article, we present a novel approach, namely distributed matching-by-clone hungarian-based algorithm (DMCHBA), to multiagent task-allocation problems, in which the number of agents is smaller than the number of tasks. The proposed DMCHBA assumes that agents employ an implicit coordination mechanism and consists of two iterative phases, i.e., the communication phase and the assignment phase. In the communication phase, agents communicate with their connected neighbors and exchange their local knowledge base until they converge on the global knowledge base. In the assignment phase, each agent builds a squared cost matrix by cloning agents and adding pseudotasks when necessary, and applying the Hungarian method for task allocation. A local planning algorithm is then applied to identify the order of task execution for an agent. The proposed DMCHBA is proven to produce conflict-free assignments among agents in finite time. We compare the performance of DMCHBA with the consensus-based bundle algorithm, the distributed recursive Hungarian-based algorithms, and the cluster-based Hungarian algorithm (CBHA) in Monte-Carlo simulations with different numbers of agents and tasks. The numerical results reveal the superior convergence and optimality of DMCHBA over all other selected algorithms.**

***Index Terms*—Autonomous robots, autonomous systems, distributed task allocation, multiagent systems, uncrewed systems.**

I. Introduction

MULTIAGENT systems, e.g., cooperative robots, have demonstrated their capabilities in critical real-world applications, such as emergency response [1], [2], [3], data collection [4], [5], and mobile target search and tracking [6]. To accomplish those complicated missions, different tasks are usually generated for multiagent systems. Therefore, an effective and efficient task-allocation strategy is critical to guarantee mission success. Task-allocation techniques have been developed for different problems, such as resource allocation/scheduling [7], workload partition in distributed systems [8], workforce scheduling and routing problems [9], target tracking by networked robots [10], multirobot path planning [11], [12], [13], and multiagent consensus [3], [14]. In this article, we consider the problem of allocating independent tasks among distributed agents.

One way of categorizing task-allocation approaches depends on where the decision is made, resulting in centralized and decentralized methods. In centralized methods, an optimal result is determined by a central agent who carries the information of all agents and tasks and performs task assignments for all agents [15]. In distributed methods, agents cooperatively work together to achieve a global goal by optimizing their local objective via reciprocity with their neighbors [16]. Another way of categorizing task-allocation methods depends on the number of agents and tasks. Single-task allocation refers to scenarios where the number of tasks is equal to the number of agents, and each agent is responsible for a single task. Multitask allocation refers to scenarios, where the number of tasks exceeds the number of available agents, necessitating that some agents perform multiple tasks. In this article, we consider the distributed multitask allocation problem for multiagent systems, which is an NP-hard problem that has been investigated in coalition formation, task scheduling, and routing [17], [18], [19].

Two popular centralized task-allocation methods are the Auction and Hungarian algorithms, both of which produce an optimal solution in finite time. On the one hand, the Auction algorithm relies on a central auctioneer who receives bids from agents for a set of tasks. When all bids are received or a prespecified deadline for bidding has passed, the auctioneer will start to determine the winner for each task by adjusting task prices to resolve conflicts if any [19]. The Auction algorithm has been applied to applications, such as multirobot coordination [15], [16] and assignment problems [20], and was extended to relaxation methods [21], [22]. On the other hand, the Hungarian algorithm [23] generates an optimal assignment through a linear-programming process based on a cost matrix. Kuhn [23] presented the Hungarian method as a single-task assignment problem to find a minimal-cost assignment [24], [25]. The Hungarian algorithm has been applied to applications, such as target detection [26], moving-target assignment [27], online multiobject tracking [28], fault-tolerant for a cooperative unmanned vehicle team Kamel et al. [29], and formation generation [30]. Recent research also has been done based on the Hungarian algorithm to address multitask allocation [18], [31], [32], [33].

While centralized task-allocation algorithms, such as the Auction and Hungarian algorithms, generate global optimal solutions, their limitations in robustness and scalability call for

Manuscript received 9 February 2023; revised 12 July 2023; accepted 2 November 2023. Date of publication 24 November 2023; date of current version 29 December 2023. This paper was recommended for publication by Associate Editor P. Tokekar and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by funds from New Mexico State University and the NASA New Mexico Space Grant Consortium (NMSGC) under the Research Infrastructure Development program. *(Corresponding author: Arezoo Samiei.)*

Arezoo Samiei is with the Klipsch School of Electrical and Computer Engineering, Las Cruces, NM 88003 USA (e-mail: arezoo@nmsu.edu).

Liang Sun is with the Department of Mechanical and Aerospace Engineering, Las Cruces, NM 88003 USA (e-mail: lsun@nmsu.edu).

Digital Object Identifier 10.1109/TRO.2023.3335656

decentralized/distributed approaches to large-scale and complicated task-allocation problems [18], [34], [35], [36], [37], [38], [39]. For distributed single-task allocation problems, Patel et al. [39] presented a decentralized genetic algorithm (GA) that exploits the parallelism inherent in a GA and continues the search for better solutions while agents complete tasks. Kamel et al. [29] presented a decentralized auction-based method for resource allocations. Choi et al. [38] presented two decentralized auction-based task-allocation algorithms to coordinate multiple autonomous vehicles, namely the consensus-based auction algorithm (CBAA) and the consensus-based bundle algorithm (CBBA). CBAA is a single-task allocation algorithm, while CBBA is a multitask allocation algorithm so that bundles of tasks are generated for task allocation. Due to onboard resource constraints, Lusk et al. [40] trade optimality for computational efficiency by using CBAA for the assignment problem that is formulated as a linear sum assignment problem. Although CBBA has been a benchmark of distributed multitask allocation algorithms, its relatively slow convergence and low optimality have been the motivations for developing new multitask allocation algorithms.

The decentralized type of the Hungarian method has also drawn researchers' attention. Giordani et al. [41] presented a distributed single-task allocation algorithm for multirobot systems, where the robots interactively execute certain steps of the Hungarian algorithm using their local information and implicitly communicate with their neighboring robots for information exchange. This algorithm produces a globally optimal solution in $O(n^3)$ with $O(n^3)$ neighboring communication messages when no shared memory nor a common coordinator is available. The decentralized Hungarian-based algorithm (DHBA), a single-task allocation algorithm developed by Ismail and Sun [18], uses an implicit communication scheme for agents to update their local cost matrices such that every agent can apply the Hungarian algorithm to generate assignments for the entire agent team. The comparison study in [18] showed that DHBA outperforms CBAA on the basis of computational time and optimality. Turpin et al. [13] proposed both a centralized and a decentralized algorithms for the namely concurrent assignment and planning of trajectories (CAPT) problem. The centralized CAPT solution uses the Hungarian algorithm for single-task allocation problems and applies the Hungarian algorithm recursively for multitask allocation problems, which was also explored in our previous work [32]. In the proposed decentralized CAPT algorithm, only the single-task allocation problem was considered and a heuristic local task-swapping mechanism was proposed to resolve local conflicts between two communicated robots. However, the performance of the decentralized CAPT algorithm was not compared with benchmark algorithms, e.g., CBBA [38]. Smith and Bullo [42] and Yu et al. [10] proposed heuristic decentralized algorithms for single-task allocation problems while these proposed algorithms were not compared with benchmark algorithms.

In our previous work [31], we presented the cluster-based Hungarian algorithm (CBHA) for multitask allocation, which uses a clustering algorithm to generate the same number of task groups as the number of agents so that the Hungarian algorithm can be applied. A local planning algorithm is then applied to determine the order of task execution for each individual agent. Using the state-of-the-art local planning and clustering algorithms, CBHA converges faster compared to other multitask allocation methods. In [32], we developed the distributed recursive Hungarian-based algorithm (DRHBA) for multitask allocation, which introduces dummy agents and tasks to obtain a squared cost matrix. Although DRHBA does not rely on any clustering algorithms, the recursive application of the Hungarian algorithm demands a fast-increasing computation time when the task-agent ratio is large. In this article, we consider two variations of DRHBA, namely, DRHBA-I and DRHBA-II. In DRHBA-I, agents maintain their locations throughout the entire task-allocation process, whereas, in DRHBA-II, an agent is assumed to "move" to the position of its previously assigned task in each iteration before proceeding to the next iteration. The comparison-study results in [32] show that DRHBA outperforms CBBA but is outperformed by CBHA on the basis of both the overall cost and the computational time. CBBA suffers a lengthy process of bundle building and the scalability of DRHBA deteriorated rapidly when the task-to-agent ratio increases. CBHA relies on the clustering process, which is not applicable when the "similarity" of tasks cannot be identified [43].

The contributions of this article include 1) a novel algorithm, namely the Distributed Matching-by-Clone Hungarian-Based Algorithm (DMCHBA), for distributed agents to allocate independent tasks with a fast converging speed and an improved overall cost compared to benchmark algorithms, 2) the analysis of the convergence and complexity of the proposed DMCHBA, and 3) comparison studies of DMCHBA versus CBBA, DRHBA, and CBHA. CBBA has been widely selected as a benchmark to compare with new multitask allocation algorithms. DRHBA and CBHA are selected due to their similarity with the proposed DMCHBA. DMCHBA starts with the creation of a squared cost matrix by cloning agents and by adding pseudotasks when needed. Then, the Hungarian algorithm is applied only once for task assignment. A local planning algorithm [e.g., traveling–salesman–problem (TSP) algorithms] is applied for each individual agent to determine the order of task execution. A key difference between DRHBA and DMCHBA is the way of building the square cost matrix such that the Hungarian algorithm can be applied. DRHBA uses dummy agents while DMCHBA uses cloned agents. In DRHBA, the cost matrix is rebuilt after each iteration, and tasks are assigned by recursively applying the Hungarian algorithm, which dominates the total run-time of executing DRHBA. DMCHBA only applies the Hungarian algorithm once and its run-time is much smaller than DRHBA. DMCHBA neither needs a lengthy bundle creation process as CBBA does, nor relies on a clustering algorithm as CBHA does, nor needs to recursively apply the Hungarian algorithm as DRHBA does. The numerical studies show that the proposed DMCHBA outperforms all other selected algorithms on the basis of both computational complexity and overall cost.

The rest of this article is organized as follows. In Section II, we present the problem statement and introduce key concepts and techniques for the presentation of DMCHBA. In Section III, the proposed DMCHBA is presented, followed by the discussions

on conflict-free resolution in Section IV and convergence in Section V. The numerical and comparison-study results are presented in Section VI. Finally, Section VII concludes this article.

II. Preliminaries

A. Task Allocation Problems

The task-allocation problem, we consider in this article is to find a conflict-free assignment of N_t independent tasks to N_a distributed agents, where $N_a \leq N_t$, such that a predefined cost function is minimized. We define conflict-free as a situation where no task is assigned to more than one agent in the final assignment. The assignment is said to be complete once all tasks have been assigned. The global cost is assumed to be the sum of the local cost. This assumption has been commonly used in literature for distributed task allocation [13], [21], [38], [44] such that the problem can be formulated as a linear sum assignment problem and solved using high-performance algorithms, such as the CBAA and the Hungarian method. This assumption ignores the interaction among agents and tasks and the state change of agents and tasks, both of which would result in the difference between the sum of local costs and the global cost. In this work, it is assumed that the tasks are independent and agents do not impact each other during the execution of tasks. The task assignment problem can be formulated as a linear sum assignment problem

$$\min_{x_{ij}} \sum_{i=1}^{N_a} \left\{ \sum_{j=1}^{N_t} c\left(p_i\left(x_{ij}\right)\right) \right\} \tag{1}$$

$$\text{subject to } \; 1 \leq \sum_{j=1}^{N_t} x_{ij} \leq L_t, \; \forall i \in \mathbb{S}_{a,N_a} \tag{2}$$

$$\sum_{i=1}^{N_a} x_{ij} = 1, \; \forall j \in \mathbb{S}_{t,N_t} \tag{3}$$

where binary variable $x_{ij} \in \{0,1\}$, $\forall i \in \mathbb{S}_{a,N_a}$, $\forall j \in \mathbb{S}_{t,N_t}$, denotes whether or not task j is assigned to agent i, i.e., $x_{ij} = 1$, if task j is assigned to agent i and 0 otherwise. The index sets for N_a agents and N_t tasks are defined as $\mathbb{S}_{a,N_a} \triangleq \{1, \ldots, N_a\}$, $\mathbb{S}_{t,N_t} \triangleq \{1, \ldots, N_t\}$, respectively. Vector $p_i(x_{ij})$ represents a sequence of ordered indices of the tasks assigned to agent i, according to x_{ij}. The local cost function $c(p_i(x_{ij}))$ computes the cost for agent i to execute tasks in p_i, which can be calculated using the attributes of tasks (e.g., position, velocity, types, etc.) and agents (e.g., position, fuel/battery level, payload limit, available functions, etc.). For example, in a robot routing problem, the cost can be represented as the distance between an agent and a task. Inequality (2) represents that every agent is assigned at least one task, but no more than L_t ($L_t \geq 1$) tasks. Equation (3) reveals that task j, $\forall j \in \mathbb{S}_{t,N_t}$, can only be signed to a single agent. This formulation assumes that every agent is expected to complete a task or a series of tasks independently such that no task has more than one agent assigned to it.

The proposed formulation in (1)–(3) may be extended to accommodate the situation where multiple agents collaboratively execute a task. In this case, (3) will become an inequality and the evaluation of the cost for executing a task by multiple agents needs to be newly derived as it should not be the sum of the local costs of each agent to execute the same tasks. The discussion of this situation is out of the scope of this article.

Algorithm 1: Hungarian Algorithm [23].

1: **Procedure** $(X, Cost)$=HUNGARIAN(C) % Matrix C is an $n \times n$ cost matrix that can be computed using the attributes of agents and tasks; matrix X is the assignment matrix, and $Cost$ is the overall cost induced by using the assignment matrix X.
2: Given the cost matrix, C:
3: Subtract the smallest entry in each row from all the entries of its row.
4: Subtract the smallest entry in each column from all the entries of its column.
5: Draw lines through appropriate rows and columns so that all the zero entries of C are covered and the minimum number of such lines is used.
6: **Procedure**: (T)est for optimality :
7: If the minimum number of covering lines is n,
8: An optimal assignment of zeros is possible and the assignment is finished.
9: **End If**
10: If the minimum number of covering lines is less than n,
11: An optimal assignment is not yet possible. Proceed to line 14.
12: **End If**
13: **End Procedure**
14: Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to line 5.
15: **End Procedure**
16: Return X and $Cost$.
17: **End Procedure**

B. Diameter of a Network

Define $\delta_G(\tau)$ as the diameter of the network whose adjacency matrix at time τ is denoted by $G(\tau) \triangleq [g_{ij}(\tau)]$, where $g_{ij}(\tau) = 1$ if agents i and j are neighbors, and $g_{ij}(\tau) = 0$ otherwise. Index $\delta_G(\tau)$ measures the topological length or extent of a graph by counting the number of edges in the shortest path between the most distant vertices, which is given by $\delta_G(\tau) = \max_{i,j}\{s_G(i,j,\tau)\}$, where $s_G(i,j,\tau)$ is the number of edges in the shortest path from vertices i to j for the network at time τ. In other words, $\delta_G(\tau)$ describes the longest shortest path between any two vertices of a graph [45], [46].

C. Hungarian Method

Kuhn [23] presented the Hungarian method to find a minimal cost assignment with $N_a = N_t$ and $L_t = 1$, as summarized in Algorithm 1. The Hungarian algorithm accomplishes the

assignment by using a cost matrix, $C \triangleq [c_{ij}]$, and the assignment process can be interpreted as a procedure for solving a maximum-weight matching problem, where the cost matrix, C, is manipulated iteratively to obtain an assignment with the minimal global cost. To achieve this, the smallest entry in each row and each column of the cost matrix is subtracted from the other entries of each row and column, respectively. When a conflict takes place, i.e., two or more agents select the same task, line 14 in Algorithm 1 is applied to find the smallest entry of each row to resolve the conflict. This process is repeated until a conflict-free assignment is obtained. Note that Algorithm 1 is a deterministic process in the sense that given the same cost matrix, C, it always produces the same assignment X and resulting $Cost$.

There are various ways to implement the Hungarian method [47], [48]. Kuhn–Munkres transforms the problem from an optimization problem of finding a max-weight matching into a combinatorial optimization problem of finding a perfect matching. It should be noted that the Hungarian method always produces an optimal assignment [49].

D. Implicit and Explicit Coordination

In this article, we assume that agents have shared mental models for coordination, which can be defined as individually held knowledge structures that help team members function collaboratively in their environments [50], [51]. It has been established in teamwork studies that shared mental models improve team ability to communicate and coordinate and serve as an aid to cognition, reasoning, and decision-making. In multiagent systems, agents coordinate effectively by using a number of implicit and explicit mechanisms and processes [52]. The *explicit coordination* mechanism refers to the processes (e.g., communication and data sharing) that agents purposely employ for coordination. For example, CBBA [38] only works under an explicit coordination mechanism because specific messages for task allocation need to be exchanged among agents to complete the task-allocation process. The *implicit coordination* relies on an agent's anticipation of information and resources needed by other agents in the network. For example, agents in DHBA [18] work under an implicit coordination mechanism because every agent does not need to exchange information, particularly for task allocation, and conflict-free task allocation can be achieved if agents have a shared knowledge base about the task and other agents (e.g., the numbers of tasks and agents, the cost for an agent to perform a task).

III. Distributed Matching-By-Clone Hungarian-Based Algorithm

As the problem described in (1)–(3) is NP hard, we present a suboptimal solution to the problem and summarize it in the proposed DMCHBA. We are targeting at solving the following optimization problem subject to the same constraints in (2) and (3)

$$\min_{x_{ij}} \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} c_{ij} x_{ij} \right) \tag{4}$$

where c_{ij} demotes the cost for agent i to perform task j. DMCHBA assigns tasks to an agent using (4) and then applies a local planning algorithm to determine the order of task execution.

DMCHBA is designed by adopting an implicit coordination mechanism and contains two iterative phases. In phase I, the communication phase, each agent communicates with its neighboring agents to exchange information that is used to build its own cost matrix. In phase II, the assignment phase, each individual agent applies the Hungarian method (see Algorithm 1) to complete the assignment. As the Hungarian algorithm only accepts a squared cost matrix, we propose a novel procedure to establish a squared cost matrix for a multitask allocation problem. Before presenting the details of DMCHBA, we first introduce several key definitions and assumptions.

A. Definitions

Definition 1 (Cloned agent set): A *cloned agent set* is a copy of all given agents whose attributes (e.g., capabilities, position, velocity, etc.) are the same as the given agents.

Definition 2 (Pseudotask): A *pseudotask* is a task whose cost to be executed by any agent is infinite.

B. Assumptions

The assumptions of the proposed DMCHBA are as follows:

1) Throughout the allocation process, the real attributes (e.g., positions, indices, capabilities, etc.) of all agents and tasks are unchanged, whereas the attributes of tasks that an agent possesses would change after every iteration of communication.
2) Each agent knows its own and other agents' attributes and the number of tasks, N_t.
3) An undirected and connected communication network [53], [54] exists among all agents throughout the process of the algorithm. An agent is able to communicate with another agent if their relative distance is within a predefined communication range limit, and these two agents are called neighbors of each other.
4) Due to the limited communication range, an agent may not be able to directly exchange information with another agent that is not a neighbor of its own in the network. Therefore, it may take a number of steps for all agents to converge to the same knowledge base about the attributes of all tasks.
5) Consider a series of discrete time steps for communication among agents. In every time step, each agent only exchanges messages with its neighbors. Also, communication can be synchronous or asynchronous.
6) Every agent employs the same deterministic function to generate cost using the agent and task attributes.
7) The tasks assigned to an agent do not change over the task execution period.

Remark 1: Assumption 1) is made to discuss the convergence of the proposed DMCHBA in a baseline setting, where the agents and tasks do not move before the task allocation process is complete. When Assumption 1) does not hold, the convergence of DMCHBA would still be achievable, which is out of the scope of this work.

Algorithm 2: DMCHBA Phase I: Communication.

For agent k at time τ,

1: **If** $\tau = 0$, % beginning of the process
2: **Procedure**: (I)nitialization:
3: Define the task-attribute vector that agent k has at time τ as $\mathbf{q}_k(\tau) = [q_{k,j}(\tau)], \forall j \in \theta$, where $q_{k,j}(\tau)$ is the attribute of the j^{th} task that agent k has at time τ.
4: **If** agent k knows j^{th} task's attribute,
5: $q_{k,j}(\tau)$ is evaluated by the corresponding knowledge.
6: **Else**
7: $q_{k,j}(\tau) = \infty$
8: **End If**
9: Considering the agent-attributes vector that agent k has as $\mathbf{a} = [a_i], \forall i \in \mathbb{S}_{a,N_a}$, where a_i is the attribute of agent i.
10: Define an adjacency matrix at time τ as $G(\tau) \triangleq [g_{ij}(\tau)]$ to represent the connected communication network among agents, where $g_{ik}(\tau) = 1$ if agent i and agent k are neighbors.
11: **End Procedure**
12: **End If**
13: **Procedure**: (C)ommunication:
14: Receive $\mathbf{q}_\ell(\tau)$ from agent ℓ with $g_{\ell k}(\tau) = 1$, where $\ell \in \mathcal{N}^k$ and $\mathcal{N}^k$ stands for the neighboring agents of agent k.
15: **For** $i = 1$ to N_t
16: **If** $\mathbf{q}_{\ell,i}(\tau) \neq \infty, \forall \ell \in \mathcal{N}^k$
17: $\mathbf{q}_{k,i}(\tau) = \mathbf{q}_{\ell,i}(\tau)$
18: **End If**
19: **End For**
20: **End Procedure**

Algorithm 3: DMCHBA Phase II: Assignment.

For agent k at time τ,

1: **Procedure**: (A)ssign_task(N_t, N_a, $\mathbf{q}_k(\tau)$, $\mathbf{a}$)
2: **If** $N_a < N_t$
3: $r = \left\lceil \frac{N_t}{N_a} \right\rceil$. % Operation $\lceil\star\rceil$ rounds $\star$ to the nearest integer greater than or equal to $\star$.
4: $n = r \cdot N_a$.
5: Add $(r-1)$ cloned agent sets.
6: Add $(n - N_t)$ pseudo tasks.
7: **Else If** $N_a = N_t$
8: $n = N_a$.
9: **End If**
10: Create the new agent- and task-index sets as $\mathbb{S}_{a,n} = \mathbb{S}_{t,n} \triangleq \{1, \ldots, n\}$.
11: Build the $n \times n$ cost matrix for agent k at time τ as $C^k(\tau) \triangleq [c^k_{ij}(\tau)] \in \mathbb{R}^{n \times n}$, $\forall i \in \mathbb{S}_{a,n}$, $\forall j \in \mathbb{S}_{t,n}$, using $\mathbf{q}_k(\tau)$ and $\mathbf{a}$.
12: $[X^k(\tau), Cost^k(\tau)]$=HUNGARIAN $(C^k(\tau))$. % $X^k(\tau)$ is the binary assignment matrix; $Cost^k(\tau)$ is the overall cost generated using $X^k(\tau)$.
13: Update assigned list of tasks of agent k by removing any pseudo tasks therein.
14: Agent k collects the task(s) assigned to it using $X^k(\tau)$.
15: Determine the order of task execution for agent k using a local planning algorithm (e.g., a TSP algorithm).
16: **End Procedure**

C. Phase I: Communication

The first phase of DMCHBA is the communication process, as described in Algorithm 2. At the beginning of the task-allocation process, i.e., $\tau = 0$, agent $k, \forall k \in \mathbb{S}_{a,N_a}$, initializes its task- and agent-attribute vectors, $\mathbf{q}_k(\tau)$ and $\mathbf{a}$, respectively, using the task and agent attributes that agent k knows at time τ (lines 2–11). The jth term in $\mathbf{q}_k(\tau)$, $q_{k,j}(\tau)$, is evaluated by the knowledge that agent k has for the jth tasks in the way that if agent k knows task j's attribute it will be evaluated by the corresponding knowledge and otherwise $q_{k,j}(\tau)$ will be given the infinite value (lines 3–7). According to Assumption 2), every agent has the same agent-attribute vector $\mathbf{a}$. Then, agent k starts to communicate with its neighbors $\mathcal{N}^k$ by exchanging task-attribute vectors, where $\mathcal{N}^k$ is the list of agent k's neighbors. Then every agent compares the received task-attribute vectors and adjusts their own available task-attribute by replacing the infinite value for each of the the unknown jth task that has been evaluated by a neighbor. After a certain number of time steps (which depends on the network formation), none of the agent's task attributes have infinite values and all task attributes have been shared among agents (lines 13–18). Note that this communication phase can be done asynchronously.

D. Phase II: Assignment

The assignment phase of DMCHBA, as summarized in Algorithm 3 and illustrated in Fig. 1, addresses an optimal way of assigning tasks to agents in which the overall cost of the assignment is minimized. According to Assumption 2), each agent is assumed to know the number of the tasks (N_t) and the attributes of all agents ($\mathbf{a}$). To generate a square cost matrix for the Hungarian algorithm, an agent compares the number of the agent (N_a) and the number of tasks (N_t). If $N_a < N_t$, then to make a square cost matrix, a set of clone agents are added, which may also need to add pseudotasks. We only need to find the minimum number of cloned-agent sets so that a square cost matrix can be built. To achieve this, the nearest integer greater than or equal to the ratio, $\frac{N_t}{N_a}$, is calculated and denoted as r. Then, the smallest square cost matrix that can be built using the proposed approach is $n \times n$, where $n = r \cdot N_a$. Therefore, $(r-1)$ sets of cloned agents should be added (lines 3–5). To build a square cost matrix, $(n - N_t)$ pseudotasks will be needed as well (line 6). When $N_a = N_t$, no cloned agents set nor pseudotasks are needed to build a square cost matrix (lines 7 and 8). New index sets $\mathbb{S}_{a,n} = \mathbb{S}_{t,n} \triangleq \{1, \ldots, n\}$ are then built for the cost matrix construction (line 10). In line

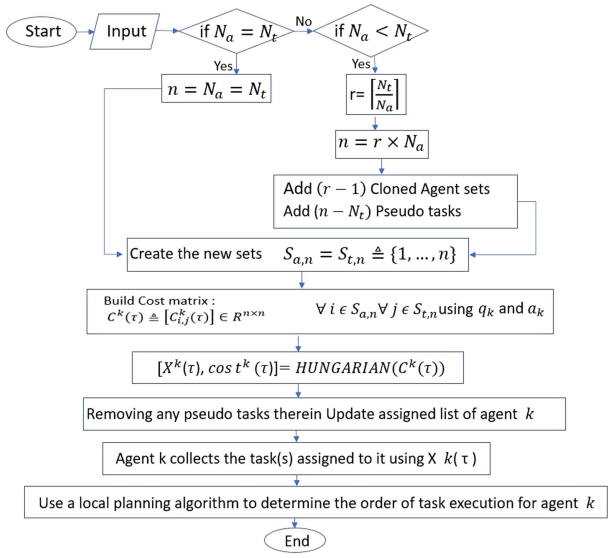

Fig. 1. Flowchart for the assignment phase of DMCHBA for agent k, $k \in \mathbb{S}_{a,N_a}$.

11, each element in the cost matrix ($c^k_{ij}(\tau)$) that agent k has at time τ is computed by using $\mathbf{q}_k$ and $\mathbf{a}$. The identification of a cost-generation function is out of the scope of this article. After the square matrix, $C^k(\tau)$, is built, the Hungarian algorithm is applied to generate the assignment. Lines 13 and 14 are to sort out the assigned tasks for each agent (by removing pseudotasks if any) and to figure out the order of task execution for each agent by using a local path planning algorithm, such as a TSP algorithm [48], [55], [56], respectively. We assume that such a local path planning algorithm guarantees to generate the shortest paths with bounded distances in finite time (line 15).

IV. Conflict-Free Resolution of DMCHBA

We define "conflict-free" as the situation, where a task is assigned to no more than one agent in the converged assignment and the assignment matrix that each agent obtains convergence to the same common conflict-free assignment matrix. As every agent completes their own task-assignment process using DMCHBA that takes the squared cost matrix built in Algorithm 3, we will explain how the conflict-free assignment is generated among agents in the following two aspects. First, we will prove that each agent will obtain the same assignment matrix using DMCHBA when given the same cost matrix that is built with the same configuration. Second, we will prove that the cost matrix built by each agent converges to the same cost matrix in a finite time. As DMCHBA uses the Hungarian algorithm (see Algorithm 1) to generate the assignment matrix, DMCHBA will generate a conflict-free assignment matrix if and only if the Hungarian algorithm generates a conflict-free assignment matrix.

A. Conflict-Free Assignment by Hungarian Algorithm

Theorem 1: For agents $i, j \in \mathbb{S}_{a,N_a}$, if $C^i(t) = C^j(t)$, $t \in [0, \infty)$, then $X^i(t) = X^j(t)$, where $X^i(t) = \text{HUNGARIAN}(C^i(t))$ and $X^j(t) = \text{HUNGARIAN}(C^j(t))$.

Proof: The Hungarian algorithm in Algorithm 1 uses a deterministic procedure, in the sense that, Algorithm 1 does not contain any random process that generates different outputs given the same input, and it also implies that running Algorithm 1 with the same input cost matrix any number of times, the same assignment will be produced [23]. This is to say that if $C^i(t) = C^j(t)$, $\forall i, j \in \mathbb{S}_{a,N_a}$ and $t \in [0, \infty)$, then we have $X^i(t) = X^j(t)$, which are obtained using Algorithm 1. To ensure $C^i(t) = C^j(t), \forall i, j \in \mathbb{S}_{a,N_a}$, as $t \to \infty$, agents i and j are required to use the same index system (i.e., the same order of agent and task indices in the cost matrix) to build their cost matrices $C^i(t)$ and $C^j(t)$. According to Assumption 1), each agent has a unique ID that is known by all other agents, a unique ordering method (e.g., an ascending order of numerical indices) of the index system for building the cost matrix is required to be adopted by every agent. ■

B. Convergence of Cost Matrix

Theorem 2: Following the communication protocol as described in Algorithm 2, $C^i(t) \to C^j(t)$, $\forall i, j \in \mathbb{S}_{a,N_a}$, in finite time.

Proof: According to Assumptions 2) and 6), each agent has the same agent-attribute vector, $\mathbf{a}$, and each agent will have the same cost when they are using the same deterministic cost-generation function. Since the cost matrix, $C^i(t)$, $i \in \mathbb{S}_{a,N_a}$, is built using the agent- and task-attribute vectors, $\mathbf{a}$ and $\mathbf{q}_i(t)$, we can obtain that $C^i(t) \to C^j(t)$, $\forall i, j \in \mathbb{S}_{a,N_a}$, as $t \to \infty$, if $\mathbf{q}_i(t) \to \mathbf{q}_j(t)$. Following the communication protocol in Algorithm 2, it guarantees $\mathbf{q}_i(t) \to \mathbf{q}_j(t)$, $\forall i, j \in \mathbb{S}_{a,N_a}$, in finite time, according to Assumptions 1) and 3). In other words, it takes a finite number of steps for a piece of information to be transmitted from one agent to another agent for a static-and-connected network (see Theorem 3) or for a dynamic network that meets the requirements in Theorem 4. So, the task-attribute vectors owned by all agents converge to the same vector in finite time. ■

V. Convergence

In this section, we discuss the convergence property of the proposed DMCHBA algorithm.

Definition 3 (Convergence): In this article, the convergence of DMCHBA implies (i) given the same cost matrix, DMCHBA produces the same assignment matrix, (ii) DMCBHA produces the conflict-free assignment matrix for all agents in finite time, and (iii) all constraints in (2)–(3) are satisfied.

A. Static Network and Convergence of DMCHBA immplementation

We first discuss the convergence of DMCHBA for multiagent systems with a static and connected communication network, whose network diameter is given by δ_{ST}. In this case, the communication phase of DMCHBA (as described in Algorithm 2) needs no more than δ_{ST} hops for a message to be transmitted from one agent to another in the connected network, in the sense that

it takes no more than δ_{ST} hops for all agents to converge to a common knowledge base, which is defined as follows.

Definition 4 (Knowledge base): The knowledge base refers to the following four factors:

1) The method of indexing agents and tasks.
2) The way of establishing the cost matrix,
3) The implementation of the Hungarian algorithm.
4) The attributes of agents and tasks.

Theorem 3 (Convergence of DMCHBA): Consider the DMCHA process over a static-and-connected network of agents with network diameter δ_{ST}. Then, the following items hold:

1) In the DMCHBA algorithm, the numbers of cloned agents and pseudotasks that need to be added are bounded.
2) The size of the list of the assigned tasks to an agent obtained after applying DMCHBA is bounded.
3) By applying DMCHBA, every agent will converge to the same conflict-free assignment matrix up to $\delta_{\text{ST}} < \infty$ iterations.
4) The time complexity of DMCHBA is $O(n^3)$.

Proof: To prove the first item, note that N_a and N_t are finite numbers, therefore, the operations in Algorithm 3 that compute the numbers of cloned agents and pseudotasks to be added generate finite numbers.

For the second item, the numbers of added cloned agents and pseudotasks are bounded by $n - N_a$ and $n - N_t$, respectively. Since N_t is finite, the number of tasks assigned to an agent is also finite and bounded by r.

To prove the third item, note that after every agent converges to the same knowledge base using no more than $\delta_{ST} < \infty$ hops, in the assignment phase of DMCHBA (as described in Algorithm 3), the same cloned agents and pseudotasks will be added to the list of each agent, which results in the same cost matrix for each agent to be given to the Hungarian algorithm for task assignment. Because the Hungarian algorithm (see Algorithm 1) always produces the same assignment matrix given the same cost matrix (see discussions in Section II-C), the entire DMCHBA is a deterministic process, in other words, no random process exists in DMCHBA that produces different assignment matrices given the same attributes of agents and tasks. Also, since the Hungarian algorithm produces a conflict-free assignment matrix in finite time [23], DMCHBA produces the conflict-free assignment for all agents in finite time.

To prove the fourth item, we divide the time to complete the entire DMCHBA process, T_{total}, into several time periods: 1) T_{add} is the time for adding clone agents and pseudo tasks, 2) T_{comm} is the time for communication, 3) T_{TA} is the time for the task assignment, i.e., the time for Algorithm 1 to complete, and 4) T_{planning} is the time for an agent to determine the order of task execution using a local planning algorithm (e.g., a TSP algorithm). Then, we have

$$T_{\text{total}} = T_{\text{add}} + T_{\text{comm}} + T_{\text{TA}} + T_{\text{planning}}. \tag{5}$$

According to Algorithm 3, the complexities of T_{add} and T_{TA} are $O(1)$ and $O(n^3)$ [48], respectively. According to Algorithm 2, T_{comm} depends on the network diameter δ_{ST}, i.e., connectivity of the communication network among agents. If the network is fully connected, which is the strongest connection, $\delta_{\text{ST}} = 2$. If the weakest connection (i.e., a chain-topology), $\delta_{\text{ST}} = n$ Therefore, the complexity of T_{comm} is $O(\delta_{\text{ST}}) \leq O(n)$.

The complexity of T_{planning} depends on the actual planning algorithm that is applied to determine the order of task execution. Let us assume that TSP algorithms are applied in this situation. For algorithms (e.g., the Nearest-Neighbor algorithm, GA, and Greedy Heuristic algorithm) to solve TSP, the Held-Karp lower bound [55] is used to evaluate the performance of a given algorithm, since finding the optimum solution might not always be feasible. If the Greedy algorithm is selected, which terminates in a reasonable number of steps and keeps the solution within 15%–20% of the Held-Karp lower bound [57], its computational complexity is $O(N^2 \log_2 N)$, where N is the number of tasks assigned to an agent and $N < n$. Therefore, the complexity of T_{total} is

$$O(1) + O(n) + O\left(n^3\right) + O\left(N^2 \log_2 N\right) = O\left(n^3\right). \tag{6}$$

■

B. Dynamic Network and Convergence of DMCHBA

Definition 5 (Maximum network diameter): Consider a multiagent system with a time-varying communication network $\mathbb{G}(\tau)$, whose matrix $G(\tau)$ varies with time τ. The maximum network diameter of the system within the ρ-iteration time interval, $(\tau[\ell], \tau[\ell + \rho - 1])$ is defined as

$$\delta_G^{\max}(\rho) \triangleq \max\left\{\delta_G(\tau[\ell]), \delta_G(\tau[\ell+1]), \ldots, \delta_G(\tau[\ell+\rho-1])\right\}$$

where $\tau[\ell]$ is the time at which an agent's ℓ's iteration occurs and $0 < \rho < \infty$ is a positive integer [38].

Theorem 4 (Convergence of DMCHBA in dynamic network): Consider the DMCHA process over a dynamic and connected network $\mathbb{G}(\tau)$ with maximum network diameter $\delta_G^{\max}(\rho)$ within ρ-iteration time interval, $(\tau[\ell], \tau[\ell + \rho - 1])$. If $\delta_G^{\max}(\rho) \leq \rho$, then it takes up to $\delta_G^{\max}(\rho)$ $(0 < \rho < \infty)$ iterations for DMCHBA to converge in the sense of Definition 3 for agents communicating synchronously or asynchronously. If $\mathbb{G}(\tau)$ is not necessarily always connected for $\tau \in (\tau[\ell], \tau[\ell + \rho - 1])$, the convergence of DMCHBA is guaranteed within $\delta_G^{\max}(\rho)$ iterations, if $\delta_G^{\max}(\rho) \leq \rho$ and if

$$\mathbb{W}(\tau[\ell]) \triangleq \mathbb{G}(\tau[\ell]) \cup \mathbb{G}(\tau[\ell+1]) \cup \cdots \cup \mathbb{G}(\tau[\ell+\rho-1]) \tag{7}$$

is fully connected, where $\mathbb{W}(\tau[\ell])$ is the graph generated by merging graphs $\mathbb{G}(\tau[\ell])$ through $\mathbb{G}(\tau[\ell + \rho - 1])$ [38].

Proof: According to Theorem 3, it takes up to $\delta_G^{\max}(\rho)$ iterations for all agents, with a synchronous communication phase, to converge to the same knowledge base if $\mathbb{G}(\tau)$ is connected for $\tau \in (\tau[\ell], \tau[\ell + \rho - 1])$, which implies that it takes DMCHBA a finite time, i.e., $\delta_G^{\max}(\rho)$ iterations, to converge, if $\delta_G^{\max}(\rho) \leq \rho$.

If $\mathbb{G}(\tau)$ is not necessarily always connected for $\tau \in (\tau[\ell], \tau[\ell + \rho - 1])$, a message sent by any agent is also guaranteed to be delivered to any other agent within up to $\delta_G^{\max}(\rho)$ iterations, if network $\mathbb{W}(\tau[\ell])$ is fully connected [58]. Even if an agent is disconnected from any other agents at a particular time, it will receive the information transmitted by any other agents

TABLE I
SIMULATION CONFIGURATION

Group	#UAVs	Area Size	# Tasks								
1	5	40×40	10	15	20	25	30	35	40	45	50
2	7	56×56	14	21	28	35	42	49	56	63	70
3	10	80×80	20	30	40	50	60	70	80	90	100
4	15	120×120	30	45	60	75	90	105	120	135	150
5	20	160×160	40	60	80	100	120	140	160	180	200
6	30	240×240	60	90	120	150	180	210	240	270	300

at any time within the time interval, $(\tau[\ell], \tau[\ell+\rho-1])$, once it connects to an agent at another time, such that $\mathbb{W}(\tau[\ell])$ is fully connected.

Asynchronous communication can be treated as a dynamic network with synchronized communication, in the sense that the situation, where an agent is not communicating with its neighbors can be regarded as the network being disconnected for that period. If graph $\mathbb{W}(\tau[\ell])$ is fully connected, it is guaranteed that this agent will converge to the global knowledge base within $\delta_G^{\max}(\rho)$ $(0 < \rho < \infty)$ steps [58]. Then, the DMCHBA process converges in finite time, even in the case when asynchronous communication is allowed [38]. ■

VI. Numerical Results and Discussion

In order to investigate the performance of the proposed DMCHBA, we conducted Monte Carlo simulations under situations with various numbers of agents and tasks. We select a scenario where the task is to visit a location such that all selected existing algorithms (e.g., CBHA) are applicable. It should be noted that the proposed DMCHBA is applicable to other types of tasks as long as the cost can be computed.

A. Simulation Environment

Consider a scenario, where N_t targeting locations need to be visited by N_a unmanned aerial vehicle (UAVs), where $N_a < N_t$. The objective of the task allocation function is to minimize the overall travel distance of all UAVs to visit all locations. The coordinates of the UAVs and targeting locations are randomly selected in a 2-D space of size $m \times m$ units, where m is selected by a predefined equation that depends on the number of UAVs. The simulation was implemented under the assumption that the targeting locations are static and UAVs move at a constant speed. Table I summarizes six groups of simulation scenarios. In each scenario, the locations of the UAVs and areas to be visited are selected randomly with a uniform distribution. The simulations were performed on a desktop computer with Intel Core i7 CPU@3.40 GHz and 8 GB RAM. 1000 simulation runs were conducted for each scenario.

To evaluate the performance of the proposed DMCHBA, we compare DMCHBA with three state-of-the-art decentralized multitask-allocation algorithms, i.e., CBBA [38], CBHA [31], and DRHBA [32]. As explained in the introduction section, two variations of DRHBA, i.e., DRHBA-I and DRHBA-II, are considered for the comparison study. In DRHBA-I, agents maintain their locations throughout the entire task-allocation process, whereas, in DRHBA-II, an agent is "moved" to the position of its last assigned task in each iteration. A key difference between DRHBA and DMCHBA is the way of building the square cost matrix so that the Hungarian algorithm can be applied. DRHBA uses dummy agents while DMCHBA uses cloned agents. In DRHBA, assigned tasks are obtained after recursively applying the Hungarian algorithm and the cost matrix is recalculated after each iteration which dominates the total run-time of executing DRHBA. In DMCHBA, the assignment is obtained by applying the Hungarian algorithm only once.

B. Local Planning Algorithms

After each individual agent obtains the list of assigned tasks to execute, a local planning algorithm is applied to determine the order of task execution (line 15 of Algorithm 3). In the simulation scenario presented in this article, the local planning algorithm is expected to minimize the travel distance for each agent to visit all assigned tasks, which can be modeled as the TSP. Different strategies have been reported in the literature to address TSP and seven state-of-the-art TSP algorithms and their computational complexity are listed in Table II. It can be seen that the lowest complexity of the seven selected algorithms for TSP is $O(w^2)$, which is dominated by the Hungarian Algorithm with $O(n^3)$. To let the proposed DMCHBA be further dominated by the Hungarian algorithm, we proposed two approximation algorithms, i.e., the heuristic local path planning algorithm (HLPPA) and the naive local path planning algorithm (NLPPA), whose complexities are, respectively, $O(w \log w)$ and $O(r)$, where $r - 1 \leq w \leq r$, and $r = \left\lceil \frac{N_t}{N_a} \right\rceil$. HLPPA and NLPPA are with the lowest computational complexity among the algorithms listed in Table II. The details of the two new algorithms are as follows.

As summarized in Algorithm 4, HLPPA orders the tasks (assigned to an agent) using their orientation angles with respect to their centroid in the clockwise direction. The agent then goes to the closest task and continues with the remaining tasks in the ordered list. The time complexity of HLPPA is $O(w \log w)$, which is proved by the following lemma.

Lemma 1: The computational complexity of HLPPA is $O(w \log w)$.

Proof: According to Algorithm 4, the entire time to complete HLPPA, T_{HLPPA} is

$$T_{\text{HLPPA}} = T_{\text{centroid}} + T_{\text{Angles}} + T_{\text{sort}}. \tag{8}$$

where T_{centroid} is the time to find the centroid of tasks, T_{Angles} is the time to find the four-quadrant inverse tangent for each task with respect to the centroid, and T_{sort} is the time to sort the tasks according to their orientation angles. The "sort" function is implemented using the "Quick Sort" algorithm in MATLAB, whose complexity is $O(w \log w)$. Then, the time complexity of T_{HLPPA} is

$$O(1) + O(1) + O\left(w \log w\right) = O\left(w \log w\right). \tag{9}$$

■

We also designed another algorithm, NLPPA, to let an agent to execute its assigned tasks by just following the order in the

TABLE II
TIME COMPLEXITY OF STATE-OF-THE-ART TSP ALGORITHMS

Greedy Algorithm [58]	Genetic Algorithm [58, 60–62]	Dijkstra Algorithm [60, 63, 64]	Bellman-Ford Algorithm [65–67]	Dynamic Programming [68, 69]	A^* Algorithm [60]	Nearest-Neighbor Algorithm [70–72]	Proposed HLPPA	Proposed NLPPA
$O\left(w^2 \log_2 w\right)$	$O\left(gwm\right)$	$O\left(w^2\right)$	$O\left(wE\right)$	$O\left(w^2 \cdot 2^n\right)$	$O\left(2\left(w-1\right)^2\right)$	$O\left(w^2\right)$	$O\left(w \log w\right)$	$O(r)$

Varable w, $r-1 \leq w \leq r$, is the number of cities in TSP, i.e., the number of tasks assigned to an agent; $r = \left\lceil \frac{N_t}{N_a} \right\rceil$; g and m are, respectively, the number of generations and the population size for the Genetic Algorithm; E is the number of edges among cities.

Algorithm 4: Heuristic Local Path Planning Algorithm (HLPPA).

For agent i,

1: **Procedure** $[List, Cost]$= HLPPA(P^x_{agent}, P^y_{agent}, P^x_{tasks}, P^y_{tasks}) % $List$ is the ordered list of tasks for agent i to execute, $Cost$ is the total cost (i.e., the total traveling distance in this algorithm) that agent i pays to execute the assigned tasks ordered in $List$, and $P^x_{agent}, P^y_{agent}, P^x_{tasks}, P^y_{tasks}$ are the x- and y-coordinates of agent i and its assigned tasks, respectively.
2: $[P^x_{mean}, P^y_{mean}] = \text{mean}(P^x_{tasks}, P^y_{tasks})$. % Find the centroid of task locations.
3: $P^x = P^x_{tasks} - P^x_{mean}$
4: $P^y = P^y_{tasks} - P^y_{mean}$
5: $Angles = \text{atan2}(P^y, P^x)$. % Find the four-quadrant inverse tangent for each task with respect to the centroid.
6: $List_{pre} = \text{sort}(Angles, 'ascend')$. % Sort the tasks using the values of $Angles$ in an ascending order.
7: $I^{closest}_{tasks}$=the index of the task closest to agent i.
8: $List$=the reordered task list starting with the $I^{closest}_{tasks}$th task in $List_{pre}$.
9: $Cost$=the total travel distance of agent i visiting all tasks in $List$.
10: **End Procedure**

Algorithm 5: Naive Local Path Planning Algorithm (NLPPA).

For agent k,

1: **Procedure** $[List, Cost] = \text{NLPPA}(X^k)$. %$X^k$ is the assignment matrix generated by line 15 of Algorithm 3.
2: $List = []$; %initialize $List$ to be an empty vector.
3: **For** $j = 1 : r$, % $r = \left\lceil \frac{N_t}{N_a} \right\rceil$
4: $\mathcal{I}_k(j \cdot k) = \text{index}(X(j \cdot k, \mathcal{I}_k(j)) = 1)$. % Store the index of task that is assigned to task k on row $j \cdot k$.
5: $List$=$[List, \mathcal{I}_k(j \cdot k)]$;
6: **End For**
7: $Cost$=the total travel distance of agent i visiting all tasks in $List$.
8: **End Procedure**

assignment matrix (e.g., X^k for agent k), as summarized in Algorithm 5. In each cloned group of agents, a task is assigned to each agent. For $r = \left\lceil \frac{N_t}{N_a} \right\rceil$ groups of agents with $(r-1)$ sets of cloned agents, there are total r tasks (including pseudotasks) assigned to each agent, which is specified by the assignment matrix. NLPPA commands an agent to execute the tasks in each of r sets of agent groups following a top–down order in the assignment matrix. The time complexity of NLPPA is $O(r)$.

C. Comparison-Study Results

Fig. 2 shows the average total cost of the results generated by the proposed DMCHBA-HLPPA and DMCHBA-NLPPA versus the selected algorithms, i.e., CBBA, CBHA, DRHBA-I, and DRHBA-II. It can be seen that the two versions of DMCHBA (i.e., DMCHBA-HLPPA and DMCHBA -NLPPA) outperform all other algorithms. Only CBHA generates a comparable overall cost to DMCHBA-HLPPA when the number of agents is less than seven. When the numbers of agents and tasks become larger, the advantageous performance of DMCHBA becomes more obvious. When $N_a = 5$, the cost generated by DMCHBA is approximately 66% of the cost by CBBA. When $N_a = 20$, the cost by DMCHBA is only approximately 40% of the cost by CBBA. In most cases, DMCHBA-NLPPA generates the second-best results but when the number of tasks becomes sufficiently large (e.g., when $\frac{N_t}{N_a} > 6$) CBHA outperforms DMCHBA-NLPPA. The two versions of DRHBA perform in between CBBA and DMCHBA when the number of agents are equal to or less than 20 and when the numbers of agents and tasks increase, the performance of the two DRHBAs deteriorates.

Fig. 3 shows the average computational time consumed by each selected algorithm. It can be seen that CBBA needs the most time to converge. The slopes of the DMCHBA and CBHA curves are much smaller than the slope of the other curves, implying the superior time efficiency of CHBA and the proposed DMCHBA. Fig. 4 shows the zoomed-in computational-time results for CBHA, DMCHBA-HLPPA, and DMCHBA-NLPPA. As CBHA uses the clustering algorithm to group N_t tasks to N_a groups and assigns them to N_a agents, CBHA only applies the Hungarian method once on a $N_a \times N_a$ cost matrix. Therefore, the increase of N_t does not significantly increase its solution time. It also implies that the increase of N_t does not affect the solution time of the selected clustering algorithm for CBHA. It can be also seen that the two DMCHBAs outperform all other selected algorithms on the basis of computational time. This implies that the time efficiency of the Hungarian method

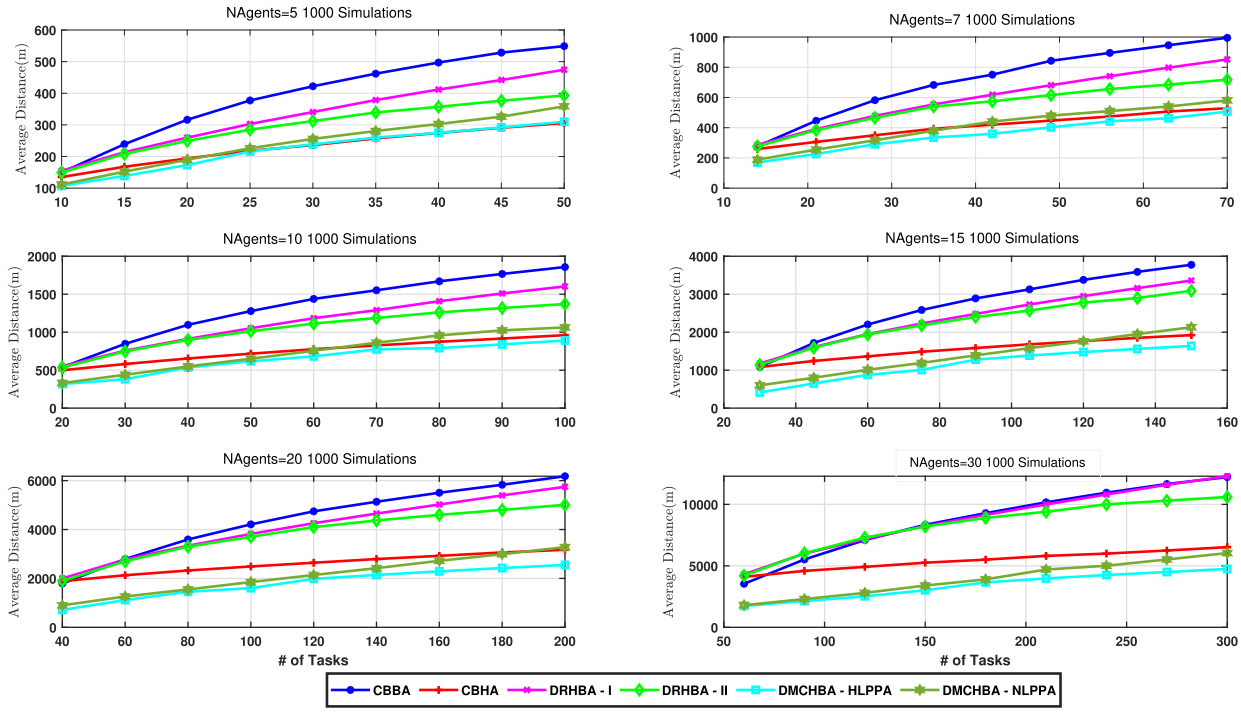

Fig. 2. Averaged total cost (i.e., distance) to execute tasks of 1000 simulations for the four selected and two proposed algorithms.

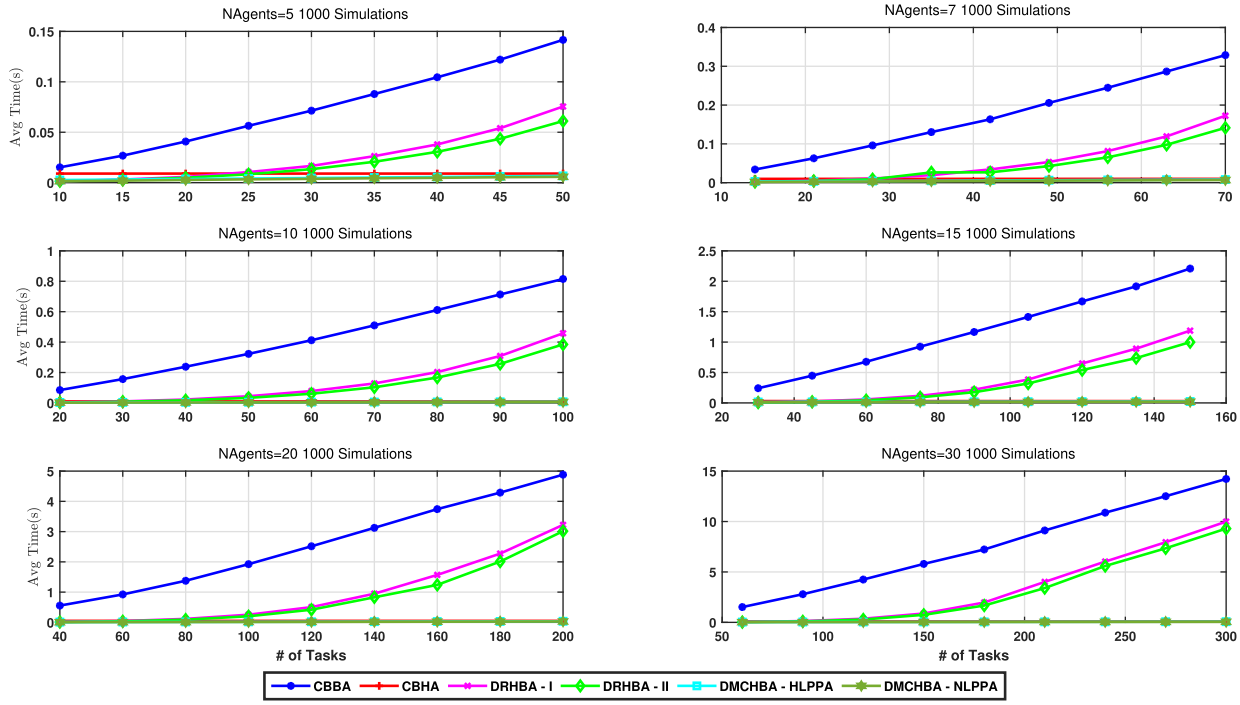

Fig. 3. Averaged computational/converging time of 1000 simulations for the four selected and two proposed algorithms..

outperforms the consensus-based auction methods (CBBA) and the recursive application of the Hungarian method (DRHBA). CBBA uses the consensus process to resolve conflict and the bundle-establishment process, which can be increasingly time-consuming when the number of tasks and agents gets larger. DRHBA eliminates the benefit of assigning multiple tasks to a single agent as only one task is assigned to an agent every iteration. It also implies that when the ratio $r = \left\lceil \frac{N_t}{N_a} \right\rceil$ is small, the cloned process in DMCHBA takes less time than the clustering algorithm in CBHA, while when r gets sufficiently large, the time for DMCHBA to complete the assignment for

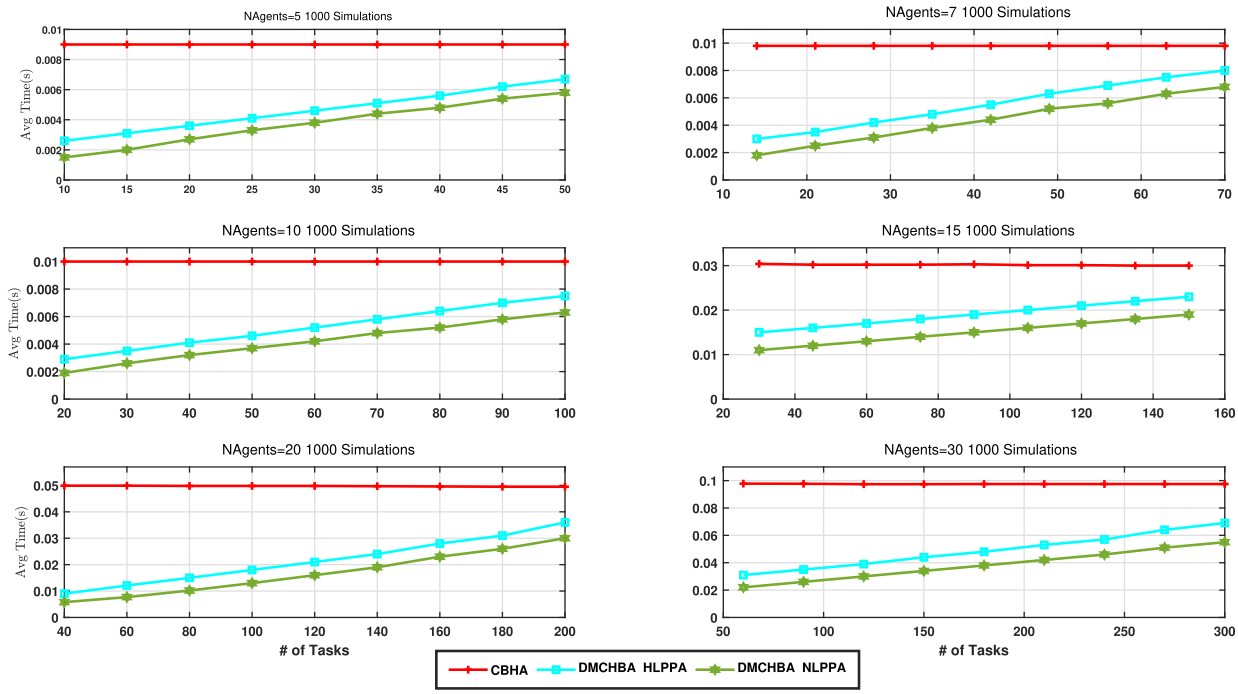

Fig. 4. Averaged computational/converging time of 1000 simulations for CBHA, DMCHBA-NLPPA, and DMCHBA-HLPPA.

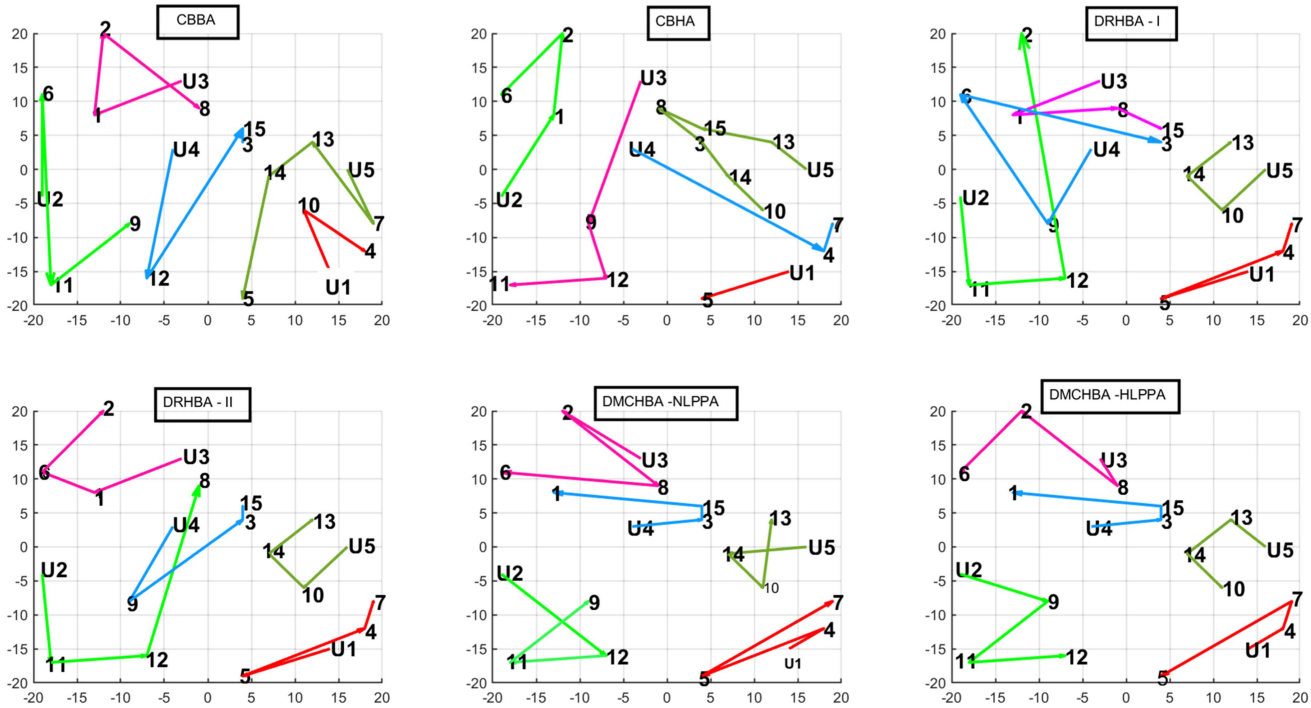

Fig. 5. Pathways generated by the algorithms for agents to execute their assigned tasks in a sample simulation with five agents (U1–U5) and 15 tasks (1–15).

an $n \times n$ cost matrix will exceed the time that CBHA takes. It should be noted that for a clustering algorithm to work in CBHA, tasks need to have "similarity" [43], while identifying similarity is a nontrivial job. For example, in a situation where three agents are to complete heterogeneous tasks like replying to emails, making a phone call, completing homework, watching a video, washing clothes, etc., CBHA would not be applicable as additional intelligence is needed to find "similarity" such that a clustering algorithm can be applied. DMCHBA is not constrained by such a situation.

It should be noted that between the two variations of DMCHBA, even though DMCHBA-HLPPA applies a local heuristic

TSP algorithm, its computation time does not increase significantly compared to DMCHBA-NLPPA, and DMCHBA-NLPPA, which arranges the order of task execution "naively," can still generate significantly better results than the selected algorithms on the basis of the overall cost (i.e., distance), especially when the numbers of agents and tasks get large (e.g., the last subfigure in Fig. 2).

D. Comparison of Resulting Paths Using Different Algorithms

Fig. 5 shows the pathway that agents take to execute their assigned tasks in a simulation of five agents and 15 tasks. It can be seen that all algorithms, except DMCHBA-HLPPA and DMCHBA-NLPPA, generate different assignments for agents. As the local path planning done by an algorithm is different, different pathways were generated. It can be seen that CBBA does not provide an optimal path for agents U1, U2, and U3, which results in a larger overall cost for all agents. DRHBA-I and DRHBA-II generate suboptimal assignments for agent U2 in both cases. DMCHBA-HLPPA generates a shorter path for agent U2 than the one by DMCHBA-NLPPA, reflecting the better performance of the HLPPA over NLPPA in path planning due to the application of a different TSP algorithm, as shown in Fig. 4.

VII. Conclusion

This article considers multitask allocation problems for multiagent systems. The original NP-hard multitask assignment problem was converted to a manageable linear sum assignment problem associated with a local planning problem. The solution to the converted problem is presented as a novel approach, namely, the DMCHBA. The proposed DMCHBA utilizes cloned agents and pseudotasks to build a square cost matrix such that the kernel task-allocation algorithm, i.e., the Hungarian algorithm, can be applied. The discussion and proof for the conflict-free assignment and convergence guarantee reveal the theoretical soundness of DMCHBA. The result of the comparison study in Monte Carlo simulations shows that DMCHBA significantly outperforms all four selected state-of-the-art multitask allocation algorithms on the basis of optimality (i.e., overall cost) and converging time.

References

[1] P. Liu et al., "A review of rotorcraft unmanned aerial vehicle UAV developments and applications in civil engineering," *Smart Struct. Syst.*, vol. 13, no. 6, pp. 1065–1094, 2014.

[2] L. Lin and M. A. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2009, pp. 709–714.

[3] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem. part 2: The asynchronous case," *SIAM J. Control Optim.*, vol. 46, no. 6, pp. 2120–2147, 2007.

[4] R. Dunford, K. Michel, M. Gagnage, H. Piégay, and M.-L. Trémelo, "Potential and constraints of unmanned aerial vehicle technology for the characterization of mediterranean riparian forest," *Int. J. Remote Sens.*, vol. 30, no. 19, pp. 4915–4935, 2009.

[5] L. Johnson, S. Herwitz, S. Dunagan, B. Lobitz, D. Sullivan, and R. Slye, "Collection of ultra high spatial and spectral resolution image data over California vineyards with a small UAV," in *Proc. 30th Int. Symp. Remote Sens. Environ.*, 2003.

[6] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu, "Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach," *Automatica*, vol. 100, pp. 75–81, 2019.

[7] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, 2019.

[8] V. B. Gylys, "Optimal partitioning of workload for distributed systems," in *Proc. Compcon Fall*, 1976, pp. 353–356.

[9] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "Workforce scheduling and routing problems: Literature survey and computational study," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 39–67, 2016.

[10] J. Yu, S.-J. Chung, and P. G. Voulgaris, "Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 327–341, Feb. 2015.

[11] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 650–665, Aug. 2006.

[12] V. Sharma, M. Savchenko, E. Frazzoli, and P. G. Voulgaris, "Transfer time complexity of conflict-free vehicle routing with no communications," *Int. J. Robot. Res.*, vol. 26, no. 3, pp. 255–271, 2007.

[13] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 98–112, 2014.

[14] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1289–1298, Aug. 2006.

[15] P. Gurfil, "Evaluating UAV flock mission performance using dudek's taxonomy," in *Proc. IEEE Amer. Control Conf.*, 2005, pp. 4679–4684.

[16] C. Bererton, G. J. Gordon, and S. Thrun, "Auction mechanism design for multi-robot coordination," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 879–886.

[17] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, Feb. 2016.

[18] S. Ismail and L. Sun, "Decentralized Hungarian-based approach for fast and scalable task allocation," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 23–28.

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

[20] J. Povh and F. Rendl, "Copositive and semidefinite relaxations of the quadratic assignment problem," *Discrete Optim.*, vol. 6, no. 3, pp. 231–241, 2009.

[21] D. P. Bertsekas, P. A. Hosein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs," *SIAM J. Control Optim.*, vol. 25, no. 5, pp. 1219–1243, 1987.

[22] P. Tseng and D. P. Bertsekas, "Relaxation methods for monotropic programs," *Math. Program.*, vol. 46, no. 1–3, pp. 127–151, 1990.

[23] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1–2, pp. 83–97, 1955.

[24] K. Yue et al., "Hungarian algorithm based virtualization to maintain application timing similarity for defect-tolerant noc," in *Proc. IEEE 17th Asia South Pacific Des. Automat. Conf.*, 2012, pp. 493–498.

[25] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[26] H. G. Tanner, "Switched UAV-UGV cooperation scheme for target detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3457–3462.

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

[28] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 3464–3468.

[29] M. A. Kamel, K. A. Ghamry, and Y. Zhang, "Fault tolerant cooperative control of multipleUAVs-UGVs under actuator faults," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2015, pp. 644–649.

[30] 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. 2012.

[31] A. Samiei, S. Ismail, and L. Sun, "Cluster-based Hungarian approach to task allocation for unmanned aerial vehicles," in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 2019, pp. 148–154.

[32] A. Samiei and L. Sun, "Distributed recursive Hungarian-based approaches to fast task allocation for unmanned aircraft systems," in *Proc. AIAA Scitech Forum*, 2020, Art. no. 0658.

[33] H. Zhu, M. Zhou, and R. Alkins, "Group role assignment via a Kuhn–Munkres algorithm-based solution," *IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Hum.*, vol. 42, no. 3, pp. 739–750, May 2012.

[34] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks*, Berlin, Germany: Springer, 2015, pp. 31–51.

[35] W. Zhao, Q. Meng, and P. W. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016.

[36] 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, 2010.

[37] H.-J. Choi, Y.-D. Kim, and H.-J. Kim, "Genetic algorithm based decentralized task assignment for multiple unmanned aerial vehicles in dynamic environments," *Int. J. Aeronautical Space Sci.*, vol. 12, no. 2, pp. 163–174, 2011.

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

[39] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, and J. W. Herrmann, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3770–3776.

[40] P. C. Lusk, X. Cai, S. Wadhwania, A. Paris, K. Fathian, and J. P. How, "A distributed pipeline for scalable, deconflicted formation flying," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5213–5220, 2020.

[41] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.*, Springer, 2010, pp. 721–730.

[42] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Trans. Autom. Control*, vol. 54, no. 9, pp. 2042–2057, Sep. 2009.

[43] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015.

[44] N. Lindsay, R. K. Buehling, and L. Sun, "A sequential task addition distributed assignment algorithm for multi-robot systems," *J. Intell. Robot. Syst.*, vol. 102, no. 2, pp. 1–12, 2021.

[45] Z. Füredi, "The maximum number of edges in a minimal graph of diameter 2," *J. Graph Theory*, vol. 16, no. 1, pp. 81–98, 1992.

[46] E. D. Demaine and M. Zadimoghaddam, "Minimizing the diameter of a network using shortcut edges," in *Proc. Scand. Workshop Algorithm Theory*, Springer, 2010, pp. 420–431.

[47] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Chelmsford, Massachusetts, USA: Courier Corporation, 1998.

[48] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[49] B. Gabrovšek, T. Novak, J. Povh, D. Rupnik Poklukar, and J. Žerovnik, "Multiple Hungarian method for k-assignment problem," *Mathematics*, vol. 8, no. 11, 2020, Art. no. 2050.

[50] S. Converse, J. Cannon-Bowers, and E. Salas, "Shared mental models in expert team decision making," *Individual Group Decis. Making: Curr. Issues*, vol. 221, pp. 221–46, 1993.

[51] C. M. Jonker, M. B. Van Riemsdijk, and B. Vermeulen, "Shared mental models," in *Proc. Int. Workshop Coordination, Organizations, Institutions, Norms Agent Syst.*, Springer, 2010, pp. 132–151.

[52] C. M. Jonker, V. Riemsdijk, M. Birna, and B. Vermeulen, "Shared mental models: A conceptual analysis," *Int. Workshop Coordination, Organizations, Institutions, Norms Agent Syst.*, pp. 132–151, 2010.

[53] N. Biggs, N. L. Biggs, and B. Norman, *Algebraic Graph Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1993, vol. 67.

[54] M. T. Clements, "Direct and indirect network effects: Are they equivalent?," *Int. J. Ind. Org.*, vol. 22, no. 5, pp. 633–645, 2004.

[55] C. Chekuri and K. Quanrud, "Approximating the Held-Karp bound for metric TSP in nearly-linear time," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci.*, 2017, pp. 789–800.

[56] K. Iwama and T. Nakashima, "An improved exact algorithm for cubic graph TSP," in *Proc. Int. Comput. Combinatorics Conf.*, Springer, 2007, pp. 108–117.

[57] H. A. Abdulkarim and I. F. Alshammari, "Comparison of algorithms for solving traveling salesman problem," *Int. J. Eng. Adv. Technol.*, vol. 4, no. 6, pp. 76–79, 2015.

[58] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. IEEE 44th Conf. Decis. Control*, 2005, pp. 2996–3000.

[59] A. R. Soltani, H. Tawfik, J. Y. Goulermas, and T. Fernando, "Path planning in construction sites: Performance evaluation of the Dijkstra, A*, and GA search algorithms," *Adv. Eng. Inform.*, vol. 16, no. 4, pp. 291–303, 2002.

[60] N. M. Razali et al., "Genetic algorithm performance with different selection strategies in solving TSP," in *Proc. World Congr. Eng.*, 2011, vol. 2, no. 1, pp. 1–6.

[61] Y. Deng, Y. Liu, and D. Zhou, "An improved genetic algorithm with initial population strategy for symmetric TSP," *Math. Prob. Eng.*, vol. 2015, no. 212794, pp. 1–6, 2015. [Online]. Available: https://www.hindawi.com/journals/mpe/2015/212794/

[62] M. Barbehenn, "A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices," *IEEE Trans. Comput.*, vol. 47, no. 2, Feb. 1998, Art. no. 263.

[63] D.-D. Zhu and J.-Q. Sun, "A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute," *IEEE Access*, vol. 9, pp. 19761–19775, 2021.

[64] P. M. Ismael, H. Y. Ibrahim, and A. B. Al-Khalil, "A real time parking reservation system based on vehicular cloud computing," in *Proc. IEEE Int. Conf. Comput. Sci. Softw. Eng.*, 2020, pp. 26–31.

[65] U. Zwick, "All pairs shortest paths using bridging sets and rectangular matrix multiplication," *J. ACM*, vol. 49, no. 3, pp. 289–317, 2002.

[66] A. Lingas, M. Persson, and D. Sledneu, "An output-sensitive algorithm for all-pairs shortest paths in directed acyclic graphs," in *Proc. Conf. Algorithms Discrete Appl. Math.*, Springer, 2022, pp. 140–151.

[67] W. Hui, "Comparison of several intelligent algorithms for solving TSP problem in industrial engineering," *Syst. Eng. Procedia*, vol. 4, pp. 226–235, 2012.

[68] A. Chentsov, M. Khachay, and D. Khachay, "Linear time algorithm for precedence constrained asymmetric generalized traveling salesman problem," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 651–655, 2016.

[69] T. S. Alemayehu and J.-H. Kim, "Efficient nearest neighbor heuristic TSP algorithms for reducing data acquisition latency of UAV relay WSN," *Wireless Pers. Commun.*, vol. 95, no. 3, pp. 3271–3285, 2017.

[70] C. A. Hurkens and G. J. Woeginger, "On the nearest neighbor rule for the traveling salesman problem," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 1–4, 2004.

[71] G. Kizilateş and F. Nuriyeva, "On the nearest neighbor algorithms for the traveling salesman problem," in *Advances in Computational Science, Engineering and Information Technology*. Berlin, Germany: Springer, 2013, pp. 111–118.

Arezoo Samiei (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Central Tehran University (IAUCTB), Tehran, Iran, in 2007, and the M.S. degree in electrical engineering control and the Ph.D. degree in electrical engineering from New Mexico State University, Las Cruces, NM, USA, in 2017 and 2023, respectively.

Her research interests include the areas of multiagent systems and unmanned aerial vehicle algorithms, with a focus on state-of-the-art multiagent system task allocation, optimization, and autonomy in complex environments.

Liang Sun (Member, IEEE) received the B.S. and M.S. degrees in automation science and electrical engineering from Beihang University, Beijing, China, in 2004 and 2007, respectively, and the Ph.D. degree in electrical and computer engineering from Brigham Young University, Provo, UT, USA, in 2012.

From 2013 to 2015, he worked as a Postdoctoral Research Fellow for a joint appointment to both the Academy Center for Unmanned Aircraft Systems Research at the US Air Force Academy, Colorado, USA, and the Unmanned Systems Laboratory at the University of Texas at San Antonio, TX, USA. He currently serves as an Associate Professor with the Department of Mechanical and Aerospace Engineering, New Mexico State University, Las Cruces, New Mexico, USA. His research interests include distributed task allocation, energy-aware autonomy for Advanced Air Mobility, and autonomy for tethered systems.