Decentralized Dynamic Task Allocation under Limited Communication Range

Alexandre Kha
Thales CortAIx-Labs, Lip6-CNRS
Palaiseau, France
alexandre.kha@thalesgroup.com

Aurélie Beynier
Lip6-CNRS
Paris, France
aurelie.beynier@lip6.fr

Christophe Labreuche
Thales CortAIx-Labs
Palaiseau, France
christophe.labreuche@thalesgroup.com

Mathieu Marchand
Thales CortAIx-Labs
Palaiseau, France
mathieu.marchand@thalesgroup.com

ABSTRACT

In Multi-Agent Systems (MAS), decentralized task allocation is an important topic that holds potential for scalability and robustness to failure points within the MAS. Classical decentralized approaches suppose that the communication graph of the agents is connected, namely that any two agents of the MAS are able to exchange information. However, this assumption is often invalid, especially when the agents have limited communication range and are moving. In this paper, we are interested in decentralized task allocation when agents have a limited communication range and receive dynamically tasks during the mission. We first introduce a procedure that extends decentralized algorithms to dynamic scenarios and limited communication range. However, agents that are not able to communicate between themselves may still plan on the same tasks and produce redundant executions, which is typically unfavorable to the allocation optimality. In response to this issue, we develop a novel algorithm leveraging generation of rendezvous points within the Consensus-Based Bundle Algorithm, that we call RDV-CBBA. This method enforces synchronization points to reduce conflicts between agents and thus task redundancy. Experiments show that RDV-CBBA outperforms significantly classical state-of-the-art methods extended to dynamic task allocation in terms of Total Distance (Min-Sum) objective, especially when the communication range is low.

KEYWORDS

Multi-Agent Systems, Decentralized Task Allocation, Dynamic Allocation, Auction-based Methods

ACM Reference Format:
Alexandre Kha, Aurélie Beynier, Christophe Labreuche, and Mathieu Marchand. 2026. Decentralized Dynamic Task Allocation under Limited Communication Range. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

1 INTRODUCTION

Multi-Agent task allocation is pivotal for applications like Search and Rescue [30], disaster response [26], logistics [23], surveillance [13], and satellite scheduling [24], requiring efficient fleet coordination. A key challenge is developing decentralized approaches where agents compute allocations locally, providing scalability.

In this paper, we study a task allocation scenario where agents have limited communication range, i.e. they can only communicate with agents within their communication range, and are all periodically notified of the arrival of new tasks during the mission, and likely while they are traveling to complete previous tasks.

Prominent task allocation methods include Distributed Constraint Optimization Problems (DCOPs) [11, 26, 29], which optimize constraint functions that quantify the value of joint allocations among agent subgroups. Another class of methods consists of auction-based algorithms like the Consensus-Based Bundle Algorithm (CBBA) [7] and its extensions (ACBBA [16], PI [30], CBTA [28] etc.), which alternate between bidding and consensus phases. Alternatively, some approaches have extended the Hungarian Method [14, 27], efficiently matching agents to task clusters. Apart from DCOP algorithms, these methods generate disjoint (conflict-free) plans. However, they typically rely on the assumption of a connected communication graph, i.e. any two agents are able to communicate with each other, by eventually using other agents to relay their messages. They also assume that all tasks to service are known in advance by the agents.

In practice, mobility and limited communication ranges often fragment the network into disjoint "islands". This is critical in dynamic scenarios where scattered agents receive new tasks but lack the connectivity to negotiate conflict-free plans.

Previous work on limited communication often focuses on static tasks [2] or relies on opportunistic coordination [22]. However, the latter is driven by chance encounters and may be insufficient for dynamic scenarios with strict objectives. Synchronization strategies based on rendezvous have also been proposed. [8, 10]. Rendezvous methods gather agents closed enough such that they able to communicate and coordinate. Nevertheless, these approaches either require computational resources to determine a rendezvous location, or lack explicit optimality-driven rendezvous selection.

Our contributions are twofold: (1) We introduce a general dynamic allocation procedure, which is based on opportunistic encounters to allow agents to handle dynamic arrival of tasks and synchronize in a decentralized manner. (2) We propose RDV-CBBA (Rendezvous CBBA), a method that explicitly integrates the selection of rendezvous points into the decentralized allocation algorithm CBBA. RDV-CBBA enforces periodic connectivity by calculating optimal meeting points, using the Barycenter or the Fermat-Weber solution [3] for total distance. This ensures that agents connected at one instant regain connectivity later, allowing them to merge knowledge, resolve conflicts, and replan as tasks appear.

2 PROBLEM STATEMENT

Notations: We consider that usual operations on sets (e.g. $|.|, \cup, \cap, \setminus$) are valid on tuples, by assimilating them to the set of their elements.

The problem addressed in this paper is formulated as a dynamic Open Vehicle Routing Problem (OVRP), a variant of the VRP where agents are not required to return to the depot. This scenario closely models surveillance missions where a fleet of UAVs is initially deployed from a base. Subsequently, new points of interest are dynamically communicated by a mission center to all agents and must be visited while minimizing the cumulative travel time of the fleet (Min-Sum objective).

Task and Agents Definition

Let $\mathcal{I} = \{i_1, \dots, i_N\}$ be the set of agents in the problem. At each timestep t, we denote the positions of the agents as $\mathbf{x}_{i_1}(t), \dots, \mathbf{x}_{i_N}(t)$. Each agent i has a velocity v_i, meaning that between two timesteps t and $t+1$, any agent i can travel a distance of at most v_i.

Let $\mathring{\mathcal{J}}$ be the set of all possible tasks. At each timestep t, a finite set of new tasks $\mathcal{J}_{\text{app}}(t) = \{j_1(t), j_2(t), \dots, j_{k_t}(t)\} \subseteq \mathring{\mathcal{J}}$ appears, with cardinality $k_t \in \mathbb{N}$. By convention, $k_t = 0$ implies that $\mathcal{J}_{\text{app}}(t) = \emptyset$. We denote by $\mathcal{J}(t) = \bigcup_{t' \leq t} \mathcal{J}_{\text{app}}(t')$ the set of all tasks that have appeared up to timestep t. Furthermore, let t_j denote the appearance time of any task $j \in \mathring{\mathcal{J}}$. During the mission, agents complete tasks, and we denote the set of tasks completed by timestep t as $\mathcal{J}_{\text{done}}(t)$. Consequently, the set of actual tasks available at timestep t is given by $\overline{\mathcal{J}}(t) = \mathcal{J}(t) \setminus \mathcal{J}_{\text{done}}(t)$.

Agents are constrained by a communication range r. At any timestep t, an agent can only communicate directly with the other agents within its communication range. We define the Communication Graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ where vertices correspond to the agents, i.e., $\mathcal{V}(t) = \mathcal{I}$. An edge exists between any two agents if they are within a distance of at most r, formally $\mathcal{E}(t) = \left\{(i,k) \in \mathcal{I}^2 \mid \|\mathbf{x}_i(t) - \mathbf{x}_k(t)\| \leq r\right\}$. We define *Connectivity Islands* (CI) as the connected components of this communication graph. Within each CI, there exists a path of edges linking any two agents, ensuring mutual reachability. We denote by $\mathcal{N}_i^+(t)$ the specific CI to which agent i belongs at timestep t, and by $D(t)$ the diameter of the graph $\mathcal{G}(t)$. Consequently, agents can only exchange information with others belonging to their current set $\mathcal{N}_i^+(t)$. While agents only communicate with neighbors within range r, messages are effectively propagated via multi-hop propagation to reach all agents in the island. An example of a MAS spatially partitioned into two CIs is presented in Figure 1.

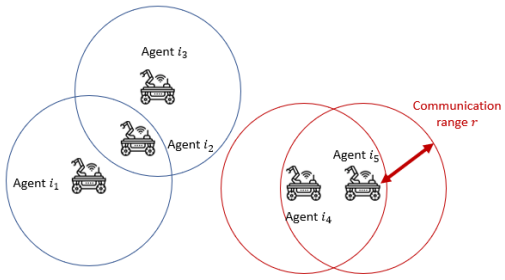


Figure 1: Two Connectivity Islands: $\{i_1, i_2, i_3\}$ and $\{i_4, i_5\}$. Agents i_1 and i_3 are not directly connected but belong to the same CI because Agent i_2 acts as a relay.

Optimization Problem

To model the optimization problem, we first introduce the following definitions. Let $\mathcal{T}_i(p_i)$ denote the time required for agent i to complete its path of tasks p_i.

Definition 1 (Admissible path). *For an agent $i \in \mathcal{I}$, a path is defined as a tuple $p_i = \left[j_1, \dots, j_{|p_i|}\right]$, where $j_\ell \in \mathring{\mathcal{J}}$ for all $\ell \in [\![1, |p_i|]\!]$. This path is admissible if and only if:*
(1) $j_\ell \neq j_m, \forall (\ell, m)$ s.t. $\ell \neq m$, and (2) $\mathcal{T}_i\Big(\left[j_1, \dots, j_k\right]\Big) \geq t_{j_k}, \forall k \in [\![1, |p_i|]\!]$. Constraint (1) ensures that an agent visits any specific task at most once. Constraint (2) ensures temporal causality, meaning an agent arrives at a task location only after the task has appeared. We denote the set of all admissible paths for any agent as $\mathcal{P}$.

Definition 2 (Admissible allocation). *An admissible path allocation $\boldsymbol{p} = \langle p_{i_1}, \dots, p_{i_N} \rangle$ is a tuple of joint admissible paths such that every task is covered, i.e., $\bigcup_{i \in \mathcal{I}} p_i = \mathring{\mathcal{J}}$. The set of all admissible allocations is denoted $\mathbb{P} \triangleq \mathcal{P}^{|\mathcal{I}|}$.*

Definition 3 (Conflict-free allocation). *An admissible allocation $\boldsymbol{p} = \langle p_{i_1}, \dots, p_{i_{N_a}} \rangle \in \mathbb{P}$ is conflict-free if $p_i \cap p_k = \emptyset$ for all distinct pairs $(i,k) \in \mathcal{I}^2$. Hence, each task is allocated to one agent.*

The task allocation optimization problem is defined as minimizing the total travel time (Min-Sum objective).

Problem 1 (Ideal Optimization Problem).

$$\min_{\boldsymbol{p} \in \mathbb{P}} \sum_{p_i \in \boldsymbol{p}} \mathcal{T}_i(p_i) \tag{1a}$$

$$s.t. \quad p_i \cap p_k = \emptyset, \qquad \forall (i,k) \in \mathcal{I}^2 \text{ s.t. } i \neq k. \tag{1b}$$

The objective function $\sum \mathcal{T}_i(p_i)$ represents the cumulative travel time of the fleet, which serves as a proxy for global energy consumption. Constraint (1b) *enforces a conflict-free allocation. Note that all tasks are covered as the space of solutions is $\mathbb{P}$ (cf. Definition 2).*

This problem formulation is "ideal" (or offline) because it assumes that agents possess a priori knowledge of all tasks $j \in \mathring{\mathcal{J}}$, including their appearance times, before the mission begins. In a realistic scenario, agents are notified of new tasks dynamically during the mission. Consequently, they must solve the allocation problem for these new tasks online, often while unable to communicate with other agents due to limited communication ranges. Therefore, in Section 4, we propose a decentralized method to solve online this dynamic problem efficiently.

3 RELATED WORK

3.1 Decentralized Task Allocation

Decentralized techniques for multiagent task allocation include Distributed Constrained Optimization Problems (DCOP) algorithms, which optimize utility sums among neighbors [5, 11, 29]. While suited for problems with high inter-dependencies, these methods are often myopic, preventing efficient path planning and are prone to conflicted allocation. Auction-based methods, such as the Consensus-Based Bundle Algorithm (CBBA) [7] typically alternate between a bidding phase and a consensus phase. CBBA guarantees convergence in a conflict-free allocation in finite number of iterations as well as 50% optimality for Diminishing Marginal Gain functions in fully-connected communication graphs. The Global CBBA (GCBBA) restores this guarantee for any communication graph and is also faster to converge [17]. Many more extensions have been designed to tackle more complex problems like communication asynchronicity (ACBBA [16]), coalitions (CBTA [28]), search and rescue (PI [30]), and human-guided allocation (I-CBBA [12]). Optimization-based algorithms like the Distributed-by-Matching Clones Hungarian-Based Algorithm (DMCHBA) [27] offer a very competitive alternative using first task clustering based on the Hungarian Method [20], and then routing. However, these algorithms primarily assume a connected communication graph at every timestep. In dynamic missions, as the network topology changes, conflict-freeness is lost and performance deteriorates because agents may do redundant actions.

3.2 Limited Communication and Dynamic Task Arrival

Cao et al. [4] analyzed the impact of packet loss on decentralized task allocation algorithms with dynamic task arrival. They found that CBBA maintains better optimality while Hungarian methods exhibit fewer conflicts. However, they did not address topological disruption caused by mobility, e.g. due to limited communication range. Ponda et al. [25] addressed time-constrained tasks and dynamic scenarios but relied on a mission center to select communicating subgroups, which does not enter our scope of study. More recently, Bai et al. [2] proposed the Distributed Auction Algorithm (DAA) for static batches, which merges plans when disjoint groups meet. Their approach relies on agents distinguishing between neighbors they had met previously and those they had not. However, relying on the "novelty" of neighbors is less relevant in dynamic settings because of the dynamicity of the set of tasks to execute.

3.3 Coordination via Rendezvous

The concept of rendezvous is well-established in control theory [6, 18, 21] and multi-agent exploration [9]. Relevant applications include: (1) Supply and Refueling: Do et al. [10] address a task allocation problem where supplier and receiver agents must rendezvous for battery recharging. Their method is centralized, first generating individual plans and subsequently selecting a rendezvous point from a candidate list to insert into the agents' trajectories. (2) Opportunistic Exploration: Luperto et al. [22] tackle area exploration with communication-constrained agents, and use backtracking to high-connectivity areas (e.g. corridors) to increase data exchange likelihood. (3) Synchronization Instants: Da Silva and Chaimowicz [8] treat exploration as a scheduling problem with random rendezvous points. While opportunistic encounters lack strict synchronization, explicit strategies like [10] ensure coordination but have not yet been adapted to dynamic auction mechanisms and may be computational depending on the size of the rendezvous candidates set. On another hand, Bai et al. [1] proposed a barycenter rendezvous for static scenarios and in a semi-centralized setting. Our work bridges this gap by calculating geometric rendezvous points (Barycenter, Fermat-Weber [3]) in a decentralized manner to forcefully synchronize agents for efficient replanning.

4 DECENTRALIZED DYNAMIC ALLOCATION FRAMEWORK

The ideal optimization problem formalized in Section 2 assumes that all tasks and their appearance times are known a priori. In this section, we present the practical online problem solved by the agents and introduce a general decentralized procedure to approximate the optimal solution of Problem 1 under dynamic conditions.

4.1 Practical Task Allocation Problem

At any timestep t, agents receive a new batch of tasks and must solve the following practical optimization problem:

Problem 2 (Practical Optimization Problem).

$$\max \sum_{i \in \mathcal{I}} \sum_{j \in \overline{\mathcal{J}}(t)} c_{ij}(p_i(t))\, x_{ij} \tag{2a}$$

$$\text{s. t.} \quad x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{I} \times \overline{\mathcal{J}}(t), \tag{2b}$$

$$\sum_{i \in \mathcal{I}} x_{ij} = 1, \qquad \forall j \in \overline{\mathcal{J}}(t), \tag{2c}$$

where the binary variable x_{ij} indicates whether task j is assigned to agent i, $p_i(t)$ is the current path of agent i, and $c_{ij}(p_i(t))$ represents the utility gain of adding task j to $p_i(t)$. To guarantee algorithmic convergence (see Section 5.2), c_{ij} serves as a proxy for the Min-Sum objective rather than its strict marginal cost. Equation (2c) *ensures all tasks are covered without conflicts.*

Crucially, agents cannot efficiently solve Problem 2 in a decentralized framework if the communication graph $\mathcal{G}(t)$ is not connected. Two disconnected Connectivity Islands may unknowingly allocate the same tasks, violating the conflict-free constraint, since they are unaware of each other's allocation. This leads to task redundancy and suboptimal performance. To address this, we present a procedure tackling the dynamic problem while mitigating inter-agent conflicts.

4.2 Dynamic Allocation Procedure

In the following analysis, we focus on the allocation procedure at timestep t and we omit the time dependency t for brevity. Recall that $\mathcal{N}_i^+$ denotes the CI to which agent $i \in \mathcal{I}$ belongs and let $\overline{\mathcal{J}_i} \subseteq \mathring{\mathcal{J}}$ denote the set of available tasks known to agent i. In this section, $\overline{\mathcal{J}_i} = \mathcal{J}_{\text{app}}$ for all $i \in \mathcal{I}$, i.e., it is a local copy of the appearing tasks.

We propose a novel dynamic allocation method (Algorithm 1) that extends the work of Bai et al. [2], initially for a single batch of tasks. The key features are:

(1) *Task Synchronization:* Upon connecting with a CI, agents communicate both their available and completed tasks to ensure coordination and prevent future redundancy (*Line 5*). They explicitly remove already completed tasks from their current paths.
(2) *Lazy Allocation:* Agents initially assign themselves the newly appearing tasks (*Line 4*). Reallocation is triggered using a conflict-free decentralized algorithm only when agents encounter a CI where task conflicts are detected (*Lines 7-9*). This strategy avoids unnecessary computations on already settled tasks, preserving resources and algorithm stability.

At the end of each timestep (*Line 10*), agents update their position $\mathbf{x}_i$ and their path p_i.

Algorithm 1 Dynamic allocation procedure for agent i

1: **procedure** DYNAMIC AGENT i
2: **while** True **do** ▷ *Iterates for each timestep t*
3: $\overline{\mathcal{J}_i} \leftarrow \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
4: $p_i \leftarrow$ Add tasks from $\overline{\mathcal{J}_i}$ to path
5: $\mathcal{N}_i^+, \bigcup_{k\in\mathcal{N}_i^+} p_k, \bigcap_{k\in\mathcal{N}_i^+} p_k \leftarrow$ Update the CI $\mathcal{N}_i^+$
6: ▷ *Check for conflicts within the CI*
7: **if** $|\bigcap_{k\in\mathcal{N}_i^+} p_k| > 0$ **then**
8: $p_i \leftarrow \emptyset$ ▷ *Reset p_i*
9: $p_i \leftarrow$ Run Allocation Algorithm within community $\mathcal{N}_i^+$ on tasks $\bigcup_{k\in\mathcal{N}_i^+} p_k$ ▷ *The algorithm is conflict-free*
10: $\mathbf{x}_i, p_i \leftarrow$ Move along path ▷ *Update position and path*

This procedure is generic and can wrap any decentralized task allocation algorithm (Line 9). The method relies on *opportunistic rendezvous* and does not strictly guarantee a conflict-free global state at all times. When new tasks appear while agents are disconnected, they may locally allocate the same tasks, leading to potential redundancy until connectivity is restored. Furthermore, this procedure requires agents to ping for neighbors at each timestep.

However, if agents are prohibited from re-taking a task they have previously conceded to another agent, convergence is guaranteed, as stated in the following theorem:

THEOREM 1 (CONVERGENCE OF ALGORITHM 1). *Given any conflict-free allocation method, and assuming that the total task set $\mathring{\mathcal{J}} = \bigcup_{t\geq 0} \mathcal{J}_{app}(t)$ is finite and agents are notified of the same task set $\mathcal{J}_{app}(t)$ at each timestep t, agents will eventually complete all tasks, possibly with intermediate conflicts.*

PROOF. Let t be the time instant when all tasks in $\mathring{\mathcal{J}}$ have appeared. Since no further tasks arrive after t, the problem reduces to a static allocation on the remaining set $\overline{\mathcal{J}}(t)$. All agents initially plan routes for the remaining tasks $\overline{\mathcal{J}}(t)$ (cf. Line 4), ensuring coverage $\overline{\mathcal{J}}(t) \subseteq \bigcup_{i\in\mathcal{I}} p_i(t)$. For any agent i, the path p_i is modified only when meeting other agents. Crucially, any task released by agent i during conflict resolution is necessarily allocated to another agent (cf. Line 9). Hence, for any task $j \in \mathring{\mathcal{J}}$ and timestep $t' \geq t$:

$$j \in \left(\bigcup_{i\in\mathcal{I}} p_i(t')\right) \cup \mathcal{J}_{\text{done}}(t'). \tag{3}$$

The path $p_i(t)$ can only be modified a finite number of times, which is bounded by $|\mathcal{I}| - 1$. Indeed, if an agent meets any other agent in the MAS, applies a conflict-free allocation algorithm with them (Line 9), and without possibility to re-obtain a lost task, then it will reach conflict-free consensus with the whole MAS. Therefore, the path of each agent stabilizes after a finite number of modifications, allowing them to complete their assigned tasks. Let $t_{\text{end}} \geq t$ be the first timestep where all agents have completed their paths. From Equation (3), it follows: $\forall j \in \mathring{\mathcal{J}}, \quad j \in \mathcal{J}_{\text{done}}(t_{\text{end}})$. □

5 RENDEZ-VOUS CBBA (RDV-CBBA)

5.1 General Procedure

In Section 4, we have presented a general procedure extending any decentralized task allocation method to handle dynamic task arrival and limited communication range. However, this procedure fails to be conflict-free because agents from different CI may plan on the same tasks and not even meet to revise their plans. This impacts negatively performance since it generates unnecessary travels. In this section, we present the method RDV-CBBA to address this drawback. The RDV-CBBA framework exploits the fact that all agents belong to a single CI at the start of the mission. The core idea is to periodically restore this global connectivity to enable efficient coordination. We propose two variants of this approach: a *Myopic* version (Algorithm 3) and a *Proactive* version (Algorithm 4). In the Myopic variant, agents first meet and then plan a rendezvous point based on their allocated tasks. Conversely, in the Proactive variant, agents dynamically compute a future rendezvous point as soon as new tasks appear, before strictly needing to meet.

In this section, the available task set $\overline{\mathcal{J}_i}$ for any agent $i \in \mathcal{I}$ corresponds to the set of tasks arrived since previous rendezvous, i.e. for any timestep t, which previous rendezvous instant is $t_1 \leq t$, we have $\overline{\mathcal{J}_i}(t) = \bigcup_{t'\in[\![t,t_1]\!]} \mathcal{J}_{\text{app}}(t')$.

5.2 Principles of the Global CBBA (GCBBA)

We recall the principles of the GCBBA, an efficient method for static task allocation which is key to RDV-CBBA. The GCBBA is an improvement of the baseline CBBA by accelerating convergence and handling any type of connected communication graph in static task allocation [17]. In this subsection, let $\hat{\mathcal{J}}$ denote generally any batch of tasks to be allocated, and $\hat{\mathcal{I}}$ any CI. In this approach, any agent $i \in \hat{\mathcal{I}}$ iteratively places a bid $c_{ij}(p_i)$ on each task $j \in \hat{\mathcal{J}}$ (Auction phase) to then negotiate assignments with its peers (Consensus phase). Bids correspond exactly to the marginal utilities in Problem 2. The convergence of GCBBA is guaranteed within $|\hat{\mathcal{J}}|$ iterations, provided that the bidding function satisfies the *Diminishing Marginal Gain* (DMG) property, similar to submodularity in resource allocation [19].

DEFINITION 4 (DIMINISHING MARGINAL GAIN (DMG)). *A bidding function c_{ij} satisfies the DMG property if, for any agent $i \in \hat{\mathcal{I}}$, any path p_i, and any two distinct tasks $j, k \in \hat{\mathcal{J}} \setminus p_i$, the inequality $c_{ij}(p_i \oplus_\ell k) \leq c_{ij}(p_i)$ holds for all insertion indices $\ell \in [\![1, |p_i| + 1]\!]$, where $(p_i \oplus_\ell j)$ denotes the path p_i with task j inserted at position ℓ.*

The GCBBA procedure is outlined in Algorithm 2. The convergence detector used originally is omitted for simplicity, but convergence is still guaranteed. It utilizes the following variables and functions, largely inherited from standard CBBA:

Algorithm 2 Global Consensus-Based Bundle Algorithm (GCBBA)

procedure GCBBA
  $T \leftarrow 0$
  while $(T < |\hat{\mathcal{J}}|)$ **do**
    for $i \in \hat{\mathcal{I}}$ **do** ▷ *Auction Phase*
      $(z_i, y_i, p_i) \leftarrow \texttt{ADD}(z_i, y_i, p_i)$
    for $t_{cons} \in [\![1, 2\,D]\!]$ **do** ▷ *Consensus Phase*
      $T_{loc} \leftarrow 2 \cdot D \cdot T + t_{cons}$ ▷ *Global iteration marker*
      for $i \in \hat{\mathcal{I}}$ **do**
        $\texttt{SEND}(y_i, s_i)$ ▷ *Broadcast to neighbors* $\mathcal{N}_i$
        $\texttt{RECEIVE}((y_m)_{m\in\mathcal{N}_i}, (s_m)_{m\in\mathcal{N}_i})$
      for $i \in \hat{\mathcal{I}}$ **do** ▷ *Conflict Resolution*
        for $k \in \mathcal{N}_i$ **do**
          $(z_i, y_i, p_i) \leftarrow \texttt{RESOLVE}(y_i, y_k, s_i)$
          $s_i \leftarrow \texttt{S-UPDATE}(T_{loc}, s_i, s_k)$ ▷ *Update timestamps*
    $T \leftarrow T + 1$
  return $(p_i)_{i\in\hat{\mathcal{I}}}$

(1) $\mathcal{N}_i = \{k \in \hat{\mathcal{I}} \setminus \{i\} \mid \|\mathbf{x}_i - \mathbf{x}_k\| \le r\}$: The set of neighbors of agent i, defined by the communication range r.
(2) $y_i = (y_{ij})_{j\in\hat{\mathcal{J}}}$: A vector where $y_{ij} \in \mathbb{R}$ represents the highest bid for task j known to agent i.
(3) $z_i = (z_{ij})_{j\in\hat{\mathcal{J}}}$: A vector where $z_{ij} \in \hat{\mathcal{I}}$ indicates the winning agent for task j as perceived by agent i.
(4) $s_i = (s_{ik})_{k\in\hat{\mathcal{I}}}$: A timestamp vector where $s_{ik} \in \mathbb{N}$ records the last GCBBA iteration (variable T in Algorithm 2) in which agent i received information from agent k.
(5) ADD: A method that attempts to insert a task $j \in \hat{\mathcal{J}}$ into agent i's path at the optimal position, provided the marginal gain exceeds the current highest bid y_{ij}.
(6) RESOLVE: A conflict resolution method based on local communication. Whether to keep or release tasks is decided through a rule table (see [7, Table 1]).

The bidding function c_{ij} is critical for aligning the GCBBA solution with the Min-Sum objective (Problem 2). However, standard marginal costs derived directly from the Min-Sum objective $(\mathcal{T}_i(p_i \oplus_\ell j) - \mathcal{T}_i(p_i))$ do not satisfy the DMG condition (Definition 4), which is required for convergence.

While "warping" technique exists to artificially enforce DMG [15], it forces agents to misreport task valuations, which distorts the optimization landscape. Therefore, we adopt the *Total Path Bid* proposed in [17], which is proven to be DMG:

Definition 5 (Total Path Bid). *For any path p_i and task $j \in \hat{\mathcal{J}} \setminus p_i$, the Total Path bid is defined as:*

$$c_{ij}(p_i) = X - \min_{1\le\ell\le|p_i|+1} \mathcal{T}_i(p_i \oplus_\ell j), \tag{4a}$$

where X is a sufficiently large constant ensuring $c_{ij} > 0$.

Using this bidding scheme, the winning agent-task pair (i^*, j^*) selected at a GCBBA iteration satisfies the following equation: $(i^*, j^*) = \arg\min_{(i,j)\in\hat{\mathcal{I}}\times\hat{\mathcal{J}}} \mathcal{T}_i(p_i \oplus_{\ell^*} j)$. Consequently, this bid prioritizes agents capable of completing the task with the lowest total path duration. While this implicitly promotes load balancing, it serves as a robust proxy for the Min-Sum objective while maintaining the necessary theoretical convergence guarantees.

5.3 Myopic RDV-CBBA

First, we introduce Myopic RDV-CBBA, wherein agents meeting at a rendezvous (RDV) use GCBBA to allocate available tasks and determine a new RDV at the end of their paths. The Myopic RDV-CBBA procedure is detailed in Algorithm 3. Initially (or upon reaching a rendezvous), agents are co-located and fully connected (*Line 4*). They execute the GCBBA algorithm on the available tasks $\overline{\mathcal{J}_i}$ to generate conflict-free paths. Critically, they identify the set of "path-ending" tasks, denoted as $\mathcal{J}_{\text{last}} = \bigcup_{i\in\mathcal{I}} \{p_{i\,|p_i|}\}$ (*Line 5*). Based on these final task locations, the agents effectively compute a consensus rendezvous (RDV) point, such as the barycenter of $\mathcal{J}_{\text{last}}$, and append it at the end of their paths (*Lines 6-7*). Other methods for selecting this point are discussed in Section 5.6. Figure 2a describes an example of myopic rendezvous planning.

Agents then execute their paths (*Line 9*). The RDV is considered reached when agents are within a distance of $r/2$ from it, as it is sufficient to ensure full connectivity. Once synchronized at the RDV, the process repeats for any new tasks received during travel.

Algorithm 3 Myopic RDV-CBBA for agent i

procedure RDV-CBBA(M) AGENT i
  while True **do**
    $\overline{\mathcal{J}_i} \leftarrow \overline{\mathcal{J}_i} \cup \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
    if (all $k \in \mathcal{I}$ at (RDV OR initial position)) **then**
      $p_i, \mathcal{J}_{\text{last}} \leftarrow$ GCBBA within $\mathcal{I}$ on tasks $\overline{\mathcal{J}_i}$
      RDV $\leftarrow$ Compute RDV based on $\mathcal{J}_{\text{last}}$
      $p_i \leftarrow p_i \oplus_{\text{end}}$ RDV ▷ *Append RDV to end of path*
      $\overline{\mathcal{J}_i} \leftarrow \emptyset$
    $\mathbf{x}_i, p_i \leftarrow$ Move along path

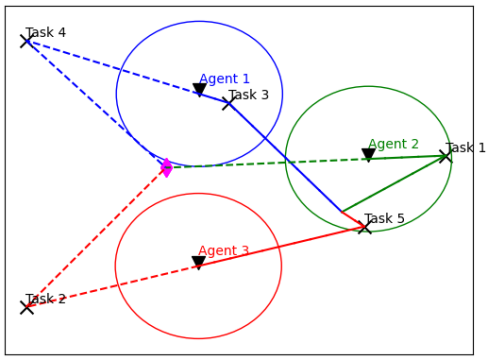
5.4 Proactive RDV-CBBA

The myopic rendezvous method allows the agents to agree on the next meeting point each time they meet. Before the agents arrive at this next meeting point, new tasks may have appeared, making this next meeting point inefficient if it is far from the new task locations. We hence propose a second version of RDV-CBBA, leveraging proactive rendezvous. In the Proactive procedure (Algorithm 4), agents initially generate a plan via GCBBA without immediately selecting a rendezvous location (*Lines 5-7*). While no new tasks appear, agents execute their paths normally. However, upon receiving a new batch of tasks, agents dynamically compute a RDV location based on the geometry of these *new* tasks and append it to their current plan (*Lines 9-10*).

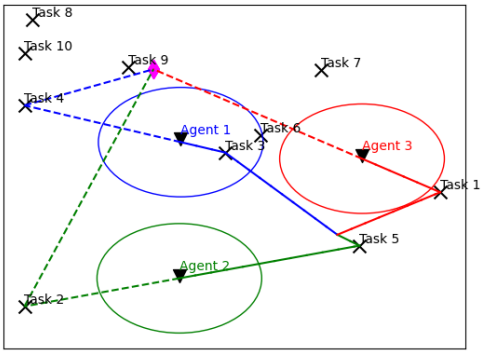
This approach allows agents to move towards the new task cluster before meeting, thereby anticipating future demand. This contrasts with the myopic strategy, which may pull agents back towards completed tasks. See an example of proactive rendezvous planning in Figure 2b. Once the RDV is reached, agents effectively re-plan, and the cycle continues.

5.5 Analysis

In this subsection, we provide the conditions for RDV-CBBA to converge under both myopic and proactive methods. Moreover, we discuss the success of convergence when new tasks are not simultaneously communicated to all agents.



(a) RDV-CBBA(M): RDV point (pink diamond) based on the barycenter of the *path-ending* tasks $\{1, 2, 4\}$.



(b) RDV-CBBA(P): RDV point (pink diamond) based on the barycenter of the *newly appearing* tasks $\{6, 7, 8, 9, 10\}$.

Figure 2: Comparison of Myopic and Proactive RDV-CBBA strategies. Triangles surrounded by a circle represent the agents and their communication range, crosses stand for tasks.

Algorithm 4 Proactive RDV-CBBA for agent i

1: **procedure** RDV-CBBA(P) AGENT i
2: **while** True **do**
3: $\overline{\mathcal{T}_i} \leftarrow \overline{\mathcal{T}_i} \cup \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
4: **if** (all $k \in \mathcal{I}$ at (RDV OR initial position)) **then**
5: $p_i \leftarrow$ GCBBA within $\mathcal{I}$ on tasks $\overline{\mathcal{T}_i}$
6: $\overline{\mathcal{T}_i} \leftarrow \emptyset$
7: RDV $\leftarrow$ None
8: **else if** $(\overline{\mathcal{T}_i} \neq \emptyset)$ AND (RDV == None) **then**
9: RDV $\leftarrow$ Compute RDV based on $\overline{\mathcal{T}_i}$
10: $p_i \leftarrow p_i \oplus_{\text{end}}$ RDV
11: $\overline{\mathcal{T}_i} \leftarrow \emptyset$
12: $\mathbf{x}_i, p_i \leftarrow$ Move along path

THEOREM 2 (CONVERGENCE OF RDV-CBBA). *Let us assume that every agents are notified of the same task set $\mathcal{J}_{app}(t)$ at each timestep t. The Myopic RDV-CBBA allocates every task set sequence $(\mathcal{J}^t)_{t\geq 0}$ while guaranteeing a conflict-free solution. The Proactive RDV-CBBA guarantees the same properties, provided that agents independently compute the same rendezvous point.*

PROOF. For the *Myopic* version, the proof follows by induction. At $t = 0$, agents are connected and obtain a conflict-free allocation and a RDV location via GCBBA. Since the RDV is appended to the end of the path, agents are guaranteed to meet again. When the RDV is reached, the connectivity is restored, and the process repeats for any accumulated tasks.

For the *Proactive* version, the logic is similar but relies on the computation of a unique RDV point. It is necessary that all agents use a deterministic algorithm to identify rendezvous on their available tasks $\overline{\mathcal{T}_i}, \forall i \in \mathcal{I}$. If this holds, they meet at the intended location and proceed to the conflict-free allocation phase (Line 5). Else, they would wait indefinitely for neighbors at their respective RDV. □

It is important to note that in our study, at any timestep t, the mission center notifies **every agents** about the new tasks $\mathcal{J}_{\text{app}}(t)$. It implies that agents share at any t exactly the same available tasks $(\overline{\mathcal{T}_i})_{i\in\mathcal{I}}$, making these variables implicitly synchronized. If this does not hold, for example due to communication failure (lost message from the mission center, delayed messages ...), this impacts the two RDV procedures as follows. (1) Myopic RDV-CBBA: if at t, at least one agent receives the set $\mathcal{J}_{\text{app}}(t)$, then this set is shared among all agents at the rendezvous point (Line 5 of Algorithm 3), and planning is done as if all agents have received it. Theorem 2 is still valid for the myopic version. (2) Proactive RDV-CBBA: if at t, at least one agent does not receive $\mathcal{J}_{\text{app}}(t)$ when no RDV is yet selected, then agents will plan on at least two different RDV (Lines 9-10 of Algorithm 4). Theorem 2 fails for the proactive version. The **Myopic RDV-CBBA is hence theoretically much more robust** to communication failure.

5.6 Methods for Selecting Rendezvous

In this section, we present three distinct strategies for selecting rendezvous points. While the formulations below minimize travel distance, they can be trivially adapted to minimize travel time by incorporating agent velocities.

Depot Rendezvous: A trivial strategy is to designate the initial depot as the rendezvous point. While this approach requires no computation, it lacks adaptability to the spatial distribution of tasks and agents, likely inefficient if tasks appear far from the depot.

Barycenter Rendezvous: A geometric approach is to compute the barycenter (centroid) of a set of points. Let $\mathcal{S}$ be a set of positions of interest. In our application, $\mathcal{S}$ represents either the locations of the final tasks in the agents' current paths (Myopic) or the locations of a newly arrived task batch (Proactive). Identifying tasks with their spatial coordinates, the barycenter B is given by $B = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} j$. It satisfies $B = \arg\min_{P\in\mathbb{R}^2} \sum_{j\in\mathcal{S}} ||j - P||^2$.

Fermat-Weber Rendezvous: The point that minimizes the sum of Euclidean distances to a set of points is the solution to the Fermat-Weber problem: $\min_{P\in\mathbb{R}^2} \sum_{j\in\mathcal{S}} w_j \cdot ||j - P||$, where w_j is a weight associated with point j. The solution is unique provided the points in $\mathcal{S}$ are not collinear. The Weiszfeld algorithm provides a robust approximation given an initial guess and a fixed number of iterations [3]. This solution is denoted FW. While the FW point theoretically

aligns with the Min-Sum objective, it does not guarantee global optimality for the combined allocation-routing problem. Indeed, simultaneously optimizing task allocation and rendezvous location is a much harder problem than solving them sequentially [10].

6 EXPERIMENTS

6.1 Description

The mission scenarios we consider take place in a rectangular domain defined by $[-20, 20] \times [-20, 20]$. Each agent $i \in \mathcal{I}$ has a uniform travel speed v_i DU/TU (Distance Unit / Time Unit), drawn from uniform distribution:

$$v_i \sim \mathcal{U}([5, 20]), \quad \forall i \in \mathcal{I}. \tag{5}$$

All agents initiate the mission from the depot at $(0, 0)$. We evaluate fleet sizes of $|\mathcal{I}| \in \{3, 4, 5\}$ and communication ranges of $r \in \{1, 3, 5\}$ DU. Between two timesteps t and $t + 1$, agents receive a new batch of tasks $\mathcal{J}_{\text{app}}(t)$ and may travel a maximum of v_i DU along their paths. In our study, 10 tasks are generated every 5 timesteps, for a total of 50 tasks. Hence, including the initial batch at $t = 0$, all tasks have arrived at $t = 20$.

Task positions are generated using two distinct processes. The first, denoted $\mathcal{D}_U$, uses a Uniform distribution (6). The second, denoted $\mathcal{D}_N$, uses a bivariate Normal distribution (7) to simulate an area of high task density.

$$j \sim \mathcal{U}_2([-20, 20]^2), \qquad \forall j \in \mathcal{J}, \tag{6}$$

$$j \sim \mathcal{N}_2\left(\begin{bmatrix} -10 \\ 10 \end{bmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}\right), \qquad \forall j \in \mathcal{J}. \tag{7}$$

Tasks generated by (7) are clipped to the mission area boundaries. The performance metrics evaluated are the Min-Sum objective (total travel time) and the number of conflicting task assignments, averaged over 100 random scenarios.

6.2 Comparing Rendezvous Methods

We refer to the RDV-CBBA variants using Depot, Barycenter, and Fermat-Weber rendezvous points as Depot, Bary, and FW, respectively. We distinguish the Myopic and Proactive versions with the suffixes (M) and (P) (e.g., FW(M) and FW(P)). For these experiments, the communication range is fixed at $r = 1$ DU. The FW rendezvous location is calculated using the Weiszfeld Algorithm [3] initialized at the barycenter, with 1000 iterations. For the Myopic version (FW(M)), the weight associated with each path-ending task $j_i^{\text{last}} \in \mathcal{J}_{\text{last}}$ is $w_i = 1/v_i$ (prioritizing slower agents). For the Proactive version (FW(P)), all weights are equal to 1. Table 1 presents the average Min-Sum values for both task distributions.

Algo	Fleet size	Min-Sum ($\mathcal{D}_U$)	Min-Sum ($\mathcal{D}_N$)
Depot(M)	3 / 5 / 10	61.0 / 69.7 / 68.3	55.8 / 58.7 / 59.2
Bary(M)	3 / 5 / 10	58.4 / 69.6 / 74.8	46.7 / 54.4 / 59.4
FW(M)	3 / 5 / 10	**57.1 / 69.3 / 69.8**	**46.1 / 53.1 / 57.6**
Bary(P)	3 / 5 / 10	**55.8** / **65.0** / **73.5**	**45.2** / **50.6** / **57.6**
FW(P)	3 / 5 / 10	56.2 / 65.3 / 74.6	45.2 / 50.8 / 60.3

Table 1: Min-Sum comparison of RDV strategies.

The results indicate comparable performance across all rendezvous methods when tasks are uniformly distributed, with FW(M) and Bary(P) performing best by a narrow margin. The Depot rendezvous also performs reasonably well in this case because the center of the area coincides with the expected value of the uniform distribution.

For tasks drawn from $\mathcal{D}_N$, the Depot rendezvous is ill-suited, as it forces agents to return to the center of the arena rather than remaining in the zone of high task density. Conversely, FW(M) and Bary(P) adapt better, placing the rendezvous within this zone.

FW(M) maintains a competitive performance with Bary(P) as well as superior robustness in practical communication settings (see Section 5.6). We reference it simply as RDV-CBBA and use it as the baseline for the subsequent comparative experiments.

6.3 Comparison with Dynamic Task Allocation Baselines

In this section, we compare the performance of RDV-CBBA against three state-of-the-art decentralized methods adapted for dynamic scenarios using Algorithm 1: GCBBA [17], the PI heuristic [30], and DMCHBA [27]. PI is an extension of CBBA designed for complex objective functions and exploration, while DMCHBA utilizes a distributed Hungarian method followed by a TSP routing phase. For GCBBA and PI, we use the bidding scheme defined in Equation (4) and a greedy insertion heuristic for path creation.

Figures 3 and 4 present the Min-Sum values and the number of task conflicts, respectively, for tasks generated via $\mathcal{D}_U$. We also include a lower bound labeled **UR-GCBBA** (Unlimited Range GCBBA), representing the performance of GCBBA in a fully connected network where conflict-free allocation is guaranteed.

Min-Sum Performance: When the communication range is strictly limited ($r = 1$), the dynamic algorithms GCBBA, PI and DMCHBA perform poorly, with performance deteriorating as the fleet size increases due to high task redundancy (multiple agents completing the same task). As the communication range increases, their performance improves, approaching that of RDV-CBBA. However, for $|\mathcal{I}| = 10$ and $r = 5$, they remain approximately 85% worse than RDV-CBBA. Although DMCHBA performs well in static, connected scenarios [27], it is severely impacted by communication constraints in the dynamic setting. This performance drop aligns with findings by Cao et al. [4] for a similar Hungarian-based approache under non-ideal communication. GCBBA and PI exhibit comparable performance, likely due to the use of the same DMG bidding scheme. RDV-CBBA demonstrates stable performance that is largely independent of r, as it relies on physical rendezvous rather than continuous communication. However, UR-CBBA still outperforms RDV-CBBA by at least 50% for 10 agents, indicating that while RDV-CBBA effectively mitigates conflicts, there is still a traveling cost for achieving synchronization at the RDV compared to perfect instantaneous communication.

Conflict Analysis: The number of conflicts for the dynamic GCBBA, PI and DMCHBA decreases as r increases but remains significant even at $r = 5$ (approximately 70 conflicts for 10 agents). DMCHBA produces fewer conflicts than GCBBA and PI, a trend also noted in [4]. Crucially, as guaranteed by Theorem 2, RDV-CBBA provides

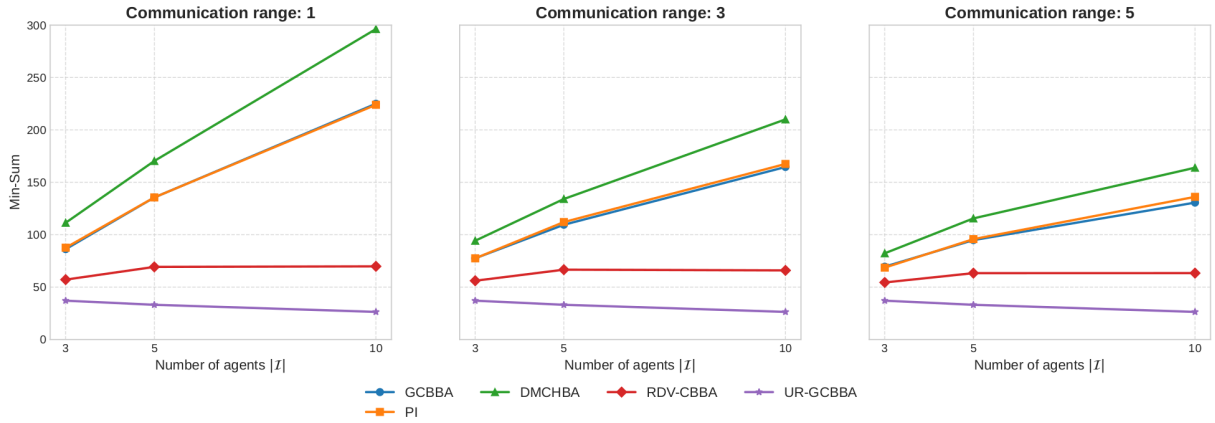


Figure 3: Min-Sum comparison of dynamic CBBA, PI, DMCHBA and RDV-CBBA.

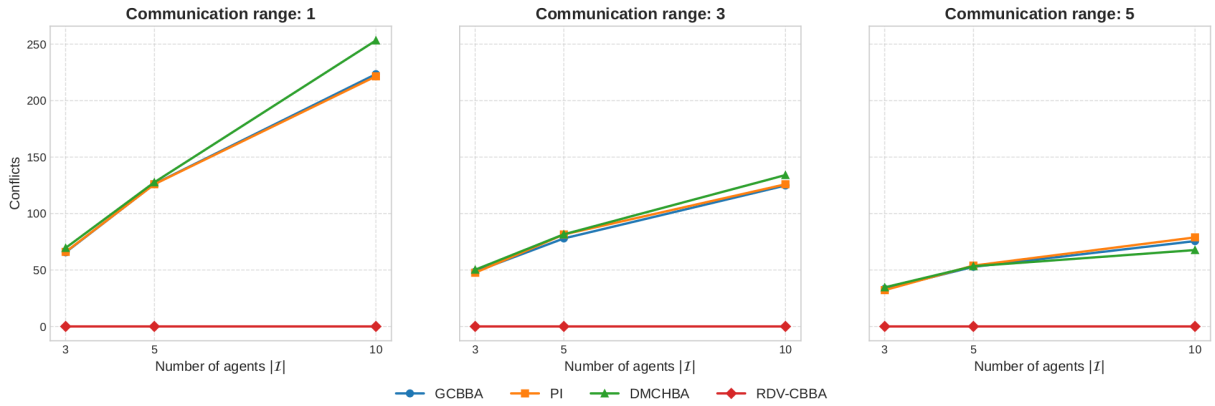


Figure 4: Number of conflicting task assignments for dynamic CBBA, PI, DMCHBA and RDV-CBBA.

zero-conflict allocations whereas DMCHBA may return allocation with conflict. RDV-CBBA thus prevents the agents from executing the same tasks several times unnecessarily.

7 CONCLUSION

In this paper, we addressed the problem of decentralized task allocation in dynamic environments where agents are constrained by limited communication ranges. This setting violates the fundamental assumption of network connectivity required for the convergence of classical decentralized algorithms.

First, we proposed a general framework to extend classical decentralized algorithms to dynamic scenarios. This method employs a "lazy" allocation strategy: agents bid only on newly appearing tasks and trigger reallocation only when conflicts are explicitly detected with neighbors. While this approach ensures finite-time completion of tasks, it is inherently *conflict-prone*. In low-communication settings, the lack of coordination opportunities leads to high task redundancy and suboptimal performance.

To overcome this limitation, we introduced **RDV-CBBA**, a novel approach that enforces periodic synchronization at computed rendezvous points to guarantee conflict-free allocations. We developed two variants: a *Myopic* strategy, where the rendezvous is derived from the current path endpoints, and a *Proactive* strategy, which anticipates future RDV based on new task clusters. Our analysis reveals a critical trade-off: while the Proactive strategy offers marginal performance gains, it is brittle in the presence of inconsistent information (e.g., packet loss). The Myopic strategy, conversely, is robust as it relies on consensus formed during physical meetings, making it the preferable choice for realistic deployments.

Experiments indicate that RDV-CBBA significantly outperforms state-of-the-art dynamic extensions of CBBA, PI, and DMCHBA in terms of the Min-Sum objective and conflict avoidance across all tested communication ranges. However, a performance gap remains compared to the theoretical lower bound of unlimited communication (UR-GCBBA), indicating potential for further optimization.

In future work, we aim to evaluate RDV-CBBA in practical simulations with non-ideal communication layers (lost messages, delays). Moreover, our RDV methods assume that all agents eventually meet at an intended RDV. Additional experiments could be to introduce agent failures. Furthermore, we plan to investigate coupled optimization approaches that solve the task allocation and rendezvous location problems simultaneously, rather than sequentially, to further close the optimality gap.

REFERENCES

[1] Xiaoshan Bai, Weisheng Yan, Ming Cao, and Jie Huang. 2017. Target assignment for robots constrained by limited communication range. *arXiv preprint arXiv:1702.04700* (2017).

[2] Xiaoshan Bai, Weisheng Yan, and Shuzhi Sam Ge. 2021. Distributed task assignment for multiple robots under limited communication range. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, 7 (2021), 4259–4271.

[3] Amir Beck and Shoham Sabach. 2015. Weiszfeld's Method: Old and New Results. *Journal of Optimization Theory and Applications* 164, 1 (2015), 1–40. https://ideas.repec.org//a/spr/joptap/v164y2015i1d10.1007_s10957-014-0586-7.html Publisher: Springer.

[4] Yan Cao, Teng Long, Jingliang Sun, Zhu Wang, and Guangtong Xu. 2023. Comparison of Distributed Task Allocation Algorithms Considering Non-ideal Communication Factors for Multi-UAV Collaborative Visit Missions. *IEEE Robotics and Automation Letters* (2023), 1–8. https://doi.org/10.1109/LRA.2023.3295999 Conference Name: IEEE Robotics and Automation Letters.

[5] Jesús Cerquides, Rémi Emonet, Gauthier Picard, and Juan A Rodríguez-Aguilar. 2018. DECIMAXSUM: Using decimation to improve max-sum on cyclic DCOPs. In *Artificial Intelligence Research and Development*. IOS Press, 27–36.

[6] Zhekun Cheng, Liangyu Zhao, and Zhongjiao Shi. 2022. Decentralized Multi-UAV Path Planning Based on Two-Layer Coordinative Framework for Formation Rendezvous. *IEEE Access* 10 (01 2022), 45695–45708. https://doi.org/10.1109/ACCESS.2022.3170583

[7] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics* 25, 4 (Aug. 2009), 912–926. https://doi.org/10.1109/TRO.2009.2022423 Conference Name: IEEE Transactions on Robotics.

[8] Alysson Ribeiro da Silva and Luiz Chaimowicz. 2025. Intermittent Rendezvous Plans with Mixed Integer Linear Program for Large-Scale Multi-Robot Exploration. arXiv:2511.12237 [cs.RO] https://arxiv.org/abs/2511.12237

[9] Julian de Hoog, Stephen Cameron, and Arnoud Visser. 2010. Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments.

[10] Haggi Do, Junwoo Jang, and Jinwhan Kim. 2025. Heterogeneous multi-robot system mission planning with cooperative replenishment through data-driven rendezvous point selection. *Intelligent Service Robotics* 18, 1 (2025), 61–73.

[11] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research* 61 (March 2018), 623–698. https://doi.org/10.1613/jair.5565

[12] Victor Guillet, Christophe Grand, Charles Lesire, and Gauthier Picard. 2025. Extending Consensus-based Task Allocation Algorithms with Bid Intercession to Foster Mixed-Initiative. In *24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-25)*. International Foundation for Autonomous Agents and Multiagent Systems, Detroit (Michigan), United States, 932–940. https://doi.org/10.5555/3709347.3743612

[13] Sara Hsaini, Rabah Ammour, Leonardo Brenner, Moulay El Hassan Charaf, and Isabel Demongodin. 2023. A Decentralized Based Approach Using Hybrid Filtered Beam Search Algorithm for Monitoring Patrols. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 1436–1441.

[14] Sarah Ismail and Liang Sun. 2017. Decentralized hungarian-based approach for fast and scalable task allocation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 23–28. https://doi.org/10.1109/ICUAS.2017.7991447

[15] Luke Johnson, Han-Lim Choi, Sameera Ponda, and Jonathan P How. 2012. Allowing non-submodular score functions in distributed task allocation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 4702–4708.

[16] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan How. 2011. Asynchronous Decentralized Task Allocation for Dynamic Environments. In *Infotech@Aerospace 2011*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2011-1441 _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2011-1441.

[17] Alexandre Kha, Aurélie Beynier, Christophe Labreuche, and Mathieu Marchand. 2025. Enhancing CBBA convergence and optimality guarantees in Multiagent Task Allocation. In *hal-05218405*. https://hal.science/hal-05218405

[18] Jaekwang Kim, Hyung-Jun Park, Aditya Penumarti, and Jaejeong Shin. 2024. Fast Marching Based Rendezvous Path Planning for a Team of Heterogeneous Vehicles. *IEEE Access* PP (01 2024), 1–1. https://doi.org/10.1109/ACCESS.2024.3444314

[19] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability* 3, 71-104 (2014), 3.

[20] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[21] J. Lin, A. S. Morse, and B. D. O. Anderson. 2007. The Multi-Agent Rendezvous Problem. Part 1: The Synchronous Case. *SIAM Journal on Control and Optimization* 46, 6 (2007), 2096–2119. https://doi.org/10.1137/040620552 arXiv:https://doi.org/10.1137/040620552

[22] Matteo Luperto, Mauro Tellaroli, Michele Antonazzi, and Nicola Basilico. 2025. Multi-robot rendezvous in communication-restricted unknown environments via backtracking and semantic frontier-based exploration. *Robotics and Autonomous Systems* 194 (2025), 105137. https://doi.org/10.1016/j.robot.2025.105137

[23] Chase C Murray and Ritwik Raj. 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies* 110 (2020), 368–398.

[24] Gauthier Picard. 2021. Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions. *arXiv preprint arXiv:2106.03548* (2021).

[25] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P How, Matt Vavrina, and John Vian. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *Proceedings of the 2010 American Control Conference*. IEEE, 3998–4003.

[26] Sarvapali D. Ramchurn, Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. 2010. Decentralized Coordination in RoboCup Rescue. *Comput. J.* 53, 9 (Nov. 2010), 1447–1461. https://doi.org/10.1093/comjnl/bxq022

[27] Arezoo Samiei and Liang Sun. 2024. Distributed Matching-By-Clone Hungarian-Based Algorithm for Task Allocation of Multiagent Systems. *IEEE Transactions on Robotics* 40 (2024), 851–863. https://doi.org/10.1109/TRO.2023.3335656 Conference Name: IEEE Transactions on Robotics.

[28] Shengli Wang, Youjiang Liu, Yongtao Qiu, and Jie Zhou. 2022. Consensus-Based Decentralized Task Allocation for Multi-Agent Systems and Simultaneous Multi-Agent Tasks. *IEEE Robotics and Automation Letters* 7, 4 (Oct. 2022), 12593–12600. https://doi.org/10.1109/LRA.2022.3220155 Conference Name: IEEE Robotics and Automation Letters.

[29] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161, 1 (Jan. 2005), 55–87. https://doi.org/10.1016/j.artint.2004.10.004

[30] Wanqing Zhao, Qinggang Meng, and Paul W. H. Chung. 2016. A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario. *IEEE Transactions on Cybernetics* 46, 4 (April 2016), 902–915. https://doi.org/10.1109/TCYB.2015.2418052 Conference Name: IEEE Transactions on Cybernetics.