2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

# Decentralized Task Allocation in Multi-Agent Systems Using a Decentralized Genetic Algorithm

Ruchir Patel[1], Eliot Rudnick-Cohen[1], Shapour Azarm[1], Michael Otte[2], Huan Xu[2,3], Jeffrey W. Herrmann[1,3]

*Abstract*—**In multi-agent collaborative search missions, task allocation is required to determine which agents will perform which tasks. We propose a new approach for decentralized task allocation based on a decentralized genetic algorithm (GA). The approach parallelizes a genetic algorithm across the team of agents, making efficient use of their computational resources. In the proposed approach, the agents continuously search for and share better solutions during task execution. We conducted simulation experiments to compare the decentralized GA approach and several existing approaches. Two objectives were considered: a min-sum objective (minimizing the total distance traveled by all agents) and a min-time objective (minimizing the time to visit all locations of interest). The results showed that the decentralized GA approach yielded task allocations that were better on the min-time objective than those created by existing approaches and solutions that were reasonable on the min-sum objective. The decentralized GA improved min-time performance by an average of 5.6% on the larger instances. The results indicate that decentralized evolutionary approaches have a strong potential for solving the decentralized task allocation problem.**

## I. Introduction

Coordinating the behaviors of autonomous agents in multi-agent systems typically involves allocating tasks across a multi-agent team. For example, if the tasks are locations that the agents need to visit, the task allocation specifies, for each agent, a sequence of locations that the agent will visit.

In a centralized system, a central planner with global knowledge allocates the tasks for all agents as needed. This approach not only fails to utilize the agents' computing resources but also leaves the system vulnerable: if the communication link between the central planner and the agents fails, then agents may not receive their assigned tasks, and those tasks will never be completed. In a decentralized system, however, each agent leverages its computing resources to determine its own task allocation, and the agents do not depend upon a single central planner. They communicate with one another to deconflict or improve their allocations.

The task allocation problem considered herein is related to the multiple traveling salesman problem (mTSP) and the vehicle routing problem (VRP), for which evolutionary algorithms (such as genetic algorithms) have been used to generate high quality solutions [1, 2]. Because they use a population of candidate solutions, evolutionary algorithms have an inherent parallelism that is compatible with decentralized task allocation. Moreover, they can search the entire space of solutions and continuously seek better solutions, unlike current decentralized task allocation approaches [3-15], which can stop at suboptimal solutions.

Although decentralized task allocation approaches have been proposed previously [3-15], we are not aware of any that have attempted to parallelize evolutionary algorithms over a team of agents. To address this gap, we developed a new decentralized evolutionary approach that utilizes a genetic algorithm (GA) that runs continuously and exchanges complete solutions (sets of task sequences) between agents. This enables the approach to continuously improve its current allocation of tasks during execution and potentially achieve globally optimal allocations for the team. We conducted experiments that compared the decentralized GA to other state-of-the-art methods, and these show that this approach often performed better than these existing methods when the objective is minimizing the time to complete all tasks.

The rest of this paper is organized as follows: Section II discusses previous work related to decentralized task allocation. Section III describes the problem formulation. Section IV discusses the solution approach that we considered. Section V details the experimental setup including the problem instances. Section VI presents and discusses the results. Section VII summarizes and concludes this paper.

## II. Related Work

Decentralized task allocation problems can be classified into market-based approaches and optimization-based approaches [3, 4]. In market-based approaches, auctioneers decide how tasks are allocated based on bids from other agents. In decentralized auction approaches, every agent acts as an auctioneer and uses bids from all other agents to auction tasks. Khamis et al. [3] noted that these approaches have advantages in scalability and adaptability.

Decentralized auction approaches have been proposed for decentralized task allocation. For example, the Consensus-Based Auction Algorithm (CBAA) and Consensus-Based Bundle Algorithm (CBBA) [5] attempt to maximize reward (minimize cost) over the entire system. Each algorithm iterates between an auction phase, which creates a task allocation using local information, and a consensus phase, which processes information from other agents. Extensions to the CBBA include the Asynchronous CBBA (ACBBA) [6]

[1]Department of Mechanical Engineering, University of Maryland, College Park, MD 20742
[2]Department of Aerospace Engineering, University of Maryland, College Park, MD 20742
[3]Institute for Systems Research, University of Maryland, College Park, MD 20742
Email: {rpatel18, erudnick, azarm, otte, mumu, jwh2}@umd.edu

and the Performance Impact Algorithm (PIA) [7], which modifies the auction and consensus phases to better optimize the global system objective. The Hybrid Information and Plan Consensus (HIPC) approach [8] uses a centralized assignment method to generate better task allocations in the auction phase by predicting what tasks other agents should complete. Nanjananth and Gini [9] proposed a parallel repeated auctions approach where every agent is an auctioneer and bidder in parallel allowing for more adaptability than the typical consensus-based auction approaches. These heuristic approaches construct a solution but do not search for better solutions.

Optimization-based approaches utilize distributed constraint optimization, game theory, metaheuristics, or other optimization techniques [10]. Because the task allocation problem is a variant of the mTSP, which is NP-hard [11], the optimization-based approaches' computational effort may, as instance size increases, grow more quickly than that of auction approaches. The Decentralized Hungarian (DH) algorithm [12] is similar to the CBAA but replaces the auction step with solving a task allocation problem via the Hungarian method [13]. Evolutionary algorithms, particularly GAs, are often used to find solutions to the VRP and mTSP [1, 2], but these typically centralized approaches cannot be used for decentralized task allocation. For decentralized task allocation, Choi and Kim [14] proposed a two-stage GA-based approach in which each agent first determines its own task sequence using a GA and then communicates with other agents to exchange tasks if that reduces costs. Ping-An et al. [15] proposed an approach in which each agent uses a GA to generate task clusters that are used to improve initial allocations generated from an auction-based approach. Although these approaches use a GA, they do not exploit its parallelism and its ability to continue searching for better solutions as tasks are completed. Decentralized GAs [16] allow for parallelizing the operations of a GA, which have been applied to problems such as scheduling [17, 18].

In the decentralized, parallelized GA proposed here (described in Section IV), each agent maintains its own population and exchanges solutions instead of running an independent GA. Thus, the decentralized GA coordinates the computing resources of every agent to search for a complete solution in a novel way.

## III. Problem Definition

The decentralized task allocation problem can be formulated as follows: A task is a location of interest (in a two-dimensional space) that some agent must visit. Given a set of $n$ tasks and a set of $m$ agents, a solution specifies task sequences for the agents such that every task is completed by an agent. The objective function is either the total distance traveled by all agents (min-sum) or the time at which the last location is visited (min-time). (These objectives are relevant because they correspond to minimizing fuel consumption and mission completion time.) The system is decentralized: there is no centralized planner, and the agents make their own task allocation decision. Agents start from different positions. Every agent initially knows all of the locations of interest but none of the locations of the other agents in the system. Each agent can communicate with all other agents in the system. Agents all travel with the same constant speed when executing tasks. The mission terminates when all locations of interest have been visited. This problem is equivalent to the mTSP where cities are locations of interest and salesmen are vehicles.

More precisely, let $\boldsymbol{x_i} = (x_{i1}, \ldots, x_{in(i)})$ be a task sequence for agent $i$ and let $n(i)$ be the number of tasks in that sequence. That is, the $j$-th task in the task sequence is task $x_{ij}$, where $x_{ij}$ is an element of $\{1, \ldots, n\}$. Let $\boldsymbol{x} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_m}\}$ be the set of task sequences for all of the agents.

Let $W$ be the two-dimensional workspace in which tasks are located. Let $d(p_a, p_b)$ be the distance between any two points $p_a$ and $p_b$ in $W$, and, for convenience, let $d(x_{ij}, x_{ik})$ be the distance between the locations of the tasks $x_{ij}$ and $x_{ik}$. Let $q_1, \ldots, q_m$ be the initial locations of the agents. Let $v$ be the speed of the agents.

Let $c_d(x_i)$ be the total distance traveled by agent $i$ to complete the task sequence defined by $\boldsymbol{x_i}$:

$$c_d(\boldsymbol{x_i}) = d(q_i, x_{i1}) + \sum_{j=2}^{n(i)} d(x_{i,j-1}, x_{ij}). \quad (1)$$

Let $c(\boldsymbol{x})$ be the cost function. We considered two cost functions. The first, the min-sum objective, is the sum of the distance traveled by the agents:

$$c(\boldsymbol{x}) = \sum_{i=1}^{m} c_d(\boldsymbol{x_i}). \quad (2)$$

The second, the min-time objective, is the time needed to complete all of the tasks, which equals the time that the last task is completed:

$$c(\boldsymbol{x}) = \max\{c_d(\boldsymbol{x_1}), \ldots, c_d(\boldsymbol{x_m})\} / v. \quad (3)$$

## IV. Decentralized GA

### A. Preliminaries

The decentralized GA uses a min-max nearest neighbors algorithm to construct some solutions. This algorithm is a constructive heuristic that generates a complete set of task sequences for all agents and allocates all tasks present. It iteratively constructs task sequences by appending the lowest cost tasks to the agents' task sequences until all tasks are allocated. The cost of a task for a particular agent equals the distance of the agent's task sequence after appending the task to the sequence. This algorithm can construct high quality task allocations on the min-max objective.

### B. Approach

Our decentralized GA is a new approach that parallelizes a GA for solving a task allocation problem across the entire team of agents and continually improves the allocations as the agents execute tasks in real time. Each agent maintains and improves a population of solutions; coordinating these populations by sharing solutions across the multi-agent system is an innovative feature of our approach. In each agent's population, a solution is a set of task sequences (not

merely the task sequence for one agent). These sequences allocate all of the tasks not yet completed. Each agent periodically shares its current best solution with the other agents and incorporates any solutions it receives into its population. Because the agents exchange high-quality solutions, they are collectively solving the task allocation problem and reaching consensus while using multiple populations and fully exploiting the parallelism in the GA. Moreover, by using a separate thread for the GA, each agent can continue the search for better solutions while the agent performs a task.

In this decentralized approach, every agent has two threads: the first runs a GA, and the second is the Task Sequence Execution thread. This structure is shown in Fig. 1. The first thread solves the task allocation problem using a modified implementation of an existing GA [19] that has been used to solve the mTSP. This evolutionary approach evolves (improves) a population of solutions over multiple iterations. In each iteration, it selects the highest quality solutions in the current population and then mutates them to generate a new population of solutions. Each solution is represented as a sequence of tasks and a set of breakpoints that divide the sequence into sequences for each agent. Every agent must have at least one task. The initial population includes a solution constructed by using the min-max nearest neighbors algorithm. The GA has no iteration limit; it runs until the mission is complete. Completed tasks are removed from the solutions as needed.

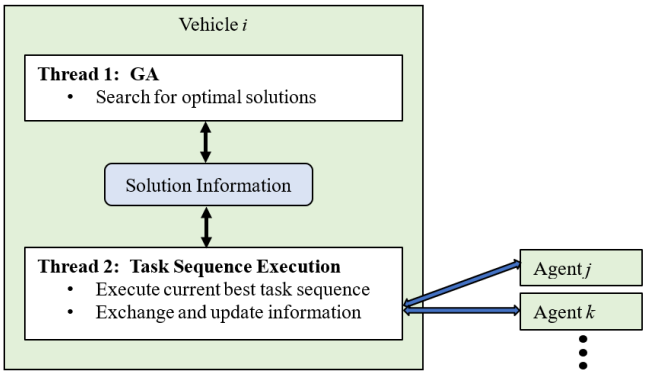



Fig. 1. Basic structure of the decentralized GA, which runs two threads that interact to generate and update the agent's task sequence.

The pseudocode for the Task Sequence Execution thread is provided in Algorithm 1. This thread periodically queries the GA thread for its current best solution and uses that solution to determine the current task sequence for the agent to execute regardless the agent's current task sequence. The Task Sequence Execution thread also exchanges with the other agents information such as tasks that have been completed, the agent's current location, and the agent's current best solution. This thread shares any solutions received with the GA thread, which incorporates them into its population. When the number of remaining tasks (locations) becomes less than the number of agents, this thread uses the min-max nearest neighbors algorithm to construct a task sequence for the agent (instead of using the GA's solution, which has at least one task for every agent).

**Algorithm 1:** Pseudocode for task sequence execution run in a separate thread on agent *i*.

```
1:  function TASK_SEQUENCE_EXECUTION()
2:  while mission not complete:
3:    if num_agents > num_locations:
4:      current_solution ← solution from nearest neighbors
5:    else:
6:      current_solution ← current best solution from GA
7:    end if
8:    task_sequence ← current_solution{i}
9:    execute(task_sequence)
10:   send information to all other agents
11:   receive information from whoever responds
12:   store received solutions for incorporation into GA
13: end while
14: end function
```

## V. Experimental Setup

We compared the decentralized GA against other decentralized task allocation approaches on problem instances of different sizes. Each case was run for 20 trials to account for variation in the simulation and the stochastic behavior of the GA approach. The methods were implemented in Python and simulated in real time using Robot Operating System (ROS) Kinetic [20]. Agents were simulated in separate processes, similar to how a decentralized system would operate. Each agent has a communication interface written in C++ that allows them to broadcast messages to all other agents in the system. Further details of the simulation environment can be found in [21]. The speed of each agent was set to 5 meters per second. The frequency at which every approach was called was set to 0.01 seconds. Simulations were run on an AMD Ryzen 7 2700 Eight-Core 3.20 GHz Processor with 16.0 GB of RAM. Simulations were terminated when all tasks were completed.

We implemented three variants of the GA, each with a different cost function. The cost function is used to determine a solution's fitness (quality). In the first variant (GA-ms), the cost function was the min-sum objective (the total distance). In the second variant (GA-mm), the cost function was the min-max objective (the maximum distance traveled by any agent). In the third variant (GA-multi), the cost function was a weighted combination of the min-max objective and the min-sum objective where $w$ is the weight on the min-sum objective:

$$c(\boldsymbol{x}) = \max_i c_d(\boldsymbol{x}_i) + w \sum_{i=1}^{m} c_d(\boldsymbol{x}_i). \quad (4)$$

The weight $w$ for the GA-multi approach was set to a value of 0.01 which was found to work best in preliminary testing.

### A. Problem Instances

Our tests used five problem instances. In each one, the locations of interest were randomly selected from a uniform distribution over a 100 meter by 100 meter region. The sizes

($n$ x $m$) of the instances were 10 x 5, 20 x 4, 30 x 3, 35 x 5, and 40 x 6. The instances of size 30 x 3 and 35 x 5 are shown in Fig. 2.

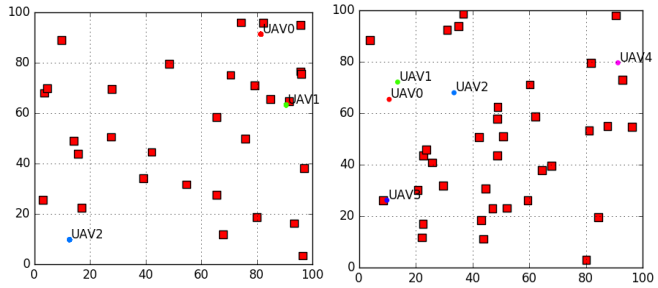



Fig. 2. Example problem instances of size ($n$ x $m$) (a) 30 x 3 and (b) 35 x 5. The agents are the circles that are labelled UAV# and the locations of interest are the red squares. Axes indicate position in meters.

### B. Decentralized Task Allocation Approaches

We ran multiple decentralized task allocation approaches as benchmarks in our experiments. The basic structure for these approaches is shown in Fig. 3 where only a Task Sequence Execution thread is run. The execution thread runs continuously and repeatedly iterates between two phases. Phase 1 constructs a task sequence for the agent when no sequence exists (e.g., its tasks have been completed by itself or other agents). Phase 2 shares information with the other agents and removes any tasks that agents have completed.

We used existing optimization-based and auction-based methods as well as a greedy nearest neighbors (NN) approach to generate benchmark solutions. In some cases, we implemented variants that use different cost functions when determining which task to add to a task sequence.

*1. Greedy Nearest Neighbors (NN)*

The greedy NN method constructs a task sequence for an agent by assigning itself the lowest cost task that has not been completed. The cost of a task equals the distance from the agent's current location to the location to be visited.

*2. Decentralized Hungarian (DH)*

The DH approach [9] uses a cost matrix in which each entry is the cost of allocating one task to one agent. This method assigns only a single task at a time to the agent. To construct a task sequence (Phase 1), the agent runs the Hungarian algorithm [13] on the cost matrix to obtain the optimal one-to-one task allocation for the system. In Phase 2, the agent exchanges cost matrices with all other agents and then updates its cost matrix. We implemented two variants of this approach. In the first variant (DH-ms), the cost function is the distance between the agent and the location to visit. In the second variant (DH-mm), the cost function equals this distance plus a penalty that equals how far the agent has already traveled.

*3. Consensus-based Auction Algorithm (CBAA)*

We implemented two variants of the CBAA [5]. In the first variant (CBAA-ms), the cost function is the distance between the agent and the location to visit. In the second variant (CBAA-mm), the cost function equals this distance plus a penalty that equals how far the agent has already traveled.

*4. Asynchronous Consensus-based Bundle Algorithm (ACBBA)*

The ACBBA [6] assigns a bundle of tasks instead of a single one. Because multiple tasks are being allocated, an agent's bids depend on the tasks that are already in its task sequence. We implemented two variants of this approach. In the first variant (ACBBA-ms), the cost function is the distance between the previous task in the bundle and the current task. In the second variant (ACBBA-mm), the cost function equals the distance that the agent will travel from the start of the mission to the current task.

*5. Performance Impact Algorithm (PIA)*

The PIA [7] modifies the CBBA to utilize a different kind of bid evaluation called "significance." It also makes improvements in how conflict resolution is achieved in the consensus phase. Our implementation used the ACBBA consensus rules. For this method, only the min-sum cost function was implemented.

Preliminary testing showed that the ACBBA and PIA variants performed best with a bundle size of at most five and were limited to five iterations. The iteration limits for the DH and CBAA approaches were set to two.

*6. Hybrid Information and Plan Consensus (HIPC)*

In the HIPC approach [8], each agent solves the task allocation problem for the entire system to determine its own task sequences. Our implementation used a min-max nearest neighbors algorithm (see Section IV.A). As with the PIA, this approach was modified to use the ACBBA consensus rules.

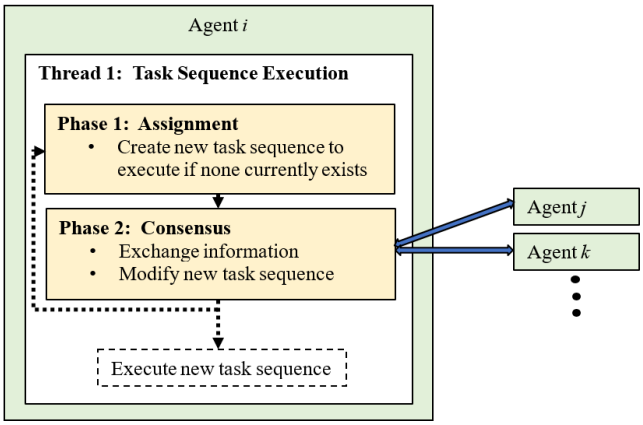



Fig. 3. In the considered decentralized task allocation approaches, the Task Sequence Execution thread first creates a task sequence and then removes completed tasks from it during execution.

## VI. Results

### A. Total Time and Distance Performance

Both total time and total distance performance metrics were considered. These metrics were evaluated for all methods and variants to analyze which variants performed best on each metric. Table I reports the average time to visit all locations of interest when using each approach on each of the five instances. Time was recorded from the start of the trial until

all locations of interest were visited. Total distance is the sum of every agent's distance travelled during the mission.

As seen in Table I, the variants of the decentralized GA approach yielded the lowest total time solutions for four of the five instances. On the 10 x 5 instance, the DH-ms and DH-mm approaches outperformed the other approaches because the small number of tasks per agent makes it easy to find a high-quality solution, but the GA-ms variant performed nearly as well. On the 20 x 4 instance, the GA-multi approach had the best average time of all methods; the GA-mm variant performed nearly as well. On the 30 x 3 instance, the GA variants outperformed all other methods on the time metric. On the 35 x 5 instance, the GA-ms and GA-multi variants outperformed all other methods by a significant margin. Finally, on the 40 x 6 instance, the GA-multi approach outperformed the other methods. The other variants of the proposed GA approach also outperformed most of the other approaches on this instance with the exception of the DH variants which had similar performance. These results show the GA variants, particularly the GA-multi approach, often outperformed the other methods on the time to complete all tasks on instances of larger size.

Table II reports the average total distance traveled by all agents when using each approach on each of the five instances.

For the total distance metric, the variants of the decentralized GA approach did not perform as well as the best decentralized task allocation approaches. The HIPC approach achieved the best total distance on the 10 x 5, 20 x 4, and 35 x 5 instances. On the 20 x 4 instance, the GA-multi's performance was nearly the same as the HIPC approach. On the 30 x 3 instance, the ACBBA variants outperformed the other methods; the GA-multi's performance was 3.6% greater (worse). On the 35 x 5 instance, the HIPC approach had the best performance; the GA-ms's performance was 4.0% greater. The GA-ms approach outperformed most approaches on the 40 x 6 instance with the exception of the PIA and HIPC approaches. The GA-ms's performance was 6.4% greater than the PIA on this instance. As the instance size increased, the GA-ms variant performed better than the other decentralized GA variants because it used the total distance as the fitness function. Also, as the instance size increased, the decentralized GA variants performed better relative to the other approaches. This indicates that the proposed approach has good scalability.

Although the decentralized GA approach yielded task allocations that were better on the min-time objective than the solutions that the other approaches created, the allocations that it found were not superior on the min-sum objective.
It may be that the min-time objective is harder for the other approaches to optimize as it requires knowledge of other agent's task sequences, which is not an obstacle for the decentralized GA approach, in which every agent considers complete solutions. The min-sum objective may be easier for the other approaches, especially when the instance is not large, so the decentralized GA approach is unable to find better solutions on the smaller instances. Moreover, the constraint that every agent must be assigned at least one task may prevent the GA from finding solutions that have better total distance because, in some cases, using fewer agents can reduce total distance.

Table I. Time (in seconds) to visit all locations of interest averaged over 20 trials for each approach on each instance. The boldface values are the best average for each instance.

| | Instance | | | | |
|---|---|---|---|---|---|
| Approach | 10 x 5 | 20 x 4 | 30 x 3 | 35 x 5 | 40 x 6 |
| NN | 19.6 | 45.5 | 41.6 | 33.4 | 26.2 |
| CBAA-ms | 14.7 | 28.8 | 41.5 | 30.8 | 26.0 |
| CBAA-mm | 14.7 | 28.8 | 41.4 | 27.6 | 26.0 |
| DH-ms | **11.6** | 29.6 | 42.1 | 25.8 | 23.5 |
| DH-mm | **11.6** | 29.6 | 42.7 | 25.9 | 23.3 |
| ACBBA-ms | 15.1 | 24.1 | 34.4 | 27.6 | 26.8 |
| ACBBA-mm | 14.6 | 25.5 | 34.4 | 24.6 | 25.8 |
| PIA | 19.2 | 31.3 | 45.1 | 34.6 | 31.8 |
| HIPC | 11.8 | 27.9 | 34.4 | 24.0 | 24.7 |
| GA-ms | 12.0 | 26.0 | 33.9 | **21.8** | 23.6 |
| GA-mm | 12.5 | 23.3 | 33.8 | 24.2 | 23.3 |
| GA-multi | 14.1 | **22.5** | **33.3** | 22.3 | **22.5** |

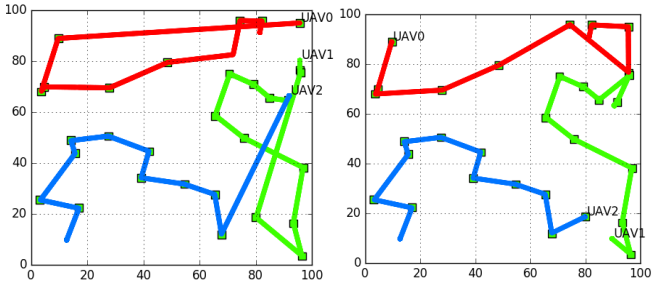
Table II. Total distance traveled by all agents averaged over 20 trials for each approach on each instance. The boldface values are the best average for each instance.

| | Instance | | | | |
|---|---|---|---|---|---|
| Approach | 10 x 5 | 20 x 4 | 30 x 3 | 35 x 5 | 40 x 6 |
| NN | 486.3 | 903.8 | 603.3 | 826.8 | 773.2 |
| CBAA-ms | 301.2 | 517.6 | 606.7 | 689.0 | 719.4 |
| CBAA-mm | 301.5 | 519.8 | 602.2 | 625.2 | 709.2 |
| DH-ms | 256.8 | 517.8 | 615.1 | 584.2 | 649.6 |
| DH-mm | 256.2 | 516.6 | 626.7 | 585.1 | 641.1 |
| ACBBA-ms | 327.8 | 404.3 | 441.1 | 624.8 | 690.8 |
| ACBBA-mm | 334.5 | 428.8 | **439.7** | 520.8 | 683.0 |
| PIA | 209.8 | 482.3 | 508.1 | 481.5 | **563.9** |
| HIPC | **198.1** | **401.6** | 445.0 | **443.0** | 590.0 |
| GA-ms | 222.7 | 448.9 | 456.6 | 460.5 | 600.0 |
| GA-mm | 268.1 | 429.6 | 468.8 | 553.9 | 636.7 |
| GA-multi | 245.6 | 404.6 | 455.8 | 494.9 | 604.0 |

### *B. Example Task Sequences*

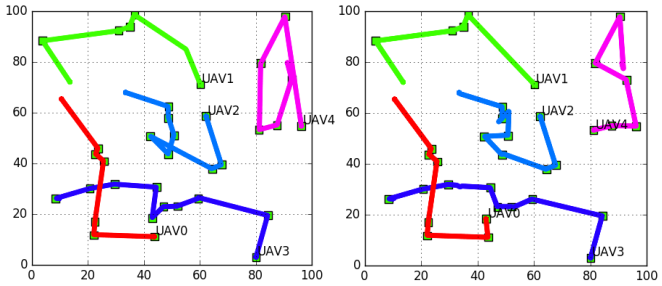
Figs. 4(a) and 4(b) present the agents' executed task sequences (paths) in the 30 x 3 instance using the DH-mm and GA-multi approaches. For this instance, the GA-multi approach produced task sequences that are better (on both the min-sum and min-time objectives) than those that the DH-mm approach constructed (see Section VI.A). This can be seen in Fig. 4(a) with the long final legs taken by all three agents running the DH-mm approach. This does not occur when using the GA-multi approach as seen in Fig. 4(b).

Figs. 5(a) and 5(b) present the task sequences (paths) generated and executed for the 35 x 5 instance using the ACBBA-mm and GA-multi approaches. The GA-multi approach produced task sequences that are better on both objectives than those that the ACBBA-mm approach constructed (see Section VI.A). In particular, UAV4's task sequence is significantly improved when using the GA-multi approach versus the ACBBA-mm which can be seen with the purple paths in Figs. 5(a) and 5(b).




(a) (b)

Fig. 4. Task sequences produced by running agents with (a) DH-mm and (b) GA-multi on the 30 x 3 instance. The colored lines are the agents' task sequences, and the agents' ending positions are denoted by the labels UAV0, UAV1, and UAV2. Axes indicate position in meters.




(a) (b)

Fig. 5. Task sequences produced by running agents with (a) ACBBA-mm and (b) GA-multi on the 35 x 5 instance. The colored lines are the agents' task sequences, and the agents' ending positions are denoted by the labels UAV0 to UAV4. Axes indicate position in meters.

### C. *Experimental Convergence*

As each agent's GA thread continues to run and the agent receives good solutions from the other agents, the quality of the solutions in the population generally improves over time early in the mission. Table III is a time history that lists the average deviation from the lowest cost solution found throughout the mission and demonstrates how quickly this improvement occurs at the beginning of the mission. The GA approach rapidly converged to high quality task sequences that are shared by all agents in the system.

## VII. Conclusion

This paper presented a new decentralized GA approach for determining task allocations for multi-agent systems. This approach exploits the parallelism inherent in a GA and continues the search for better solutions while the agents complete tasks. We evaluated three variants of this approach by comparing their solutions to those generated by existing decentralized task allocation approaches. Our experiments considered both the min-sum and min-time objectives.

Table III. Average relative difference in cost between current best solution and best solution found (as percentage) for the GA variants at each sampled time for the four largest problem instances. Time (in seconds) is the duration from the time that the first iteration finished and yielded a complete solution.

| Variant | Time (s) | Instance | | | |
|---|---|---|---|---|---|
| | | 20 x 4 | 30 x 3 | 35 x 5 | 40 x 6 |
| GA-ms | 0 | 31.4 | 30.6 | 28.5 | 30.0 |
| | 1 | 5.0 | 3.8 | 3.2 | 12.1 |
| | 2 | 4.6 | 2.5 | 0.7 | 7.2 |
| | 3 | 4.3 | 2.4 | 0.5 | 2.0 |
| | 4 | 3.8 | 2.1 | 0.4 | 1.9 |
| GA-mm | 0 | 26.4 | 21.6 | 25.6 | 49.0 |
| | 1 | 8.2 | 1.1 | 14.9 | 17.2 |
| | 2 | 7.3 | 1.9 | 12.2 | 11.7 |
| | 3 | 5.6 | 1.7 | 16.5 | 7.7 |
| | 4 | 4.2 | 0.6 | 14.2 | 5.4 |
| GA-multi | 0 | 37.3 | 22.7 | 34.4 | 55.4 |
| | 1 | 7.1 | 2.3 | 12.8 | 13.8 |
| | 2 | 4.0 | 1.6 | 11.9 | 12.0 |
| | 3 | 3.9 | 1.2 | 11.8 | 7.2 |
| | 4 | 3.9 | 0.9 | 11.7 | 5.1 |

The results showed that, on average, the decentralized GA rapidly converged to high quality solutions on the instances considered and performed better on larger instances than existing approaches on mission completion time. It performed almost as well as the ACBBA and HIPC approaches on the total distance metric. The results demonstrate the performance advantages of the decentralized GA over standard approaches, particularly when the objective is minimizing mission completion time. Future work can develop enhancements within this new class of decentralized task allocation approaches that use parallelized evolutionary algorithms over a team of agents.

Future work should consider implementing and testing improvements to the decentralized GA approach. These include tuning the GA's parameters (such as the population size), adding more genetic operations such as crossover, and removing the constraint that every agent must be assigned at least one task. Other evolutionary algorithms such as particle swarm optimization [22] can also be considered. Also, alternative problem scenarios with unknown and moving targets may produce different results. We can also consider larger test instances to better understand the scalability of our approach.

## Acknowledgement

The authors thank Sharan Nayak for providing the multi-agent simulation framework used in this paper's experiments. This work was supported in part by grant FA8750-18-2-0114 from the Air Force Research Laboratory (AFRL).

## REFERENCES


[1] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300-313, 2016.

[2] A. Singh, "A review on algorithm used to solve multiple traveling salesman problem," *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 4, pp. 598-603, 2016.

[3] 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 2015*, pp. 31-51, 2015.

[4] V. Singhal and D. Dahiya, "Distributed task allocation in dynamic multi-agent system," in *International Conference on Computing, Communication & Automation*, pp. 643-648, 2015.

[5] L. Brunet, H. L. Choi, and J. How, "Consensus-based auction approaches for decentralized task assignment," in *AIAA guidance, navigation and control conference and exhibit*, p. 6839, 2008.

[6] L. Johnson, S. Ponda, H. L. Choi, and J. How, "Asynchronous decentralized task allocation for dynamic environments," in *Infotech@ Aerospace 2011*, p. 1441, 2011.

[7] 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 Transactions on Cybernetics*, vol. 46, no. 4, pp. 902-915, 2016.

[8] L. B. Johnson, H. L. Choi, and J. P. How, "Hybrid information and plan consensus in distributed task allocation," in *AIAA Guidance, Navigation, and Control (GNC) Conference,* p. 4888, 2013.

[9] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 900-909, 2010.

[10] V. Singhal and D. Dahiya, "Distributed Task Allocation in Dynamic Multi-Agent System," in *International Conference on Computing, Communication and Automation (ICCCA2015)*, pp. 643-648, 2015.

[11] M. R. Garey, and D. S. Johnson, "Computers and intractability: a guide to the theory of NP-completeness", *New York: Freeman*, 1979.

[12] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 23-28, 2017.

[13] H. W. Kuhn, "The Hungarian Method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97, 1955.

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

[15] G. Ping-An, C. Zi-Xing, and Y. Ling-Li, "Evolutionary computation approach to decentralized multi-robot task allocation," in *2009 Fifth International Conference on Natural Computation*, vol. 5, pp. 415-419, 2009.

[16] J. A. Sarma, "An analysis of decentralized and spatially distributed genetic algorithms," Ph.D. dissertation, George Mason Univ., Fairfax, VA, 1998.

[17] L. Zhang and T. N. Wong, "Distributed genetic algorithm for integrated process planning and scheduling based on multi agent system," in *7th IFAC Conference on Manufacturing Modeling, Management, and Control*, pp. 760-765, 2013.

[18] G. Iordache, M. S. Boboila, F. Pop, C. Stratanm V, Cristea, "A decentralized strategy for genetic scheduling in heterogeneous environments," *Multiagent and Grid Systems*, vol. 3, no. 4, pp. 355-367, 2007.

[19] J. Kirk, "Fixed start/end point multiple traveling salesmen problem - genetic algorithm," *MATLAB® Central File Exchange,* 2014, https://www.mathworks.com/matlabcentral/fileexchange/21299, [Accessed: July 11, 2019].

[20] M. Quigley, K. Conley, B. Gerkey, J. Faust, J., T. Foote, J. Leibs, ... and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[21] S. Nayak *et al.*, "Experimental Comparison of Decentralized Task Allocation Algorithms Under Imperfect Communication," *IEEE Robotics and Automation Letters,* vol. 5, no. 2, pp. 572-579, 2020.

[22] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363-371, 2002.