ACID: Ant Colony Inspired Deadline-Aware Task Allocation and Planning

Chia E. Tungom, Chen Jiamu and Chang Kexin

February 26, 2024

ACID: Ant Colony Inspired Deadline-Aware Task Allocation and Planning

Chia E. Tungom
OneWo Space-Tech Service Co., Ltd.
Shenzhen, China
chemago99@yahoo.com

Jiamu Chen
OneWo Space-Tech Service Co., Ltd.
Shenzhen, China
chenjm18@vanke.com

Kexin Chang
OneWo Space-Tech Service Co., Ltd.
Shenzhen, China
changkx01@vanke.com

Abstract

Efficient task allocation in multi-worker service environments, coupled with the necessity to adhere to strict deadlines, poses a multifaceted challenge in fields like service operations, operations management, logistics, and resource management. This study tackles the novel problem of 'deadline-aware multi-worker task planning.' This entails optimizing the allocation and planning of tasks among multiple workers to maximize task completion within specific time constraints. We introduce ACID, a novel flexible graph optimization metaheuristic algorithm, inspired by and extending traditional ant colony optimization. ACID uniquely incorporates heuristic information from task features and generational or iteration performance data. This includes task order, completion time, and both individual and swarm-level performance metrics, to devise an effective worker task plan. Through extensive experiments and simulations across varied scenarios, ACID demonstrates a significant improvement in task completion rates under tight deadlines compared to conventional methods. This research offers a valuable tool for industries requiring efficient task distribution while ensuring adherence to deadlines, with broad applications in logistics, e-commerce, manufacturing, and service industries.

CCS Concepts: • **Scheduling**; • **Task Allocation**; • **Multi-agent systems**; • **Swarm intelligence**;

Keywords: Ant Colony Optimization, Task Planning, Task Allocation, Decision Making

ACM Reference Format:
Chia E. Tungom, Jiamu Chen, and Kexin Chang. 2024. ACID: Ant Colony Inspired Deadline-Aware Task Allocation and Planning. In *Proceedings of 8th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI 2024)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Efficient allocation and planning of tasks in operational environments are crucial for industries ranging from hospitality to transportation, including healthcare and emergency services, to optimize productivity and adhere to strict deadlines. Notably, service-oriented companies like Uber, Didi, Lyft, Dordash, Meituan, and Instacart rely on effective real-time task assignment to meet consumer demands. The ride-sharing sector alone, for instance, has amassed a global value of $1.2 trillion as of 2023

In the commercial real estate service sector, efficient coordination of operations like maintenance and repair tasks is vital. These operations range from plumbing and electrical fixes to appliance repairs and inspections, essential for timely resolutions and service continuity

The development of schedules for large, distributed workforces introduces significant optimization challenges. Effective algorithms are needed to map tasks to workers, considering each individual's capabilities, locations, and time constraints

To tackle this problem, we have developed a specialized Ant Colony Optimization (ACO) inspired algorithm named ACID. This algorithm employs a metaheuristic search approach to derive near-optimal solutions. Its innovations include a deadline-focused heuristic function, a turn-based construction process, and an integrated local-global pheromone learning policy

The remainder of the paper is structured as follows: We first formulate the DA-MuW-TaP problem and discuss related works. Next, we detail ACID's allocation methodology and the decision-making policies of ants. Subsequently, we present experimental results from four geographical coverage scenarios, analyze the performance differences, and conclude with suggestions for future enhancements.

2 Problem formulation

In this section, we introduce the formulation of the *dead-line aware multi-worker task planning* **DA-MuW-TaP** problem. To solve this problem requires finding the optimal allocation of tasks to workers and deciding the order in which the tasks are completed by finding a plan for each of the k

workers $\mathcal{W} = \{w_1, ..., w_k\}$ such that all requested tasks $\mathcal{T}$ are completed on time.

The worker-task relation is represented by an unweighted directed graph $G = (V, E)$. Each worker w_i has a set of goal vertices $V_g \in V$ which are the tasks the worker is eligible to undertake at any given time T. A worker's tasks are to be planned such that the worker moves from one goal vertex to the next completing a tasks one at a time as shown in Fig 3. For instance, given a worker w_1 with goal vertices $\{V_g^1 = v_\tau^1, v_\tau^2, v_\tau^3, v_\tau^4, v_\tau^5\}$, we can find an optimal plan such that the worker completes tasks $\{v_\tau^1, v_\tau^2, v_\tau^4\}$ in the sequential order $\{v_\tau^2 \mapsto v_\tau^4 \mapsto v_\tau^1\}$ while the other tasks $\{v_\tau^3, v_\tau^5\}$ are allocated to other worker(s). Note that the goal vertices of two workers can overlap i.e two or more workers can be eligible for completing a given task. The objective is to find the optimal plan for all workers such that all tasks are completed on time.

2.1 Tasks and Deadlines

A task $v_\tau^j \in \mathcal{T}$ where j is the task id is defined with features $(W_\tau, d_\tau^j, geo_\tau^j, x_\tau^j)$ where W_τ is the set of workers that can undertake the task, d_τ^j is the task-deadline, geo_τ^j is the geo-location and x_τ^j is other features related to task (can be task constraints). We say that task v_τ^j is completed and on time if any worker $w_i \in W_\tau$ arrives at goal vertex v_τ^j and completes the task before its deadline d_τ^j but not on-time if it is completed after the deadline. A worker can only complete one task at a time. In this study we assume planning for tasks on a daily basis. Tasks for the day are provided along with the staff or workers available. This is realistic in most cases and in many applications. Tasks that take longer than a day can easily be in-cooperated into the problem and when assigned to the worker, the worker will not be made available for other tasks until the task is completed.

2.2 Tasks Allocation and Planning

The allocation of workers to task in the problem we set to solve is subject to the following constraints (1) no more than one worker can be assigned the same task (2) a worker cannot complete two tasks simultaneously. A task allocation and plan for all workers and tasks on a given day is given by $\mathcal{P} = \{p_i : i \in k\}$ where p_i is the task plan, the ordering of goal vertices for worker i (w_i).

2.3 Allocation and Planning Objective

The objective of the **DA-MuW-TaP** problem is to allocate and plan execution $\mathcal{P} = \{p_i : i \in k\}$ for all available tasks for workers on a daily bases such that the number of tasks completed on time is maximized. Each worker has a task execution schedule or plan π_i with worker i w_i to complete a number of tasks $\mathcal{T}_i \in \mathcal{T}$. Each task has a deadline d_τ^j and a worker execution time $t_{i,j}$ (the time taken to complete task j by worker i). From the plan π_i we compute a score for a worker by examining if the worker w_i completing a task v_τ^j (or simply task j) at time $T_{i,j}$ $(j \in \mathcal{T}_i)$ before the deadline d_τ^j using the formula in Equation 1. Note that j is used for simplicity to represent task v_τ^j.

$$\mathcal{R}_i = \sum_{j \in \mathcal{T}_i} 1[T_{i,j} < d_\tau^j] \tag{1}$$

$\mathcal{R}_i$ gives a score of 1 for each on-time task completion. For the all tasks and workers, the overall score is computed using Equation 2

$$\mathcal{R} = \sum_{i \in k} \mathcal{R}_i \tag{2}$$

Our objective therefore is to allocate tasks and schedule a plan so as to maximize the number of on-time task completed by all workers.

3 Related Work

The Deadline-Aware Task Planning (DA-MuW-TaP) problem addresses the efficient allocation of tasks to workers within specified deadlines, with related challenges extensively explored in the literature.

The study by Huang et al.

Despite differences in terminology, the DA-MATP problem is analogous to the proposed DA-MuW-TaP problem of allocating human workers to tasks with deadlines but hasn't been explored in the problem scenario which this study aims to cover. The mobile robots/agents correspond to workers, the pick stations correspond to tasks, and order preparation corresponds to task completion. Both aim to maximize tasks completed on time.

A widely related problem studied is multi-agent pickup and delivery (MAPD) planning, which assigns agents to delivery tasks involving pickup and destination locations

Overall, the literature on coordinating multiple agents for deadline-aware task completion is still nascent. DA-MATP provides a new problem formulation for multi-agent deadline-aware planning with applications in warehouse automation. In this study, the same formulation is adapted to model human worker allocation problems in DA-MuW-TaP. Additionally, this study proposes a graph based ACO modelling approach to solve the DA-MuW-TaP proobleem.

4 Solution methodology

To overcome the shortcomings of existing methods for the **DA-MuW-TaP** problem, we proposed ACID, an Ant Colony Inspired algorithm for **DA-MuW-TaP** that takes into account worker and task features and the collective performance of all workers during allocation and planning to evolve solutions. In this section, we briefly introduce ACO which helps set the foundation for understanding our proposed methodology in the followup section.

4.1 Ant Colony Optimization (ACO) Primer

ACO is a powerful metahheuristic that has been used to solve many hard problems in the field of operations research pioneered by it's performance on the TSP a popular benchmark for testing the performance of optimization algorithms

Ant Colony Optimization (ACO) comprises essential building blocks for solving optimization problems that include ***Ant Agents, Pheromone Trails, Probabilistic Decision Rule, Local Search Heuristics and Pheromone Update Mechanism***. Modeled after real ants, artificial *ant agents* navigate the solution space to find optimal solutions, iteratively selecting components like cities in the Traveling Salesman Problem (TSP) through a *probabilistic process*. Communication among agents relies on decentralized *pheromone trails*, where pheromones signify solution quality and are adjusted based on desirability. The ants' *decision rule* combines pheromone levels and heuristic information, incorporating optional *local search heuristics* for solution refinement. *Pheromone levels are updated* after each iteration, reinforcing paths leading to better solutions. ACO's adaptability extends beyond TSP, making it a versatile metaheuristic algorithm for various combinatorial optimization challenges.

In the context of the Traveling Salesman Problem: Cities represent the problem components, and the goal is to find the shortest possible tour that visits each city exactly once and returns to the starting city. Ants construct solutions (tours) by selecting cities probabilistically based on pheromone levels and heuristic information.

Building new variants of ACO beyond TSP involves several key steps. First is problem representation, which defines how solutions are represented in the context of the specific problem. Following this, is the definition of pheromone update rules, the pheromone update mechanism is tailored to align with the problem-specific evaluation criteria. Adjusting heuristics comes next, involving the modification of the heuristic information used by ants based on problem-specific characteristics. Additionally, if the optimization problem includes constraints, it is crucial to incorporate them into the construction and update mechanisms. By customizing these components, ACO can be effectively applied to a broad spectrum of optimization problems, underscoring its versatility and robustness as a metaheuristic algorithm.

4.2 Ant Colony Inspired DA-MuW-TaP Algorithm (ACID)

The solution framework for the DA-MuW-TaP is as shown in Figure 1. We solve the DA-MuW-TaP for a specific scenario in which task request are sent in through text, voice, and/or picture description and a multi-modal algorithm deciphers the task, labeling its attributes or features $(d_\tau^j, geo_\tau^j, x_\tau^j)$. The workers scheduled for service on the given day are provided and the worker task relation is mapped. A worker has a relation to a task if he/she can undertake the task. The workers and tasks are then fed to an allocation and planning algorithm to produce an optimal task allocation and plan for workers. In this study we focus on the task allocation and planning algorithm which takes as input the task-worker relation and produces a plan for workers.

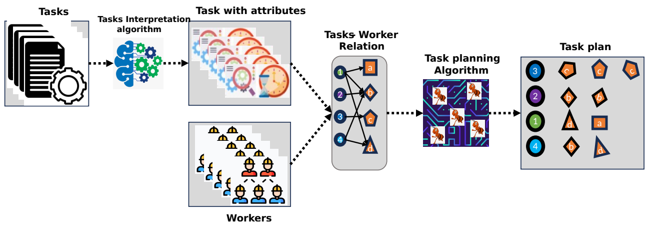

Figure 1. Framework for the DA-MuW-TaP Problem

Our proposed algorithm is Ant colony inspired which requires us to define an appropriate representation of the workers and tasks as a graph. After defining a representation, we outline the steps involves in the decision making process which include solution initialization, ant movement through a Probabilistic Decision Rule, Local Search Heuristics, Pheromone Update Mechanism, and termination criterion.

4.2.1 Representation. A worker-task graph $G_{w,v}$ is constructed using the worker-task relation and filled with 0's for worker not having the skill for a task and 1's otherwise. The graph is of size number of workers by number of tasks and tells us which task each worker is eligible to undertake.

Each ant $Ant_i \in \{Ant_1, ..., Ant_k\}$ in the swarm represents a worker and it's solution p_i represents a task plan for a worker. The number of ants in the swarm therefore is equal to the number of workers available at any given day and the solutions for ants in the swarm represents the plan for all workers.

This solution representation enables ants to independently plan their tasks while also taking into consideration the plan of other ants. Using the worker-task graph $G_{w,v}$, an ant Ant_i can only choose from it's eligible tasks V_g^i (goal vertices) at any point in time if the tasks has not been chosen by another worker already. This presents a challenge, with the question of what is the most efficient way for ants to choose a task such that it doesn't affect other ant choices negatively. We address with ACID using a specialized turn based ant movement strategy.

We represent the likelihood of a worker to be assigned a task with pheromone trails $P_{w,v}$. The intensity of the pheromone trail represents the desirability of a worker to choose a task and can be retrieved from the pheromone graph as shown in Fig 2.

4.2.2 Algorithm Steps. Haven designed a representation of the worker-task and pheromone graph, we are left to define how the ant colony updates the pheromone graph

Worker Task Graph $G_{w,v}$

	v_τ^1	v_τ^2	v_τ^3	v_τ^4
w_1	1	1	0	1
w_2	0	1	0	0
w_3	0	0	1	0
w_4	0	1	0	1

Pheromone Graph $P_{w,v}$

	v_τ^1	v_τ^2	v_τ^3	v_τ^4
w_1	0.9	0.3	0	0.8
w_2	0	1.0	0	0
w_3	0	0	1.0	0
w_4	0	0.5	0	0.7

Figure 2. Task-Worker relation and Pheromone graph representation

through a collective learning procedure inorder to discover the most suitable allocation and plan for workers. The steps involved in the learning process include pheromone initialization, ant movement, pheromone update mechanism and termination.

Initialization: Pheromone for worker-task not connected is initialized to 0 and $1/|\mathcal{T}_i|$ otherwise, where $|\mathcal{T}_i|$ is the number of goal vertices for worker i. This strategy gives the same preferences to all tasks for a worker at initialization. The swarm updates this graph through it's learning process with the objective to find a preferential allocation and plan for all workers.

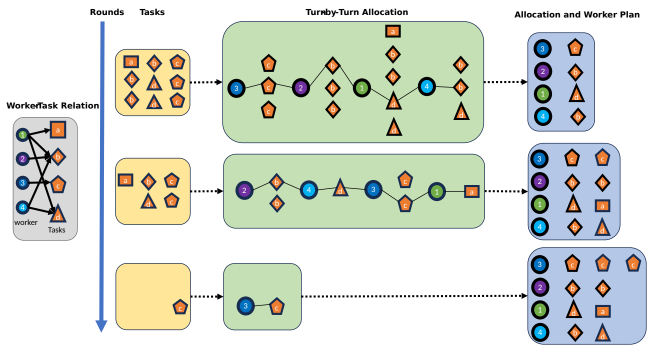

Figure 3. Framework for the DA-MuW-TaP Problem

Ant Movement: This is an important step in the solution search process and determines how ants communicate and traverse the worker task graph and. This enaables the ant agents to iteratively learn how to make better decision for themselves and for the entire swarm. Ant movement requires each ant agent to make a choice of choosing and adding a task too it's task plan. How the ants move, communicate and makes these choices determines how they refine their choices and ultimately how the final solution is reached. We design a **Round Base Turn-by-Turn allocation** movement of ants in which ant take turns choosing which tasks to add to their plan as shown in Figure 3. The design is such that all tasks are equally distributed across workers and the sequence is of allocation is taken into account. This is important because the order in which a worker completes a task affects his/her efficiency.

Round Base Turn-by-Turn allocation is a way in which ant agents sequentially decide which goal vertex or task to add to their plan. Ants take turns deciding which task to add to their plan influenced by pheromone and heuristic information provided to them and pick a task probabilistically using Equation 3. After each round, there are a given number of tasks left to be allocated and a worker is eligible for a round if it has a goal vertex in the round. The rounds continue until all tasks have been allocated and each agent has come up with a plan. During each round, agents are selected randomly to decide on a task to add to their plan. More specifically, if we have k-ants, each ant is randomly chosen to pick a task. After the ant has picked a tasked the next ant from the remaining k-1 ants is again randomly selected to pick a task. This continues until all ants have picked a task. After all the ants have completed their plan, the solution of every individual ant is evaluated along with the global solution and is used to update the pheromone intensity.

In each generation, ants take turns in rounds sequentially adding a task to their plan based on a combination of pheromone intensity and a heuristic factor. After each generation, ants agents update their best allocation plan p_i^{best} which is the plan in which the highest number of its tasks are completed on time evaluated using Equation 1. The global best solution $\mathcal{P}^{best}$ for the entire swarm is also updated after every generation. If there is a tie in performance for p_i^{best} or $\mathcal{P}^{best}$, both of the solutions are stored in a list. The probability of Ant_i adding a goal vertex v_τ^j to it's plan when deciding is given by Equation 3.

$$P_{i,j} = \frac{(\varphi_{i,j})^\alpha * (H_{i,j})^\beta}{\sum_{i,l} (\varphi_{i,l})^\alpha * (H_{i,l})^\beta} \tag{3}$$

where $\varphi_{i,j}$ and $H_{i,j}$ are the pheromone and heuristic values between ant i and task j respectively and α, β their respective influence parameter on the choice. l is every goal vertex available for selection at the given point for ant Ant_i i.e $l \subseteq V_g^i$.

Heuristic information between a worker w_i and task τ^j is determined using the distance needed to travel to the task and the time taken to complete the task by the worker. Therefore the longer it takes for a worker and the further the worker is from the task, the less likely it will be for the worker to finish the task in time. The heuristic information used for deciding on a task is given by Equation 4.

$$H_{i,j} = \frac{1}{d_{i,j} + t_{i,j}} \tag{4}$$

where $d_{i,j}$ and $t_{i,j}$ are respectively the distance and time for Ant_i to complete task j or v_τ^j. If the time is higher than the max completion time when added to the plan, the heuristic information is set to 0. This avoids agents from choosing tasks that lead to late task completion.

Pheromone Update: Pheromone update takes place at the end of each generation. A generation is one complete task allocation and planning step. Pheromone is updated based on the order in which the task where added to the plan and the quality of the local and global best solution. The update of the pheromone graph $P_{w,v}$ by Ant_i against each task v_τ^j is given by Equation 5.

$$\varphi_{i,j} = (1-\rho)\varphi_{i,j} + \frac{1}{R_j}\Theta_{i,j}\Big(\frac{\Upsilon_{p_i^{best}}}{\Phi_{p_i^{best}}} + \frac{\Upsilon_{\mathcal{P}^{best}}}{\Phi_{\mathcal{P}^{best}}} * \Lambda_{i,j}^{\mathcal{P}^{best}}\Big) \quad (5)$$

where ρ is the evaporation rate parameter of the pheromone usually in the range [0,1]. R_j is the rank of task v_τ^j in the plan of Ant_i. This is important because it ensures the order in which a worker completes a task is taken into account and therefor a worker that is to complete a task first in a good plan is given higher priority and is likely to be chosen first during planning. $\Theta_{i,j}$ is binary and is 1 if task j is completed on time by worker i and 0 otherwise. this enables that we only update pheromone to tasks that are completed on time and therefore reduce preference for tasks that will not be completed on time acting as a natural learned constraint.

A worker w_i has it's individual task plan p_i and this plan is a subset of a global plan $\mathcal{P}$. A good local and global solution is one in which the most or all tasks will are completed on time. We make use of the local and global solutions p_i^{best} and $\mathcal{P}^{best}$ to update the pheromone graph. $\frac{\Upsilon_{p_i^{best}}}{\Phi_{p_i^{best}}}$ is proportion of all task completed on time to all task in task in the current best plan of Ant_i. $\Upsilon_{p_i^{best}}$ is the number of tasks completed on time and $\Phi_{p_i^{best}}$ is the total number of tasks in plan p_i^{best}. This gives us the quality of the local solution. $\frac{\Upsilon_{\mathcal{P}^{best}}}{\Phi_{\mathcal{P}^{best}}}$ is the proportion of the total number of task completed on time to all the tasks available for the global best solution. this gives us the quality of the global solution. The global solution only contributes to pheromone deposition when Ant_i possesses task j in both the local and global best solution represented by $\Lambda_{i,j}^{\mathcal{P}^{best}}$. $\Lambda_{i,j}^{\mathcal{P}^{best}}$ is 1 if a j of Ant_i in p_i^{best} is in $\mathcal{P}^{best}$ as the task for Ant_i otherwise the value is 0. This enables the global best solution to contribute only when the local and global solution align allowing us to reinforce learning in promising solutions.

With the heuristic and pheromone information formulation, ants can effectively learn an allocation and planning of tasks iteratively over a set number of generations.

Note that during Round Base Turn-by-Turn allocation, if all the tasks presented to an ant are all above the deadline, we ignore adding any task to the plan of the set worker. After all allocations have been made and there are tasks that will be out of the deadline, do a Round Base Turn-by-Turn allocation in which we randomly add tasks to a workers plan without considering heuristic and pheromone information. This allocation doesn't affect the swarm learning ability as they swarm iteratively refine solutions by only updating pheromone to task completed on time.

Termination: The termination criteria determine the end of the task learning and allocation process. We end the solution search process when the best global solution does not change in 20 continuous generations or when the final generation is reached. The number of generations depends on the complexity of the problem. Empirically, we saw that setting the generations to 10000 works well across all problem sizes and sets given that early termination can be initiated when the solution converges.

Algorithm 1 Round Based Turn-By-Turn Allocation Algorithm

```
Require: Workers $\mathcal{W}$, Tasks $\mathcal{T}$
1: Task Plan $\mathcal{P} = \{\}$
2: while $\mathcal{T}$ not empty do
3:     Workers $\mathcal{W}$ with goal vertex in $\mathcal{T}$
4:     Shuffle $\mathcal{W}$
5:     for $w_i$ in $\mathcal{W}$ do
6:         $Ant_i$ adds a task $v_\tau^j$ to plan $p_i$ using Equation 3
7:         Add $p_i$ to plan $\mathcal{P}$
8:         Remove task $v_\tau^j$ from $\mathcal{T}$
9:     end for
10: end while
11: return $\mathcal{P}$
```

Algorithm 2 ACID

```
Require: Workers $\mathcal{W}$, Tasks $\mathcal{T}$, Generations $Gen$, Termination $Tem$, Worker-Task Graph $G_{w,v}$ Parameters $\alpha$, $\beta$
1: Initialize Pheromone Graph $P_{w,v}$, set $Tem = 0$, set $Gen = 0$
2: while not terminal $Gen$ or $Tem$ do
3:     Generate Plan using Algorithm 1
4:     Update $\mathcal{P}^{best}$ evaluated using Equation 2
5:     for $w_i$ in $\mathcal{W}$ do
6:         Update $p_i^{best}$ evaluated using Equation 1
7:     end for
8:     Update pheromone graph $P_{w,v}$ using Equation 5
9:     if $\mathcal{P}^{best}$ Updated in $Gen$ then
10:        Reset $Tem$ to $Tem = 0$
11:    else
12:        Update $Tem$ to $Tem = Tem + 1$
13:    end if
14:    Update $Gen$ to $Gen = Gen + 1$
15: end while
16: return $\mathcal{P}^{best}$
```

5 Empirical Evaluation

In this section, we illustrate the efficacy of ACID in daily task allocation and planning scenario through comprehensive experimentation and comparison with other methods. Our implementation of is in Python, and we carry out these experiments using a 2.2 GHz CPU equipped with 32 GB RAM. The comparison involves evaluating ACID against ROSETTA, ST-SAP and LT-SAP on simulated daily task allocation on a synthetic dataset whose distribution is derived from a real world scenario used daily in production. We vary the number of tasks and workers to evaluate the scalability of the algorithm.

The parameters α and β for ACID are determined from empirical evaluation. One of the key advantages of ACID is the fact that it has very few parameters to be tuned, with only the heuristic and pheromone α and β influencing the choice of a task at anay point in time. From the empirical results, we determine $\alpha = 0.79$ and $\beta = 0.21$ performed better showing that pheromone information is most influential in choosing a task. The nummber of generations $Gen = 1,000,000$ and Termination $Tem = 100$.

To generate a DA-MuW-TaP instance, we meticulously considered the distribution of jobs and workers for each day of the week. While direct access to the complete dataset was not available, we were granted permission to execute specific operations, enabling us to extract statistics related to the variety of tasks performed each day, as well as the average number of workers possessing relevant skills. For each task type, we also acquired the average time required for completion.

Utilizing these obtained statistics, we meticulously crafted a synthetic dataset for each day of the week, aiming to closely mirror the characteristics of the real-world dataset. Following the generation of synthetic data, a thorough comparison was conducted to ensure its alignment with the actual dataset. The number of tasks and workers were initially constrained to a specific geographical location, and subsequent scaling applied to achieve 4 varried sets name *Area1, Area2, Area3 and Area4.*

For clarity, the synthetic data generation process involved the following key steps:

- **Task and Worker Distribution:** We replicated the distribution of tasks and workers for every day of the week based on statistical insights derived from our partial dataset.
- **Task Completion Time:** The average time required to complete each task type was obtained from empirical data and used to determine task durations in the synthetic dataset.
- **Geographical Constraints:** The number of tasks and workers was initially confined to a specific geographical location, ensuring that the synthetic dataset's characteristics align with the intended real-world scenario.
- **Scaling Up:** Geographical scalability was achieved by directly increasing the number of tasks and workers, simulating an expanded operational scope. The scale of the tasks and agents or workers used for simulation are as shown in 1 to 4 on four different geographic areas named Area1 to Area4

Post-synthesis, a comparison was conducted to validate the synthetic data against the actual dataset. This verification process involved assessing key statistical parameters, ensuring that the synthetic data accurately represented the distribution of tasks, workers, and task completion times. The dataset was then employed in our experiments to assess the scalability and performance of the ACID algorithm in various geographical scenarios.

These synthetic datasets formed the basis for our comprehensive experimentation, allowing us to evaluate the efficacy of ACID in daily task allocation and planning scenarios. The subsequent sections provide detailed insights into our experimental methodology, including the specific parameters and evaluation criteria employed.

For simulation and evaluation, we built a simulation environment specifically designed for sequential decision-making and evaluation using OpenAI gym

5.1 Results and Discussion

In the empirical evaluation of daily task allocation and planning scenarios, we compare the performance of four algorithms: ST-SAP, LT-SAP, ROSETTA, and ACID on ontime completion, throughput and performance gains relatiive to the baseline ST-SAP. The results demonstrate that the proposed ACID algorithm outperforms the other approaches of ST-SAP, LT-SAP, and ROSETTA across different evaluation criteria and geographical coverage areas. Specifically, ACID achieved substantially higher on-time task completion rates and throughput compared to the other methods.

In Area1 (Table I), ACID attained the highest number of on-time completed tasks across all agent scenarios, surpassing the next best method ROSETTA by 31-45%. This superior performance underscores ACID's ability to effectively allocate tasks to meet deadlines even with relatively small number of workers. The percentage improvements over LT-SAP (Table V) further highlight ACID's strengths, with gains ranging from 49-100% in on-time completions.

The trends persist in larger geographic areas as well. In Area 2 (Table II), ACID accomplished up to 99% more on-time tasks than LT-SAP, exemplifying its scalability. Similar observations are made in Area 3 and Area 4, with ACID accomplishing up to 127% and 161% more on-time tasks than LT-SAP respectively.

A key factor driving ACID's effectiveness is the heuristics and pheromone update mechanisms which enable the ant colony to learn efficient allocations tailored to deadline requirements over generations. By balancing pheromone

Table 1. PERFORMANCE COMPARISON OF ALGORITHMS IN AREA 1

Algorithms	500 agents		750 agents		1000 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
ST-SAP	1970	2743	2118	3312	2198	3701
LT-SAP	2143	3202	2587	3532	2484	3663
ROSETTA	2736	3238	2967	3801	3984	4109
ACID	3956	4109	4011	4109	4179	4109

Table 2. PERFORMANCE COMPARISON OF ALGORITHMS IN AREA 2

Algorithms	750 agents		1000 agents		1250 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
ST-SAP	3508	6113	3930	7299	4229	8168
LT-SAP	4695	6986	5600	6811	5224	7058
ROSETTA	5967	7052	6325	7561	8282	8706
ACID	7744	8214	7828	7798	8061	8854

Table 3. PERFORMANCE COMPARISON OF ALGORITHMS IN AREA 3

Algorithms	1000 agents		1250 agents		1500 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
ST-SAP	6760	10716	8185	15780	8448	15041
LT-SAP	9585	13771	9904	14995	10356	14596
ROSETTA	11238	13211	13538	14738	15147	15805
ACID	15361	16650	14498	15730	14894	18649

Table 4. PERFORMANCE COMPARISON OF ALGORITHMS IN AREA 4

Algorithms	1250 agents		1500 agents		1750 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
ST-SAP	12939	22440	14400	34502	16740	31161
LT-SAP	17733	24565	21256	30323	19195	31785
ROSETTA	21019	27163	25992	31477	31578	33062
ACID	33790	34288	27951	31609	27368	41524

Table 5. Percentage Improvement from baseline LT-SAP IN AREA 1

Algorithms	500 agents		750 agents		1000 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
LT-SAP	8%	16%	22%	6%	13%	-1%
ROSETTA	38%	18%	40%	14%	81%	11%
ACID	100%	49%	89%	24%	90%	11%

Table 6. Percentage Improvement from baseline LT-SAP IN AREA 2

Algorithms	750 agents		1000 agents		1250 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
LT-SAP	33%	14%	42%	-6%	23%	-13%
ROSETTA	70%	15%	60%	3%	95%	6%
ACID	120%	34%	99%	6%	90%	8%

Table 7. Percentage Improvement from baseline LT-SAP IN AREA 3

Algorithms	1000 agents		1250 agents		1500 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
LT-SAP	41%	28%	21%	-4%	22%	-2%
ROSETTA	66%	23%	65%	-6%	79%	5%
ACID	127%	55%	77%	0%	76%	23%

Table 8. Percentage Improvement from baseline LT-SAP IN AREA 4

Algorithms	1250 agents		1500 agents		1750 agents	
	#on-time	throughput	#on-time	throughput	#on-time	throughput
LT-SAP	37%	9%	47%	-12%	14%	2%
ROSETTA	62%	21%	80%	-8%	88%	6%
ACID	161%	52%	94%	-8%	63%	33%

trails and heuristics, ants probabilistically build solutions that maximize on-time completions. Local and global search components further guide this learning process towards high-quality solutions.

In contrast, methods like ST-SAP perform static allocations without considering iterative feedback or self-correction. LT-SAP incorporates some learning but with less specialized heuristics for deadline-aware optimization. While competitive, these limitations constrain performance. ROSETTA is more adaptive but remains inferior to the metaheuristic search capabilities of ant colony approaches in ACID.

Furthermore, ACID also optimizes throughput across coverage areas, meeting or exceeding the rates attained by other techniques. This demonstrates efficient utilization of worker capacity while adhering to deadlines.

ACID's two-fold strengths of leveraging ant colony principles and incorporating deadline-focused design yield significant performance improvements on key metrics over a spectrum of evaluation scenarios. The consistent substantial gains affirm ACID's advantage as an allocation methodology where on-time task completion is vital.

While ACID demonstrates clear performance gains, further research can explore dynamic environments to incorporate unpredictability like new tasks or workers could reveal how to improve ACID's adaptability to volatility.

5.2 Conclusion

This study introduced a modeling approach for the dead-line aware multi-worker task planning (DA-MuW-TaP) problem and presents the ACID algorithm, inspired by Ant Colony Optimization principles, as a solution to this complex problem. The contribution of this study extends beyond the introduction of the DA-MuW-TaP problem and the ACID algorithm. It provides valuable insights into the comparative performance of cutting-edge algorithms, emphasizing ACID's consistent outperformance in meeting task deadlines. The robustness of ACID positions it as a promising solution for applications where precise and timely task allocation is critical. This research introduces a modelling and use of ant colony optimization using ACID in the landscape of task allocation and planning. The success of ACID underscores its potential for practical implementation in real-world scenarios. Future research directions may delve deeper into the scalability and adaptability of ACID, exploring its performance in dynamic environments and uncovering opportunities for further enhancements. The introduction of ACID marks a substantial advancement in addressing the DA-MuW-TaP problem, presenting a valuable contribution to the field of optimization algorithms.

References

[1] Duman NO, Ergun O, Behroozi M. Shared Last Mile Delivery. Reengineering the Sharing Economy: Design, Policy, and Regulation. 2023 Mar 31:210.

[2] Schwieterman JP, Craig C. Primed for growth: Amazon Air's freighter fleet, flight activity, and payload capacity compared to FedEx and UPS. Research in Transportation Economics. 2023 Nov 1;101:101337.

[3] Clapper Y, Berkhout J, Bekker R, Moeke D. A model-based evolutionary algorithm for home health care scheduling. Computers & Operations Research. 2023 Feb 1;150:106081.

[4] Tungom CE, Ding X, Wang L, Zhou C, Chen J, Cheng X, Yuan J. Applied Decision Focused Learning: An End-to-End Decision System for Task Allocation. InFuzzy Systems and Data Mining VIII 2022 Nov 4 (pp. 98-108). IOS Press.

[5] Guastaroba G, Côté JF, Coelho LC. The multi-period workforce scheduling and routing problem. Omega. 2021 Jul 1;102:102302.

[6] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE computational intelligence magazine. 2006 Nov;1(4):28-39.

[7] Huang T, Shivashankar V, Caldara M, Durham J, Li J, Dilkina B, Koenig S. Deadline-aware multi-agent tour planning. InProceedings of the International Conference on Automated Planning and Scheduling 2023 Jul 1 (Vol. 33, pp. 189-197).

[8] Okumura K. Lacam: Search-based algorithm for quick multi-agent pathfinding. InProceedings of the AAAI Conference on Artificial Intelligence 2023 Jun 26 (Vol. 37, No. 10, pp. 11655-11662).

[9] Alessandro Farinelli, Antonello Contini, and Davide Zorzi. 2020. Decentralized Task Assignment for Multi-item Pickup and Delivery in

Logistic Scenarios. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS ’20). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1843–1845.

[10] Xiaohu Wu, Yihao Liu, Xueyan Tang, Wentong Cai, Funing Bai, Gilbert Khonstantine, and Guopeng Zhao. 2022. Multi-Agent Pickup and Delivery with Task Deadlines. In IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT ’21). Association for Computing Machinery, New York, NY, USA, 360–367. https://doi.org/10.1145/3486622.3493915

[11] Dorigo M, Stützle T. Ant colony optimization: overview and recent advances. Springer International Publishing; 2019.

[12] Fidanova S, Fidanova S. Ant colony optimization. Ant Colony Optimization and Applications. 2021:3-8.

[13] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W. Openai gym. arXiv preprint arXiv:1606.01540. 2016 Jun 5.

[14] Ma H, Hönig W, Kumar TS, Ayanian N, Koenig S. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. InProceedings of the AAAI Conference on Artificial Intelligence 2019 Jul 17 (Vol. 33, No. 01, pp. 7651-7658).

[15] Lau TT, Sengupta B. The multi-agent pickup and delivery problem: Mapf, marl and its warehouse applications. arXiv preprint arXiv:2203.07092. 2022 Mar 14.

revised 20 February 2024