# Task Group Allocation for Solving Multi-Load Agent Pickup and Delivery problem

Anonymous Author(s)
Submission Id: «EasyChair
submission id»

## ABSTRACT

This document outlines the formatting instructions for submissions to AAMAS-2025. You can use its source file as a template when writing your own paper. It is based on the file 'sample-sigconf.tex' distributed with the ACM article template for LaTeX.

## KEYWORDS

Multi-load agent, Pickup and delivery problem, Task allocation

## 1 INTRODUCTION

The multi-load agent pickup and delivery problem (MLAPD) is a variant of the pickup and delivery problem which is a well-known combinatorial optimization problem in the field of logistics and transportation. In the MLAPD problem, there are multiple agents, each of which can carry multiple tasks at the same time. The goal is to allocate tasks to agents to optimize the completion time of all tasks. The MLAPD problem is a NP-hard problem, and it is difficult to solve it optimally in polynomial time [1].

## 2 RELATED WORK

## 3 PROBLEM DEFINITION

In this section, we introduce the multi-load agent pickup and delivery problem, which we address in this paper. Before this, we first give the definitions of map, task, agent, comparative metrics and MLAPD problem.

*Definition 3.1 (Map).* A map $G = (V, E)$ is the , whose vertices $V$ correspond to locations and edges $E$ represent edges between adjoining locations. The shortest distance between two locations $v_i$ and $v_j$ is denoted as $dis(v_i, v_j)$.

*Definition 3.2 (Task).* A task $\tau \in \Gamma$ is a tuple $< v_\tau^s, v_\tau^g, t_\tau^r, t_\tau^d, t_\tau^c >$, where $v_\tau^s$ and $v_\tau^g$ are the pickup location and delivery location, respectively. $t_\tau^r, t_\tau^d$ and $t_\tau^c$ are the release time, deadline and completed time, respectively.

*Definition 3.3 (Agent).* An agent $a \in A$ is represented as a tuple $< v_a, c_a, \Gamma_a, S_a >$, where $v_a$ is location, $c_a$ represents the maximum capacity, $\Gamma_a$ is the set of tasks, and $S_a$ is the schedule of $\Gamma_a$. The

$S_a = < v_1, v_2, ..., v_n >$, where $v_i$ is a pickup or delivery location of task $\tau \in \Gamma_a$.

A task $\tau \in \Gamma$ is released at $t_\tau^r$. When system assigns $\tau$ to an agent $a$, $a$ needs to reach $v_\tau^s$ before $t_\tau^d$, and then deliver $\tau$ to $v_\tau^g$ while recording the completed time $t_\tau^c$. If $\tau$ could not be completed before $t_\tau^d$, then the task is considered as failed.

*Definition 3.4 (Comparative metrics).* In this paper, we demonstrate algorithmic superiority through three comparative metrics, service time ($ST$), makespan ($MS$) and completion ratio ($CR$). $ST = \sum_{\tau_i \in \Gamma} (t_{\tau_i}^c - t_{\tau_i}^r)$, is defined as the sum of differences between the completed times and the release times of all tasks. $MS = max\{t_\tau^c\}$ ($\forall \tau \in \Gamma$) is time when last task has completed. $CR = \sum_{\tau_i \in \Gamma} X_{\tau_i} / |\Gamma|$, where $X_{\tau_i}$ is a decision variable, is completion ratio of $\Gamma$. If task $\tau_i$ is completed, then $X_{\tau_i}$ is 1; otherwise, it is 0.

*Definition 3.5 (MLAPD).* The multi-load agent pickup and delivery problem (MLAPD) is a tuple $P = < G, A, \Gamma >$, defined by a map $G$, a set of agents $A$, and a set of tasks $\Gamma$. Moreover, $P$ amounts to finding the allocation of tasks to agents, which optimize three comparative indicators, ($ST, MS, CR$). Time is discretized into timesteps, and agents can only move to adjacent locations or wait at its current location in one timestep. Howerver, a conflict occurs when two agents are at the same location or pass through the same edge at the same time. The time required to avoid conflicts needs to be considered, since it will affect the comparative metrics. Therefore, we assue that agents can take $\theta$ time to avoid a conflict.

## 4 SOLUTION

In this study, Task-Group Allocation Algorithm (TGA) is developed to solve MLAPD problem efficiently. The TGA method first analyzes the likelihood of carpooling between each task, and then divide the set of tasks into groups via K-Capacity Hierarchical Clustering Algorithm (KCHC). Then, we allocate task groups to agents.

### 4.1 Task Group

The biggest difference between multi-load agent and single-load agent is that multi-load agent can load multiple items at the same time. This means that multi-load agent performing a task affects the completion time of the "co-passenger" tasks which are also executed to the same agent at same time. As shown as in Figure 1, there is an agent, $a$, and three tasks, $\tau_1$, $\tau_2$ and $\tau_3$. The agent $a$ is located at pickup location of $\tau_1$. The blue lines are the shortest paths for agent $a$ to complete $\tau_1$ and $\tau_2$. The red lines are the shortest paths for agent $a$ to complete $\tau_1$ and $\tau_3$. When we allocate $\tau_1$ and $\tau_2$ to the agent $a$, the agent $a$ needs to take the blue paths to complete $\tau_1$ and $\tau_2$ by greedy strategy, and it will not impact the completion time of $\tau_2$ but lengthen the completed time of $\tau_1$. Howerver, if we allocate $\tau_1$ and $\tau_3$ to the agent $a$, $a$ needs to take the red paths to
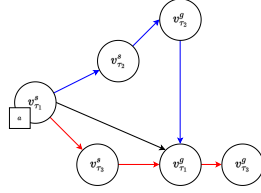
Figure 1: The impact of co-passenger tasks

complete $\tau_1$ and $\tau_3$. While it also lengthens the completed time of $\tau_1$, it is obvious that $\tau_3$ has less of an effect on $\tau_1$ as compared to $\tau_2$. Therefore, we need to consider the impact of "co-passenger" tasks when allocating tasks to agents. We
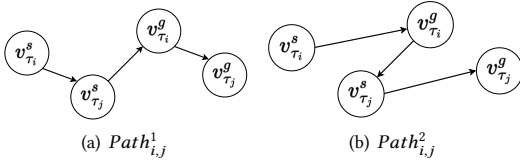


(a) $Path_{i,j}^1$       (b) $Path_{i,j}^2$

Figure 2: All possible paths of two tasks

$$\Delta_{i,j} = min \tag{1}$$

THEOREM 4.1. *Given a set of tasks* $\Gamma_N$,

$$Cost \leq \sum_{\tau_i \in \Gamma_N} dis(v_{\tau_i}^s, v_{\tau_i}^g) - 2 * (N-1)\Delta \tag{2}$$

PROOF. this is a proof. □

## 4.2 K-Capacity Hierarchical Clustering

Based on Theorem 4.1, we can divide the set of tasks into groups. The K-Capacity Hierarchical Clustering Algorithm (KCHC) is proposed to divide the set of tasks into groups. The KCHC algorithm is shown in Algorithm 1.

---
**Algorithm 1** K-Capacity Hierarchical Clustering Algorithm
---
**Require:** $N$ tasks $\Gamma_N$, $K$ capacity.
**Ensure:** $M$ task groups $\Gamma_M$.
    $M \leftarrow 0$
---

## REFERENCES

[1] Xiaoshan Bai, Andres Fielbaum, Maximilian Kronmüller, Luzia Knoedler, and Javier Alonso-Mora. 2022. Group-based distributed auction algorithms for multi-robot task assignment. *IEEE Transactions on Automation Science and Engineering* 20, 2 (2022), 1292–1303.