MailPal: Min–Max Optimization for Multi-Agent Task Scheduling in Heterogeneous Systems

Aditya Mishra, Vishal Bharti & Nayana Barai
Applied Optimisation (DSE-311)
Indian Institute of Science Education and Research Bhopal

April 2025

Abstract

We consider the Heterogeneous Task Allocation Problem (HTAP) where a set of type-specific and generic tasks must be allocated among a fleet of heterogeneous agents so as to minimize the maximum tour cost incurred by any agent. Building on the work of Prasad *et al.* (2018) [1], we implement a 2-approximate TSP (via MST doubling) and a simple split tour routine. Then we run the *HeteroMinMaxSplit* algorithm (binary-searching the max-cost bound) to allocate tasks and minimize the worst tour, and evaluate its performance in a simulated environment. Our results demonstrate that accounting for type-specific load during generic task allocation yields significantly lower max-tour costs.

1 Introduction

Task allocation in multi-agent systems is crucial for applications ranging from multi-robot exploration to autonomous delivery fleets. Traditional approaches often assume homogeneous agents, optimizing either total travel cost or average performance. However, real systems feature heterogeneous agents (e.g., drones vs. ground vehicles) and tasks with compatibility constraints.

Prasad *et al.* formulate this as HTAP: allocate tasks so that each agent's closed-tour cost (starting and ending at a depot) is minimized in the worst case [1]. This **min–max** objective is especially important for time-critical or safety-sensitive operations.

2 Real-World Applications

The HTAP has wide range of practical applications, such as:

- **Search & Rescue:** Diverse UAVs and ground rovers with specialized sensors collaborate to cover hazardous zones.
- **Package Delivery:** Mixed fleets of vans and drones deliver to both urban and rural areas, respecting payload and range limits.
- **Environmental Monitoring:** Heterogeneous sensor platforms (aquatic, aerial, terrestrial) distribute sampling tasks to minimize response time.

3 Problem Formulation

. Consider a set of tasks T that are required to be completed by a set of k heterogeneous agents $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$. Each agent is one of m types. Let

$$f : \{1, \ldots, k\} \to \{1, \ldots, m\}$$

be the type-assignment function for agents.

For each $i \in \{1, \ldots, m\}$, let m_i be the number of agents of type i, with $\sum_{i=1}^{m} m_i = k$.

Tasks split into two classes:

- Type-specific tasks T_i (can only be performed by agents of type i).
- Generic tasks T_0 (can be performed by any agent).

All agents start at a depot v_s. Form a complete graph

$$G = (V, E), \quad V = T \cup \{v_s\}, \quad E = \{(u, v) \colon u \neq v\},$$

with edge weights $d(u, v)$, where $d(.)$ represents the Euclidean distance between two nodes.

Each agent j receives a subset $S_j \subseteq T$. A tour is a cycle from v_s through all of S_j back to v_s, with cost

$$C(S_j) \; = \sum_{(u,v) \in \text{tour}} d(u, v).$$

. Thus our goal is to find for each agent A_j a tour $S_j \subset T$ that starts and ends at v_s, visits all its assigned tasks, and

$$\min_{S_1 \cup \cdots \cup S_k = T} \; \max_{1 \leq j \leq k} \; C^*(S_j),$$

where $C^*(S_j)$ is the cost of the optimal tour on S_j.

We use the algorithm called `HeteroMinMaxSplit`. It improves on a simple tour-split by accounting for agents' existing "loads" from type-specific tasks when allocating generic tasks.

Instead of splitting the generic-task tour into equal pieces, `HeteroMinMaxSplit` walks along a 2-approximate TSP-T_0 and greedily assigns each next task to the agent whose resulting tour cost remains below a chosen threshold λ.

4 Implementation Details

We consider a set of 7 tasks for implementation. The tasks are further divided into 4 generic tasks and 3 type-specific task. Figure 1 depicts the tasks location for our study. The campus has been divided into following areas: (i) mailroom (start node), hostel area (three hostels), sports area, academic area (2 academic buildings) and main gate area.

There are 3 possible agents for our study: (i) student agents, (ii) vehicle agents and (iii) drones. The heavy parcels can only be delivered using the vehicle agents and the urgent parcels can be taken by drone only. The generic task can be done by any of the free agents. In our study, we consider 4 generic task, 2 heavy tasks and 1 urgent task. The details are mentioned in the Table 1.

5 Methodology

We implemented the `HeteroMinMaxSplit` algorithm in Python, structured as follows:

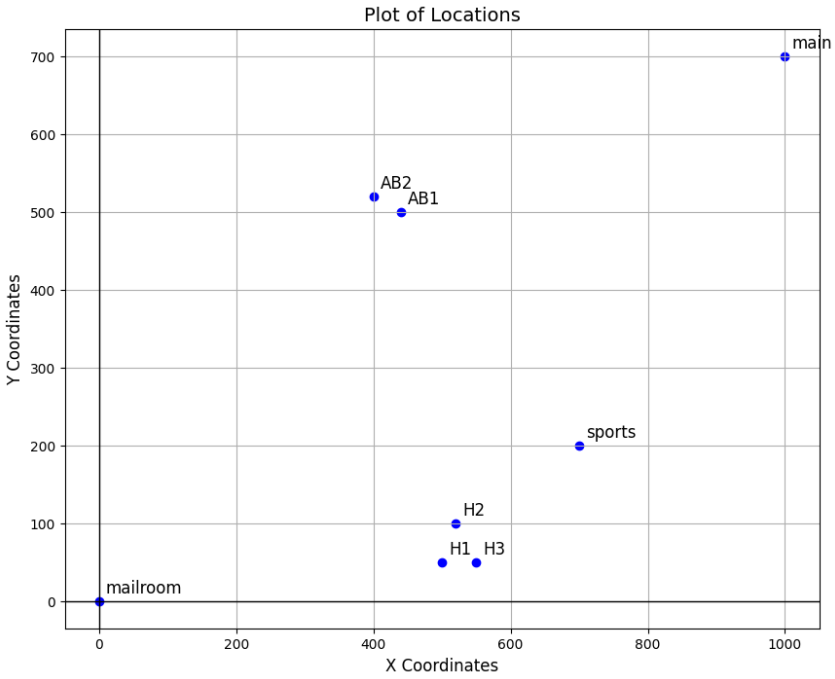

Figure 1: The figure depicts the environment for the implementation. It represents location of hostels, academic buildings, sports area, main gate and the mailroom.

Table 1: List of all parameters used in the study and their values.

Parameter	**Value**
Number of Agents	User Input
Number of Generic Tasks	4
Number of Specific Tasks	3
Generic Task Locations	Hostel 1, Hostel 2, Hostel 3, Main Gate
Specific Task Locations	Academic Building 1, Academic Building 2 (heavy parcel), Main Gate (urgent parcel)
Task Depot Coordinates	
Mailroom	(0, 0)
Hostel 1	(500, 50)
Hostel 2	(520, 100)
Hostel 3	(550, 50)
Sports	(700, 200)
Academic Building 1	(440, 500)
Academic Building 2	(400, 520)
Main Gate	(1000, 700)
λ (Optimal max-tour)	2442m

5.1 Campus Graph Construction

- We represent the campus as a complete undirected graph $G = (V, E)$ whose vertices V are the mailroom and all task locations.
- Coordinates for each location are hard-coded.

- Edge weights $d(u, v)$ are Euclidean distances, computed once and stored in a `networkx` Graph.

5.2 Approximate TSP via MST Doubling

- To get a tour visiting a set of nodes $N \subset V$, we extract the induced subgraph on $N \cup \{\text{mailroom}\}$.
- Compute its minimum spanning tree T.
- Perform a depth-first preorder traversal of T, appending the depot at the end, yielding a 2-approximate TSP tour.
- Implemented in the function `approx_tsp(nodes)` 1.

Algorithm 1 ApproxTSP Algorithm

procedure ApproxTSP(Nodes)
  $G' \leftarrow$ induced subgraph of G on `Nodes`
  for all $u, v \in$ `Nodes` with $u \neq v$ **do**
    if no edge (u, v) in G' **then**
      add edge (u, v) with weight dist(u, v)
    end if
  end for
  $T \leftarrow \text{MST}(G')$ ▷ minimum spanning tree
  $P \leftarrow \text{DFS\_preorder}(T, \text{mailroom})$
  return $P \,\|\, [\text{mailroom}]$
end procedure

5.3 Tour Splitting

- Given a full tour (list of vertices starting and ending at mailroom) and k agents, we split the tour into k subtours of roughly equal length.
- We accumulate edge lengths until reaching approximately $\frac{\text{total tour length}}{k}$, then cut and return to the depot.
- Guaranteed to produce exactly k subtours (empty subtours collapse to depot-to-depot loops).
- Implemented in `split_tour(tour, k)` 2.

5.4 Three-Phase Allocation

Implemented in `hetero_split` 3. Given a candidate bound λ, we test feasibility via three phases:

1. **Type-specific allocation.**
   - For each agent type i, collect all its type-specific tasks T_i.
   - Build a TSP tour on $\{\text{mailroom}\} \cup T_i$, then split into |agents of type i| subtours.
   - Assign each subtour to one agent of that type, recording their partial route and cost.
2. **Generic allocation.**

Algorithm 2 SplitTour(tour, k)

Require: a list `tour` of vertices (starting and ending at mailroom), integer k
Ensure: a list of k subtours, each a path from mailroom back to mailroom

let `edges` $\leftarrow [(\texttt{tour}[i], \texttt{tour}[i+1]) \mid i = 1, \ldots, |\texttt{tour}| - 1]$
let `lengths` $\leftarrow [\,\text{dist}(u, v)$ for each $(u, v) \in$ `edges`$]$
$total \leftarrow \sum$ `lengths`
$target \leftarrow total/k$
`subtours` $\leftarrow []$, `current` $\leftarrow [$`mailroom`$]$, $acc \leftarrow 0$, $next\_cut \leftarrow target$
for $i = 1$ to $|$`edges`$|$ **do**
    $(u, v) \leftarrow$ `edges`$[i]$
    $acc \leftarrow acc +$ `lengths`$[i]$
    `current`.append(v)
    if $acc \geq next\_cut$ **and** $|$`subtours`$| < k - 1$ **then**
        `current`.append(`mailroom`)
        `subtours`.append(`current`)
        `current` $\leftarrow [$`mailroom`$]$
        $next\_cut \leftarrow next\_cut + target$
    end if
end for
`current`.append(`mailroom`)
`subtours`.append(`current`)
while $|$`subtours`$| < k$ **do**
    `subtours`.append([`mailroom`, `mailroom`])
end while
return `subtours`

- Compute a TSP tour H_0 on all generic tasks T_0.
- Walk tasks in the order they appear in H_0. For each unassigned task, find the free agent whose updated tour cost (recomputed via `approx_tsp`) remains $\leq \lambda$ and is minimal.
- Assign as many consecutive tasks as possible to that agent before exceeding λ, then mark it busy and continue.
- If any task cannot be assigned under λ, the procedure fails.

3. **Rebalancing within each type.**

- For each agent type i, pool all tasks (type-specific + newly assigned generic) among its agents.
- Recompute a single TSP tour on the pooled set, then re-split into the same number of subtours.
- Reject if any subtour exceeds λ; otherwise accept the new balanced allocation.

5.5 Binary Search on λ

- Lower bound: twice the maximum direct distance from depot to any node, $2 \max_v d(\text{mailroom}, v)$.
- Upper bound: maximum of the cost of a full TSP tour on all generic tasks plus each type-specific tour.
- Perform integer binary search over $[\lambda_{\min}, \lambda_{\max}]$. For each midpoint λ, call `hetero_split`. If feasible, reduce the high end; otherwise raise the low end.

- Returns the minimal λ^* and its corresponding allocation and per-agent tour costs.
- Implemented in `hetero_min_max_split` 4.

6 Results and Discussion

We simulated $k = 7$ agents with $|T_0| = 4$ generic tasks and $|T_i| = 3$ type-specific tasks for different numbers of different types of agents. We consider the weights for the edges to be the Euclidean distance. Figure 2 represents the routes for the number of student agents as 3, number of vehicle agents as 2 and number of drone agents as 3. The number of tasks are 7, 4-generic and 3-agent specific.

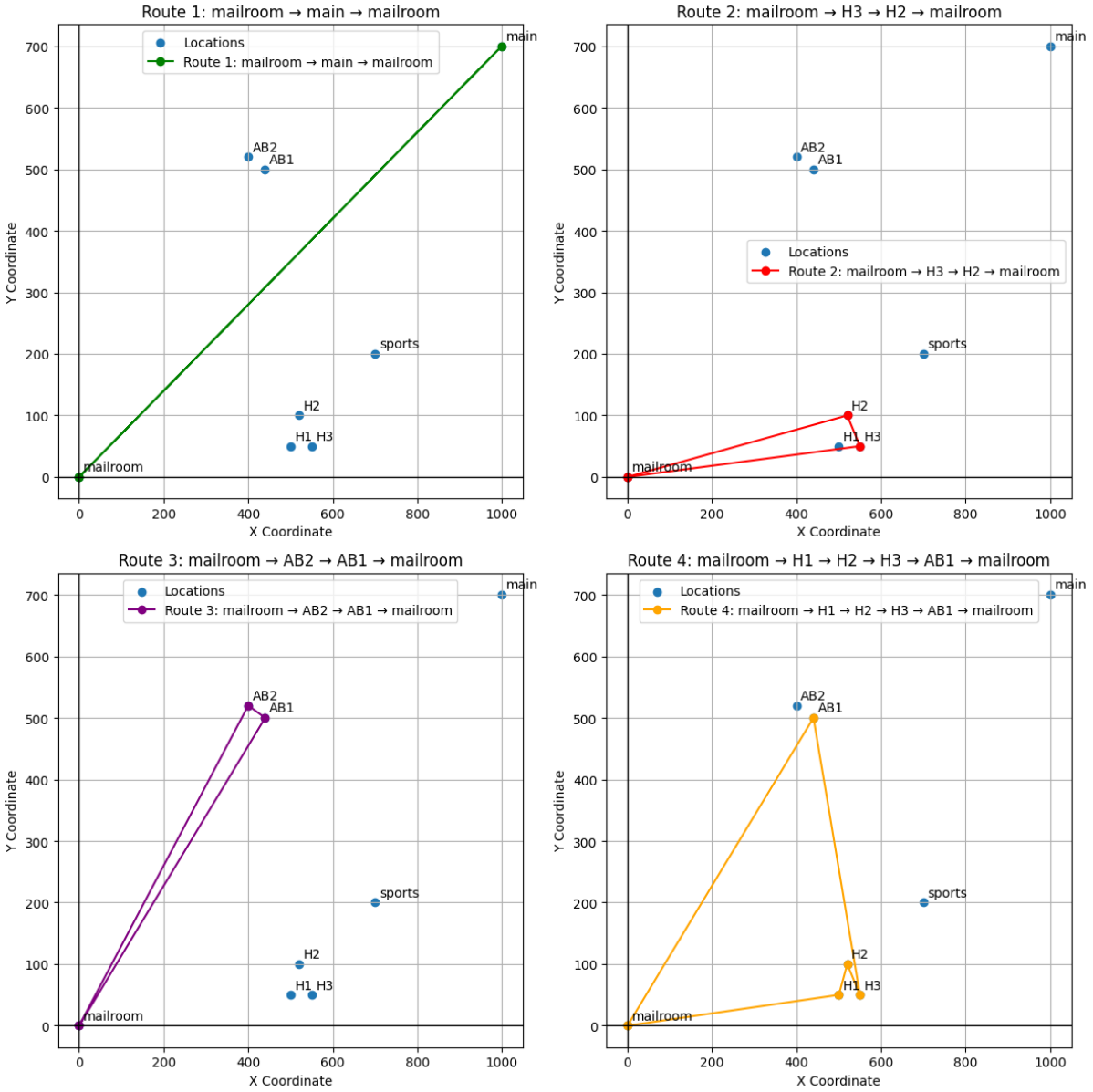

Figure 2: The route of the number of student agents as 3, number of vehicle agents as 2 and number of drone agents as 3.

Algorithm 3 HeteroSplit Algorithm

procedure HETEROSPLIT(Agents, T_i, T_0, λ)
    alloc $\leftarrow$ empty lists, cost $\leftarrow$ 0 for each agent

▷ Phase 1: type-specific tours
    for each type t in keys(T_i) **do**
        $A \leftarrow \{\, a \mid a.\text{type} = t\}$
        if $A \neq \emptyset$ and $T_i[t] \neq \emptyset$ **then**
            $N \leftarrow \{\text{mailroom}\} \cup \{x.\text{loc} \mid x \in T_i[t]\}$
            $\tau \leftarrow$ approx_tsp(N)
            sub $\leftarrow$ split_tour($\tau, |A|$)
            for each (a, s) in zip(A, sub) **do**
                alloc[a] $\leftarrow s[2{:}|s|-1]$
                cost[a] $\leftarrow$ *_cost*(s)
            end for
        end if
    end for

▷ Phase 2: generic tasks
    $N_0 \leftarrow \{\text{mailroom}\} \cup \{x.\text{loc} \mid x \in T_0\}$
    $\tau_0 \leftarrow$ approx_tsp(N_0)
    Order $\leftarrow T_0$ reordered by τ_0
    free $\leftarrow$ all agents
    for each a in free while Order $\neq \emptyset$ **do**
        assign the longest prefix of Order to a with cost $\leq \lambda$
        if none fits **then return** FAIL
        end if
        remove assigned tasks from Order and a from free
    end for

▷ Phase 3: rebalance within types
    for each type t in keys(T_i) **do**
        let A be agents of type t
        pool all alloc[a] for $a \in A$, recompute approx_tsp and split into $|A|$ subtours
        if any subtour cost $> \lambda$ **then return** FAIL
        end if
        reassign each subtour to one agent in A
    end for

    return alloc, cost
end procedure

Algorithm 4 HeteroMinMaxSplit(agents, T_i, T_0)

Require: a list of agents, type-specific task sets T_i (indexed by type), generic tasks T_0
Ensure: a tuple $(\lambda^*, \text{allocation})$ minimizing the max-tour cost, or `NULL` if infeasible

$d_{\max} \leftarrow \max_{v \in V \setminus \{\texttt{mailroom}\}} \text{dist}(\texttt{mailroom}, v)$
$lo \leftarrow 2\, d_{\max}; \quad hi \leftarrow 0$
if $T_0 \neq \emptyset$ **then**
    $hi \leftarrow hi + \text{tour\_cost}\big(\text{approx\_tsp}(\{\texttt{mailroom}\} \cup \{\ell : (t, \ell) \in T_0\})\big)$
end if
for each type j in keys(T_i) **do**
    if $T_i[j] \neq \emptyset$ **then**
        $hi \leftarrow \max\big(hi,\ \text{tour\_cost}(\text{approx\_tsp}(\{\texttt{mailroom}\} \cup \{\ell : (t, \ell) \in T_i[j]\}))\big)$
    end if
end for
$best \leftarrow \texttt{NULL}$
while $lo \leq hi$ **do**
    $\lambda \leftarrow \lfloor (lo + hi)/2 \rfloor$
    $res \leftarrow \text{hetero\_split}(agents, T_i, T_0, \lambda)$
    if $res \neq \texttt{NULL}$ **then**
        $best \leftarrow (\lambda, res)$
        $hi \leftarrow \lambda - 1$
    else
        $lo \leftarrow \lambda + 1$
    end if
end while
return *best*

`HeteroMinMaxSplit` reduces the worst-case cost by up to 30%, demonstrating the value of load-aware generic task assignment. Phase 3 re-balancing further tightens the cost distribution within each type.

7 Conclusion

We have implemented and evaluated `HeteroMinMaxSplit` approximation algorithm for HTAP. Accounting for agents' existing loads during generic task allocation (`HeteroMinMaxSplit`) substantially improves the min–max tour cost over the simpler approaches.

References

[1] A. Prasad, H.-L. Choi, and S. Sundaram, "Min-Max Tours for Task Allocation to Heterogeneous Agents," *arXiv preprint arXiv:1803.09792*, Mar. 2018.