Bipartite Matching and Routing with Congestion Costs:
A convex approach to robot task assignment and the multi-agent pathfinding problem

Kelly Ho

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2023

Committee:

Lillian Ratliff

Sam Burden

Program Authorized to Offer Degree:

Electrical and Computer Engineering

University of Washington

Abstract

Bipartite Matching and Routing with Congestion Costs:
A convex approach to robot task assignment and the multi-agent pathfinding problem

Kelly Ho

Chair of the Supervisory Committee:

Lillian Ratliff

Department of Electrical and Computer Engineering

Motivated by the congestion-aware path planning and task assignment problem emerging from traffic assignment, ride-sharing, and multi-robot task assignment problems, this thesis incorporates shortest path and assignment problems with congestion by formulating them as a convex optimization problem to minimize the overall matching and routing cost. Two problem settings considered are the street network of Seattle City and the multi-robot system in a warehouse where the drivers and robots are the agents, and the delivery locations are the tasks. We apply our formulation to the defined problem settings and introduce congestion-based cost functions to help agents avoid congestion and reach Wardrop equilibria. Finally, we present a Frank-Wolfe-based algorithm combined with shortest path algorithms such as Dijkstra's and A* algorithms and matching algorithms such as the Hungarian (Kuhn-Munkres) algorithm to find the optimal solution while considering congestion in the system. Furthermore, we extend balanced assignments to unbalanced assignments in our simulation. The results demonstrate that the proposed optimization problem formulation and algorithm effectively provide optimal matching and routing solutions with congestion avoidance.

Contents

List of Figures

Acknowledgments

I would like to express my gratitude towards my academic adviser, Professor Lillian Ratliff, for accepting me as her Master's student and providing support for my thesis. I am grateful for her patience and wisdom that pushed me to overcome challenges and grow enormously throughout my graduate school education. Thank you for giving me the opportunity to work with you and believing in my ability to do well. I am also grateful for Professor Sam Burden being one of the committee members and sharing all the advice and support for my work. It is a great honor to work with Professor Lillian Ratliff and Professor Sam Burden, and I will forever appreciate this experience.

I would like to acknowledge my postdoctoral advisor, Dan Calderone, who made this thesis possible. This work is a joint effort with Dan, and I am grateful for his constant guidance and support. The mentorship he provided me has helped me step out of my comfort zone and explore new problems and knowledge with a great sense of achievement. Thank you for taking a chance on me and bringing me into the world of research. It is an honor to work with Dan, and I appreciate the chance of having such a great mentor.

Finally, I would like to thank my family for their unconditional love and support in pursuing my dreams and goals. I appreciate having you in my life and sharing all of my journeys with you. Thank you for always believing in me and making me a better person.

Dedication

I would like to dedicate this thesis to my mother and grandmother who have been supporting and encouraging me to achieve my dreams and overcome challenges in life. Thank you for being great role models and providing constant love and support. I love you both and I appreciate everything that you have done for me.

1 Introduction

As traffic assignment problems (TAPs) and linear assignment problems (LAPs) have been applied in aeronautical [1], robotics [2], and operational [3] areas, there exists a wide variety and broad literature on TAPs and LAPs. In addition, new problem formulations and algorithms are emerging to solve both TAPs and LAPs simultaneously in ride-sharing and multi-robot task allocation (MRTA) fields. Both TAPs and LAPs are NP-hard, meaning that the difficulty of the problem increases as the size of the problem increases. On top of that, having more agents in the system leads to congestion or collision in the system, which causes higher travel costs or damage to the agents. Consequently, there is literature for TAPs and LAPs to include congestion and collision as part of the formulation and algorithm design. For instance, robot taxis search the shortest paths to complete ride demands while avoiding congestion in traffic [4], and warehouse robots deliver packages based on matching assignments from pods to trucks while avoiding collisions [5]. However, problem formulations and algorithms targeting all three aspects, routing, matching, and congestion or collision, have not been formally studied. Inspired by the ride-sharing and MRTA applications, this thesis focuses on the setup where a group of agents performs path planning from their locations to task locations determined from the optimal matching assignment while avoiding congestion or collisions. We formulated a convex optimization problem combining shortest path and task assignment problems with congestion. Also, we present a Frank-Wolfe-based algorithm [6] incorporating shortest path algorithms and the matching algorithm for balanced assignments. Moreover, the proposed method is extended to unbalanced assignments.

Contribution. The main contribution of the thesis is providing an optimization formulation and proposing a Frank-Wolfe-based [6] algorithm combining Dijkstra's [7] or A^* [8] algorithms, Hungarian (Kuhn-Munkres) [9] algorithm, and congestion-based potential functions to address shortest paths, balanced task assignments, and congestion avoidance simultaneously. A secondary contribution is extending our algorithm to unbalanced assignments with a linear program design for the matching assignment part of the algorithm. We show equivalence between the Wardrop equilibria [10] and the global solution of the potential minimization problem with proofs and interpretation of the dual variables. Apart from that, we define cost functions to allow agents to pick congestion-free paths and converge to Wardrop equilibrium, known as user equilibrium, or social Wardrop equilibrium, known as system optimal [10]. Finally, our model and algorithm are demonstrated on the street network of Seattle City where the agents are drivers delivering

packages to optimally assigned destinations on the shortest paths while avoiding congestion in traffic and on a 2D autonomous warehouse problem where the agents are robots being assigned to optimal delivery locations and navigating on shortest paths while avoiding congestion and extra turn penalty.

Organization. The rest of the thesis is arranged as follows. Section 2 presents the related work in TAPs and LAPs. Section 3 defines the problem setup in the context of the street network of Seattle City and the multi-robot warehouse system. Section 4 describes proofs for previous optimization formulations of routing, matching, and congestion problems. Then, we illustrate the proposed optimization problem formulation with proofs and dual variables interpretation. Section 5 outlines the shortest path algorithms, such as Dijkstra's [7] and A^* [8] algorithms, matching algorithms, such as the Hungarian (Kuhn-Munkres) algorithm [9], and the Frank-Wolfe algorithm [6] separately. Next, we introduce the proposed Frank-Wolfe-based algorithm combined with the shortest path and matching algorithms. Section 6 demonstrates the simulation results of the proposed algorithm. Finally, Section 7 concludes the thesis with an analysis of the results and potential directions for future work.

2 Related Work

In this section, we discuss prior studies and related work this thesis builds upon to provide background knowledge for analyzing results shown in Section 6. The section provides an overview of the literature that covers TAPs and LAPs with congestion and extension to unbalanced assignments.

2.1 Traffic Assignment Problems with Congestion

Recent work in TAPs considers congestion in the network and its impact on finding the optimal routes, specifically in ride-sharing and multi-robot task assignment applications. The paper [4] studied and incorporated congestion in the shared autonomous vehicle routing problem, where autonomous taxis are shared by several passengers to provide point-to-point transportation. The authors formulated the problem as a linear program with dial-a-ride service constraints, such as considering the number of passengers or vehicles to be routed and scheduled, time windows, and user preferences for system optimal dynamic traffic assignment. Network design for ride-sharing with the use of toll lanes and its pricing has been studied in [11] to relieve congestion and achieve ride-sharing user equilibrium [12, 13], where each user is on the optimal route, and switching to an unused route increases the travel cost. Similarly, [14] developed a model to tackle the congestion effect caused by e-hailing and achieve Wardrop equilibrium for solo drivers and e-hailing service providers in the network with several components formulated as optimization problems. On top of that, the paper [15] addressed the integrated task, which combines path planning and task assignment problems for large-scale robot networks with uncertainties such as temporary motion and communication failures of the robot caused by inaccurate conflict prediction. With the assumption of having an independent task assignment algorithm, the authors focused on obtaining the optimal routing solution with congestion and collision avoidance for each task-robot assignment part. Particularly, they designed a conflict graph to show the travel cost of the path planning for each task-robot assignment while combining the predicted path conflicts of any two assignments. Then, an integrated optimization problem is formulated to minimize the total travel cost while avoiding congestion and collisions in the network and solved using the greedy method. The aforementioned literature inspires our problem formulations and congestion cost design to obtain optimal routing with congestion and collision avoidance. Moreover, our work is an extension of [15], where we solve for both the optimal matching assignment and routing with a given set of agents and tasks.

2.2 Traffic Assignment and Linear Assignment Problems

A specific application discussed in this thesis is the MRTA problem, which aims to find the optimal assignment and path planning between the robots and tasks. The solution to MRTA problems can be found through a centralized planner by computing all robot-to-task assignment costs, such as expected travel costs, to determine the optimal assignment [16]. Consequently, task allocation and coordination are two fundamental problems in multi-robot systems. Typically, the multi-robot systems are modeled with task allocation and coordination separated into layers, so recent work explores various formulations and algorithm designs to concurrently handle both problems at the same level. However, there has not been much literature on finding the optimal routing and matching solutions while avoiding congestion and collision at the same time. The paper [17] proposed a novel framework called Co-MutaR to tackle multi-robot task allocation and coordination together while enabling robots to perform multiple tasks concurrently based on the Contract Net Protocol as a task allocator to form coalitions of actions. However, their framework did not include the impact of congestion on assignment and routing costs. As for [2], the authors proposed a centralized system to optimize task assignment and shared resources, which are the routes to the assigned destinations. Because the cost of performing the tasks varies depending on different traveled routes, they introduced a penalization term to avoid interference and congestion from agents sharing the same routes. Nonetheless, their algorithm mainly focused on the task assignment component without path planning because the routing costs were assumed to be given in their problem setup. While the work from [18] focused more on the path planning component using a proposed local cooperative A^* algorithm [8] incorporated with the conflict-based searching strategy. [19] designed an algorithm solving the shortest path and matching assignment problems simultaneously by combining a task allocation algorithm based on market auctions and an improved A^* shortest path algorithm with collision avoidance. Similarly, [20] incorporated the Hungarian (Kuhn-Munkres) algorithm [9] for task assignment and Dijkstra's algorithm [7] for shortest path problems. The above literature motivates our problem formulation incorporating task assignment and path planning with congestion and collision avoidance for the given set of agents and tasks. Particularly, [2] leads to our congestion cost design with an extra penalty term to prevent robots from changing directions when navigating to assigned tasks, causing higher travel costs and times. Furthermore, we extend the work from [19] and [20] by designing a Frank-Wolfe-based [21] algorithm combining Hungarian (Kuhn-Munkres) algorithm to solve for optimal matching with A^* algorithm or Dijkstra's algorithm to solve for optimal routing.

3 Problem Setup

In this section, we briefly review TAPs and LAPs with congestion included and discuss the framework of our problem setup in the context of the street network of Seattle City and the 2D autonomous multi-robot warehouse system.

3.1 Traffic Assignment Problems with Congestion

TAPs focus on finding optimal paths for vehicles to travel from origins to destinations while satisfying constraints such as the number of available vehicles and drivers [22]. Therefore, TAP is a combinatorial optimization problem with the objective of minimizing the total travel costs, such as commute time, routing costs, and congestion costs. In graph theory, TAPs can be modeled as a graph consisting of nodes connected by edges with different weights and can be solved by searching for the shortest paths between origins and destinations from the graph structure. This introduces the shortest path problem, which aims to find a path between the source node and the destination node that minimizes the sum of weights (costs) associated with the edges traveled [23].

Consider a path connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. The edge flow vector, $x \in \mathbb{R}^{|\mathcal{E}|}$, indicates the amount of traffic flow in each edge and has an associated non-decreasing and continuous latency function in each edge, $\ell_e(x_e)$. At some points in this work, we will impose the following extra assumption on $\ell_e(\cdot)$.

Assumption 1. *$\ell_e(\cdot)$ is strictly increasing in its argument.*

Given this assumption, the inverse latency function $\ell_e^{-1}(\cdot)$ will be well-defined and strictly increasing as well. We can organize the latency functions on each edge into a vector denoted $\ell(x) \in \mathbb{R}^{|\mathcal{E}|}$ and consider a set of paths $\mathcal{P}$ from an origin to a destination. For any path $P \in \mathcal{P}$, we can define an indicator vector $\xi \in \mathbb{R}^{|\mathcal{E}|}$ that tells which edges are used in that path.

$$(\xi_P)_e = \begin{cases} 1 & \text{; if edge } e \text{ is in path } P \\ 0 & \text{; otherwise} \end{cases}$$

The latency function of path P is given by the sum of the edge latencies $\ell_P(x) = \sum_{e \in P} \ell_e(x_e)$. Note

that given the indicator vector for the path, we can write this as

$$\ell_P(x) = \sum_{e \in P} \ell_e(x_e) = \ell(x)^\top \xi_P \tag{1}$$

Often in these problems, we consider paths that correspond to different population of people (sometimes called commodities), often distinguished by different origin-destination pairs or other distinguishing features. For a set of populations $\mathcal{I}$, we can distinguish the flow for each population $i \in \mathcal{I}$ using a superscript. Thus, $x^i \in \mathbb{R}^{|\mathcal{E}|}$ is the flow on the edges from population i. We denote the set of paths for each population as $\mathcal{P}^i$. If there are multiple populations that all use the same edges then the total edge flow x will be given by

$$x = \sum_i x^i, \tag{2}$$

When it comes to designing the cost function, the cost function is linear, with no congestion present in the system. To include congestion, the problem can be modeled as a potential game, with the cost function being a potential function described in Definition 1.

Definition 1. *(Potential Game) [24]. In game theory, a game is said to be a potential game if the incentive of all players to change their strategy can be expressed using a single global function called the potential function. The potential function in our problem is defined as [25]*

$$\phi(x) = \sum_{e \in E} \int_0^{x_e} \ell_e(s)\, ds, \quad \textit{where} \quad \frac{\partial \phi(x)}{\partial x_e} = \ell_e(x_e) \tag{3}$$

The potential function derived from private costs allows agents to achieve Wardrop equilibrium, also known as user equilibrium, where each agent selfishly picks the best route, and their travel cost increases when switching to unused routes. Therefore, a flow is stable if and only if all used paths have the same minimal latency, whereas unused paths may have larger latency at user equilibrium.

Definition 2. *(Wardrop Equilibrium) [12,13] A set of feasible flow vectors $\{x^i\}_{i \in \mathcal{I}}$ is at Wardrop equilibrium if for every population $i \in \mathcal{I}$ and two feasible paths for that population $P, P' \in \mathcal{P}^i$ such that some nonzero*

portion of the population chooses path P, then

$$\ell_P(x) \leq \ell_{P'}(x) \tag{4}$$

From the edge flow perspective, if mass from population i chooses path P, then $x_e^i > 0$ for every $e \in P$. Therefore, we can write the above definition more explicitly in terms of the edge flows. For paths $P, P' \in \mathcal{P}^i$ with indicator vectors ξ, ξ' respectively

$$\ell_P(x) = \ell(x)^\top \xi \leq \ell(x)^\top \xi' = \ell_{P'}(x) \tag{5}$$

when $\text{supp}(\xi) \subseteq \text{supp}(x^i)$ where $\text{supp}(\cdot)$ refers to the support of a mass distribution.

By minimizing the potential function, the agents reach the Wardrop equilibrium, where all agents choose the route that is optimal for them, and no agent can improve their cost by switching routes. The population is likely not at the socially optimal solution for all agents (best average solution). Therefore, no agent can personally improve their cost, and they will not switch routes. If we want to find the socially optimal solution instead of the Wardrop equilibrium for each agent, we can minimize the social cost.

Definition 3. *(Social Wardrop Equilibrium) [10]. A feasible flow vector x is at social optimum if it minimizes the social cost, $\sum_{e \in E} x_e \ell_e(x_e)$.*

To reach system optimal, economists and modelers applied marginal cost road pricing as Pigouvian taxes to facilitate socially optimal routing solutions. Based on Definition 4, the marginal cost function [26] of a latency function shows the rate of change to total latency caused by flow changes in the edge. Specifically, the second term of the marginal cost function indicates the increase in travel time caused by adding one unit of flow, i.e., $\ell'(x)$, multiplied by all the flow, x, that suffers from this increase [27]. As a result, the Pigouvian taxes are implemented as road tolls in our problem based on the marginal cost function to ensure socially optimal routing instead of selfish routing.

Definition 4. *(Marginal Cost Function) [28]. If ℓ is continuous and differentiable, the marginal cost is*

$$\ell^*(x) = \frac{\partial}{\partial x}(\ell(x) * x) = \ell(x) + \ell'(x)x \tag{6}$$

Definition 5. *(Pigouvian Taxes) [29]. A Pigouvian tax is a flow-dependent toll added to each edge to regulate socially harmful activities and prevent the marginal private interest from dominating the marginal social interest. For edge latency $\ell_e(x_e)$ this toll is given by $\ell'_e(x_e)x_e$*

3.2 Linear Assignment Problems

LAPs concentrate on finding the optimal assignment between a given set of agents and tasks while satisfying constraints such as balanced assignments where one agent is assigned to at most one task [30]. As a result, LAP is a combinatorial optimization problem that minimizes the total cost of the assignment, which is the sum of the costs for each agent performing the assigned task. In graph theory, the linear assignment problem is also the minimum weighted sum bipartite matching problem with a goal of obtaining a matching for a minimum sum of weights of the edges [31].

Consider $\mathcal{A}$ to be a set of agents and $\mathcal{T}$ to be a set of independent tasks. Assuming that the assignment of agent i to task j incurs a cost c_{ij}, we can construct a cost matrix, C, to show the assignment cost between each agent and each task. When the number of agents is the same as the number of tasks, C is a square matrix. Therefore, the overall cost function can be written as $\text{Tr}(C^\top M)$ where $M \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$ is a permutation matrix that encodes the probability, M_{ij}, of agent i performing task j, and gives the optimal assignment indicated by non-zero elements.

Definition 6. *(Permutation Matrix). A permutation matrix is a square matrix whose entries are all either 0 or 1, and which contains exactly one 1 entry in each row and each column, e.g.* $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Permutation matrices are not convex sets. However, we can write convex relaxation of LAPs by relaxing the feasible set of permutation matrices to the set of doubly stochastic matrices, where the rows and the columns sum to 1 as shown in Definition 7. According to Birkhoff's theorem 1, doubly stochastic matrices are the convex hull of permutation matrices.

As we will show explicitly in Section 4.2 and specifically Problem (12), this convex relaxation will be a linear program. For a generic set of costs C, the optimizer of the linear program will be at a vertex, which will correspond to a permutation matrix, and thus (generically) this convex relaxation is exact.

Definition 7. *(Doubly Stochastic Matrix). A square matrix is doubly stochastic if all its entries are non-negative and the sum of the entries in any of its rows or columns is 1, i.e.,* $M\mathbf{1} = \mathbf{1},\ M^\top \mathbf{1} = \mathbf{1}$.

Definition 8. *(Convex Hull) [32]. A set of points in an Euclidean space is defined to be convex if it contains the line segments connecting each pair of its points. The convex hull of a given set X is the smallest convex set containing X.*

Theorem 1. *(Birkhoff–von Neumann Theorem) [33]. Every doubly stochastic matrix is a convex combination of permutation matrices.*

3.3 Proposed Problem

Our problem statement merges the shortest path and matching problems with congestion and further extends it to unbalanced assignments. We first introduce the framework in the street network of Seattle City and then move on to the autonomous multi-robot warehouse in a grid world.

3.3.1 Street Network Problem

Consider a path connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ of the street network of Seattle City. The edges are driving roads, and the nodes are the intersections of roads, as shown in Figure 1. Let $\mathcal{A}$ be a set of agents and $\mathcal{T}$ be a set of independent tasks. N_a of agents and N_t of tasks are initialized as random nodes on the Seattle City map. The agents are the drivers delivering packages to various warehouse locations as the tasks. In addition, the goal of the agents is to minimize the overall routing and matching cost by obtaining optimal one-to-one matching and traveling through the shortest paths to the assigned warehouse location while avoiding congestion in traffic as much as possible.

We model the whole problem by combining the above TAP and LAP setups with congestion. First, the potential function in Equation (1) is set as the objective cost function, and solving the minimization of the potential function ensures agents reach Wardrop equilibria. Then, the assignment cost, c_{ij}, of agent i and task j is calculated as the shortest path length with congestion included between the location of agent i and the location of task j. After populating the overall cost matrix, C, we can solve for the optimal matching assignment, M, which encodes the corresponding shortest path traveled by agents to optimally assigned tasks. To connect the matching problem to the routing problem, we add M_{ij} amount of mass to edges

traveled in the shortest paths by the agents and thus incorporate the optimal matching information when solving for optimal edge flow. As a result, our problem setup allows us to obtain both the optimal edge flow and matching assignment solutions and avoid congestion by reaching Wardrop equilibria.

Figure 1: Street Network of Seattle City

For unbalanced assignments, the cost matrix, C, and the optimal matching matrix, M, become non-square. Consequently, the M matrix is no longer a permutation matrix and is not doubly stochastic. Regardless, the unbalanced assignment problems can still be formulated as an optimization problem where the M matrix has multiple non-zero elements. The constraints for the M matrix are the columns sum to 1, meaning that a task is completed by all agents, and the rows sum to the ratio of the number of tasks divided by the number of agents, meaning that the agents equally split the tasks. Thus, the main difference observed with one-to-N matching is having more distributed probability in the optimal matching M matrix, and the same street network problem setup works for unbalanced assignments.

3.3.2 Multi-Robot Warehouse Problem

Consider a two-dimensional grid graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. The nodes are the grids of a warehouse floor with edges connecting each node to its four nearest neighbors, as shown in Figure 2. Let $\mathcal{A}$ be a set of agents and $\mathcal{T}$ be a set of independent tasks. N_a of agents and N_t of tasks are initialized as random nodes on the grid map. The agents are the robots delivering packages to various delivery locations as the tasks. The delivery locations are placed on the boundary of the grid map while the agents are placed at random non-boundary nodes. In addition, the goal of the agents is to minimize the overall routing and matching cost by obtaining optimal one-to-one matching and traveling through the shortest paths to the assigned delivery location while avoiding collision. Moreover, we would like to minimize the number of turns the robot makes to improve efficiency because it takes time for the robot to change its moving direction.

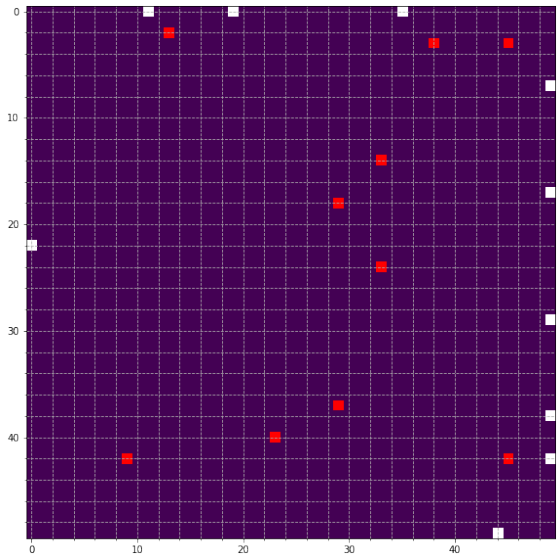


Figure 2: 50×50 Multi-Robot Warehouse Floor

The overall problem setup for the multi-robot warehouse system is similar to the setup for the street network of Seattle City. The key difference is to keep track of the number of turns robots make and incorporate the information with the matching and routing problems. We first expand all the nodes in the grid graph in five directions, which are center, east, west, north, and south. Each node is relabeled with the coordinate and the direction it plans to travel next, i.e., (0, 0, W), meaning that the robot is at coordinate (0,0) and going to travel west to the node at coordinate (0, 1). The edges of the graph are expanded as well to connect nodes based on the additional direction information. With the new grid graph setup, we can modify the previously

defined potential function by adding a turn penalty term to prevent agents from making many turns when navigating to the delivery locations.

With the additional turn penalty term, the multi-robot warehouse problem setup addresses the routing, matching, and congestion avoidance aspects together. The optimal solutions are obtained at Wardrop equilibria, and the setup can be extended to solve unbalanced assignments with the same method mentioned for the street network setup.

4 Background Problem Formulations

This section first discusses the linear program and convex formulations of previous results on routing games and then talks about the linear program formulation of previous results on matching problems.

4.1 Routing Formulations

As mentioned in Section 3, we can model routing games as a directed graph with nodes connected by edges having directions associated with them based on graph theory. Here, we consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ and define the node-edge graph incidence matrices $E_{\text{i}}, E_{\text{o}}, E \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ as follows:

$$\left[\, E_{\text{i}} \,\right]_{ve} = \begin{cases} 1 & ; \text{ if edge } e \text{ goes into node } v \\ 0 & ; \text{ otherwise} \end{cases}$$

$$\left[\, E_{\text{o}} \,\right]_{ve} = \begin{cases} 1 & ; \text{ if edge } e \text{ comes out of node } v \\ 0 & ; \text{ otherwise} \end{cases}$$

and $E = E_{\text{i}} - E_{\text{o}}$ which encodes the relation of node-node pairs to represent the graph structure [12, 13]. The source-sink vector $S \in \mathbb{R}^{|\mathcal{V}|}$ indicates the source and sink nodes as follows:

$$S_v = \begin{cases} 1 & ; \text{ if node } v \text{ is a source} \\ -1 & ; \text{ if node } v \text{ is a sink} \\ 0 & ; \text{ otherwise} \end{cases}$$

The source node is where flow enters the network and the sink node is where flow exits the network.

4.1.1 Routing Linear Program Formulation

After defining the graph structure, we look at the routing game. In the case with constant edge costs (no congestion), the potential function in (1) becomes simply the linear function $c^\top x$, and we get a linear program for computing shortest paths with constant edge weights [22]. The primal linear program is presented first as it is formulated based on the routing game problem statement directly. Then, we show the derivation and formulation of the dual linear program and explain the meaning of dual variables and dual problem setup.

Finally, the proofs of this formulation based on previous results are demonstrated.

Primal:

$$\min_{x} \quad c^{\top} x \tag{7a}$$

$$\text{s.t.} \quad Ex = Sm,\ x \geq 0 \tag{7b}$$

where,

- $x \in \mathbb{R}^{|\mathcal{E}|}$: an edge flow vector that gives the portion of traffic flow on each edge.
- $c \in \mathbb{R}^{|\mathcal{E}|}$: a vector of constant costs for the edges.
- $E \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$: the incidence matrix of the graph.
- $S \in \mathbb{R}^{|\mathcal{V}|}$: a source-sink vector indicating where the flow enters and exits the network.
- $m \in \mathbb{R}$: the amount of population mass traveling through the network (often set to 1 for shortest path problems).

The primal linear program is a minimization problem with the objective function (7a) being the total travel cost from the source to the sink computed by multiplying the cost of each edge with the flow of each edge. The constraints (7b) are the conservation of flow where the amount of flow going into a node equals the amount of flow going out of the same node plus (or minus) any flow entering (or exiting) the graph at that node and each edge flow is non-negative. The term Sm indicates that m units of mass enter at the source and exit at the sink. Lastly, the solution is the optimal edge flow that achieves the minimum total travel cost from the source to the sink while satisfying the conservation of flow.

4.1.2 Linear Routing: Dual Derivation

The dual problem provides a different perspective of understanding the problem and its objective function allows us to find the lower bound on the primal minimization problem or the upper bound on the primal maximization problem. We derive the dual problem by minimizing the Lagrangian function of the primal

problem as follows:

$$\min_x \max_{v,\mu} \mathcal{L} = \min_x \max_{v,\mu \geq 0} c^\top x - v^\top (Ex - Sm) - \mu^\top x = \min_x \max_{v,\mu \geq 0} (c^\top - v^\top E - \mu^\top) x + v^\top Sm$$
$$\Rightarrow \max_{v,\mu \geq 0} \min_x (c^\top - v^\top E - \mu^\top) x + v^\top Sm = \max_{v,\mu \geq 0} \Big(\min_x (c^\top - v^\top E - \mu^\top) x \Big) + v^\top Sm$$

By solving for the inner minimization problem explicitly for x, i.e., taking the derivative with respect to x and setting it equal to 0, the dual objective function (8a) and constraints (8b) are obtained.

$$\frac{\partial \mathcal{L}}{\partial x} = c^\top - v^\top E - \mu^\top = 0 \Rightarrow \max_{v,\mu \geq 0} \Big(\min_x (c^\top - v^\top E - \mu^\top) x \Big) + v^\top Sm$$
$$= \max_{v,\mu \geq 0} \Big(\min_x 0 \Big) + v^\top Sm = \max_{v,\mu \geq 0} v^\top Sm$$

Dual:

$$\max_{v,\mu} \quad v^\top Sm \tag{8a}$$
$$\text{s.t.} \quad c^\top = v^\top E + \mu^\top, \ \mu^\top \geq 0 \tag{8b}$$

Here the dual variables have the following interpretaions that we will discuss later on.

- $v \in \mathbb{R}^{|\mathcal{V}|}$: a vector of the value function on the nodes encoding the cost-to-go from the origin to the destination.
- $\mu \in \mathbb{R}^{|\mathcal{E}|}$: a vector that denotes the inefficiency of using each edge.

4.1.3 Linear Routing: KKT Conditions

The rest of the subsection contains proofs for the routing game linear program formulation. We derive the Karush–Kuhn–Tucker (KKT) conditions by taking the derivative of the Lagrangian function with respect to the primal and dual variables and illustrate the solution obtained from the formulation is indeed optimal.

KKT Conditions:

$$\begin{aligned}
&\text{Feasibility:} && Ex = Sm,\ x \geq 0 \\
& && \left(\frac{\partial \mathcal{L}}{\partial v} = -Ex + Sm = 0 \Rightarrow Ex = Sm\right) \\
&\text{Stationarity:} && c^\top = v^\top E + \mu^\top,\ \mu^\top \geq 0 \\
& && \left(\frac{\partial \mathcal{L}}{\partial x} = c^\top - v^\top E - \mu^\top = 0 \Rightarrow c^\top = v^\top E + \mu^\top\right) \\
&\text{Complementary Slackness:} && 0 = \mu^\top x \iff 0 = \mu_e x_e,\ \forall e
\end{aligned}$$

Proof: To show that the primal problem above finds an optimal route from the source to the sink, let $\xi \in \mathbb{R}^{|\mathcal{E}|}$ be an indicator vector for route r, where

$$\xi_e = \begin{cases} 1 & ;\text{if edge } e \in \text{route r} \\ 0 & ;\text{otherwise} \end{cases}$$

If $\xi \geq 0$ is a feasible route from the source to the sink such that $E\xi = S$ based on the feasibility condition, we can take the stationarity condition and multiply it by ξ so that $c^\top \xi = v^\top E\xi + \mu^\top \xi$. Then, we can plug the feasibility condition $E\xi = S$ into the stationarity condition to derive

$$\underbrace{\sum_{e \in r} c_e}_{\text{total travel cost}} = c^\top \xi \text{ (objective function)} = v^\top E\xi + \mu^\top \xi \text{ (Stationarity condition)}$$

$$= v^\top S + \mu^\top \xi \text{ (feasibility condition)} = \underbrace{v_s - v_d}_{\text{optimal cost from source to sink}} + \underbrace{\mu^\top \xi}_{\geq 0\text{, inefficiency of route } r}$$

Note that the term $v_s - v_d$ is the same for *any* route from the source to the sink, and the term $\mu^\top \xi$ is positive. It follows from this that the total travel cost for any route is greater than or equal to $v_s - v_d$. Thus, $v_s - v_d$ is the optimal travel cost. This same intuition also shows that the elements of v encode optimal cost-to-go information at each node.

If the support of ξ is contained in the support of the flow vector x at optimum, then $\xi_e > 0$ only if $x_e > 0$. Consequently, $\mu^\top x = 0 \implies \mu^\top \xi = 0$, which proves that ξ is an optimal and efficient route if and only if it's support is contained in the support of x.

4.1.4 Linear Routing: Dual Interpretation

Here, we give several further thoughts on intuition for the dual program. For the dual linear program, we can observe that the objective function (8a) represents the minimum distance from the source to the sink

$$v^\top Sm = m(v_{\text{source}} - v_{\text{sink}})$$

scaled by the value of the mass m. This shows that the goal of the dual problem is to maximize the lower bound of total travel cost from the source to the sink in the primal problem. The constraints (8b) describe the edge cost as difference in the value functions between the start and end node plus the inefficiency of taking the same edge (where the inefficiency is non-negative). Explicitly,

$$c_e = v_j - v_i + \mu_e$$

where edge e goes from node i to node j. When optimizing the dual function, we vary the elements of v to maximize the difference between the source and sink nodes while making sure that $v_j - v_i$ is always a lower bound on the cost of traveling each edge e, i.e., $\mu_e \geq 0$. By maximizing the dual objective, we are trying to increase the difference between v_{source} and v_{sink} as much as possible. The set of edges that eventually prevent us from further increasing this difference are the edges with minimum cumulative cost from the source to the sink. The dual variables indicate the shortest path by complementary slackness, i.e., $\mu_e = 0$ for any edge in the optimal route.

4.1.5 Routing Game Convex Formulation

This subsection focuses on the routing game formulated as a convex optimization problem with a non-constant latency cost associated with each edge. This shortest path with congestion model is the center of the traffic assignment problem. The formulation here was first introduced in [13]. A review of the literature and many other references are found in [34]. After discussing the primal problem, we derive the dual problem with an explanation of the meaning of the dual variables and solution. Then, we show the proofs of this formulation based on previous results.

4.1.6 Congestion Routing: Primal Formulation

We now give the primal formulation for the routing game.

Primal:

$$\min_{x} \quad \sum_{e} \int_{0}^{x_e} \ell_e(s)\, ds \tag{9a}$$

$$\text{s.t.} \quad Ex = Sm,\ x \geq 0 \tag{9b}$$

where,

- $x \in \mathbb{R}^{|\mathcal{E}|}$: an edge flow vector that gives the portion of traffic flow on each edge.
- $\ell(x) \in \mathbb{R}^{|\mathcal{E}|}$: a non-decreasing and continuous latency cost function for each edge as a function of x.
- $E \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$: the incidence matrix of the graph.
- $S \in \mathbb{R}^{|\mathcal{V}|}$: a source-sink vector indicating where the flow enters and exits the network
- $m \in \mathbb{R}$: the amount of population mass traveling through the network (often set to 1 for shortest path problems).

The primal convex problem is a minimization problem with the objective function (9a) being the potential function (1) to account for congestion. The constraints (9b) are the conservation of flow where the amount of flow going into a node equals the amount of flow going out of the same node plus (or minus) any flow entering (or exiting) the graph at that node and each edge flow is non-negative. The term Sm indicates that m units of mass enter at the source and exit at the sink. Lastly, the solution is the optimal edge flow that achieves the minimum total travel cost from the source to the sink while satisfying the conservation of flow.

4.1.7 Congestion Routing: Dual Derivation

For the derivation of the dual, we will make use of Assumption 1 that the latency functions $\ell_e(\cdot)$ are strictly increasing and thus invertible. The dual objective function in this formulation allows us to find the lower bound on the primal minimization problem, and we derive the dual problem by minimizing the Lagrangian

function of the primal problem as follows:

$$\begin{aligned}
\min_x \max_{v,\mu} \mathcal{L} &= \min_x \max_{v,\mu \geq 0} \left(\sum_e \int_0^{x_e} \ell_e(s)\, ds \right) - v^\top (Ex - Sm) - \mu^\top x \\
&\Rightarrow \max_{v,\mu \geq 0} \min_x \left(\sum_e \int_0^{x_e} \ell_e(s)\, ds \right) - v^\top (Ex - Sm) - \mu^\top x \\
&\Rightarrow \max_{v,\mu \geq 0} \left[\min_x \left(\sum_e \int_0^{x_e} \ell_e(s)\, ds \right) - (v^\top E + \mu^\top) x \right] + v^\top Sm \\
&\Rightarrow \max_{v,\mu \geq 0} \left[\min_x \sum_e \left(\int_0^{x_e} \ell_e(s)\, ds - \Big(\big[v^\top E \big]_e + \mu_e \Big) x_e \right) \right] + v^\top Sm
\end{aligned}$$

We solve for the inner minimization problem explicitly for x, i.e., taking the derivative with respect to x and setting it equal to 0, to obtain the dual objective function (11a) and constraints (11b). The derivative condition is given by

$$\frac{\partial \mathcal{L}}{\partial x_e} = \ell_e(x_e) - \big[v^\top E \big]_e - \mu_e = 0 \;\Rightarrow\; \ell_e(x_e) = \big[v^\top E \big]_e + \mu_e$$

Defining new dual variables $c_e = \big[v^\top E \big]_e + \mu_e$ and plugging in above gives.

$$\max_{v,\mu \geq 0} \left[\min_x \sum_e \left(\int_0^{x_e} \ell_e(s)\, ds - c_e x_e \right) \right] + v^\top Sm = \max_{c,v,\mu \geq 0} v^\top Sm - \sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds$$

where on the right hand side, we have applied the Legendre transform detailed in Proposition 1 below.

Proposition 1 (Legendre Transform of Routing Game Potential)**.**

$$\min_{x_e} \sum_e \underbrace{\int_0^{x_e} \ell_e(s)\, ds}_{(1)} - \underbrace{c_e x_e}_{(2)} = - \sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \tag{10}$$

Taking the derivative of the LHS and setting it equal to 0 gives $\ell_e(x_e) - c_e = 0$ for each edge e. Given

Assumption 1, this gives that $x_e = \ell_e^{-1}(c_e)$. We now make an argument based on integration by parts.

$$\text{Substitutions:} \quad w = \ell_e(s) \;\Rightarrow\; \partial w = \ell_e'(s)ds, \quad \partial v = \partial s \;\Rightarrow\; v = s$$

$$\begin{aligned}
\text{Integration by Parts for (1):} \quad \underbrace{\int_0^{\ell_e^{-1}(c_e)} \ell_e(s)\, ds}_{(1)} &= wv - \int v dw \\
&= \left(\ell_e(s)s|_{s=0}^{s=\ell_e^{-1}(c_e)}\right) - \int_0^{\ell_e^{-1}(c_e)} s\ell_e'(s)\, ds \\
&= \ell_e(\ell_e^{-1}(c_e))\ell_e^{-1}(c_e) - \int_0^{\ell_e^{-1}(c_e)} s\ell_e'(s)\, ds \\
&= \underbrace{c_e\ell_e^{-1}(c_e) - \int_0^{\ell_e^{-1}(c_e)} s\ell_e'(s)\, ds}_{(3)}
\end{aligned}$$

Plugging in $x_e = \ell_e^{-1}(c_e)$ and (3) into the LHS of (10) gives

$$\begin{aligned}
\text{(LHS)} &= \sum_e \Bigg(\underbrace{c_e\ell_e^{-1}(c_e) - \int_0^{\ell_e^{-1}(c_e)} s\ell_e'(s)\, ds}_{(3)} - \underbrace{c_e\ell_e^{-1}(c_e)}_{(2)} \Bigg) \\
&= -\sum_e \int_0^{\ell_e^{-1}(c_e)} s\ell_e'(s)\, ds \\
&= -\sum_e \int_0^{\ell_e^{-1}(c_e)} s\, d\ell_e(s) \quad \left(s\ell_e'(s)\, ds = s\, d\ell_e(s)\right) \\
&= -\sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \qquad (\text{let } s = \ell_e(s))
\end{aligned}$$

The dual problem is then given by

Dual:

$$\max_{c,v,\mu} \quad v^\top Sm - \sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \tag{11a}$$

$$\text{s.t.} \quad c^\top = v^\top E + \mu^\top,\ \mu^\top \geq 0 \tag{11b}$$

Here the dual variables have the following interpretations.

- $v \in \mathbb{R}^{|\mathcal{V}|}$: a vector of the value function on the nodes encoding the cost-to-go from the origin to the destination.

- $\mu \in \mathbb{R}^{|\mathcal{E}|}$: a vector that denotes the inefficiency of using each edge.

4.1.8 Congestion Routing: KKT Conditions

The rest of the subsection derives the KKT conditions to prove the solution obtained is optimal for the routing game convex problem formulation.

KKT Conditions:

$$\begin{aligned}
&\text{Feasibility:} && Ex = Sm,\ x \geq 0 \\
& && \left(\frac{\partial \mathcal{L}}{\partial v} = -Ex + Sm = 0 \Rightarrow Ex = Sm\right) \\
&\text{Stationarity:} && \ell(x)^\top = v^\top E + \mu^\top,\ \mu^\top \geq 0 \\
& && \left(\frac{\partial \mathcal{L}}{\partial x} = \ell(x)^\top - v^\top E - \mu^\top = 0 \Rightarrow \ell(x)^\top = v^\top E + \mu^\top\right) \\
&\text{Complementary Slackness:} && 0 = \mu^\top x \iff 0 = \mu_e x_e,\ \forall e
\end{aligned}$$

Proof: To show that the primal problem above finds a traffic flow pattern x where all routes the population uses from the source to the sink have equal minimal travel time, let $\xi \in \mathbb{R}^{|\mathcal{E}|}$ be an indicator vector for route r, where

$$\xi_e = \begin{cases} 1 & ;\text{if edge } e \in \text{route r} \\ 0 & ;\text{otherwise} \end{cases}$$

If $\xi \geq 0$ is a feasible route from the source to the sink such that $E\xi = S$ based on the feasibility condition similar to the linear program case, we can take the stationarity condition and multiply it by ξ so that $\ell(x)^\top \xi = v^\top E\xi + \mu^\top \xi$. Then, we can plug the feasibility condition $E\xi = S$ into the stationarity condition to derive

$$\underbrace{\sum_{e \in r} \ell_e(x_e)}_{\text{total travel cost}} = \ell(x)^\top \xi \text{ (objective function)} = v^\top E\xi + \mu^\top \xi \text{ (Stationarity condition)}$$

$$= v^\top Sm + \mu^\top \xi \text{ (feasibility condition)} = \underbrace{v_s - v_d}_{\text{optimal cost from source to sink}} + \underbrace{\mu^\top \xi}_{\geq 0, \text{ inefficiency of route } r}$$

The only difference here with the non-congested case is that the edge costs $\ell(x)$ now depend on the flow x. As in the uncongested case, $\mu^\top \xi = 0$ if and only if the support of ξ is contained in the support of the population flow vector x. Consequently, $\mu^\top x = 0 \implies \mu^\top \xi = 0$, which proves that ξ is an optimal and efficient route if and only if its support is contained in the support of x. Moreover, this condition is equivalent to Equation 5 in the definition of a Wardrop equilibrium (Definition 2). For two paths P, P' from the source to the sink with indicator vectors ξ, ξ' respectively where the support of ξ is in the support of x

$$\ell_P(x) = \ell(x)^\top \xi = \ell(x)^\top \xi' + \mu^\top \xi' \leq \ell(x)^\top \xi' = \ell_{P'}(x)$$

Intuitively, this means that any path taken by agents in the population is optimal. Due to the congestion effects encoded in $\ell(x)$, the population mass will often spread out over multiple routes; however, the analysis above guarantees that all those routes will have equal travel time at equilibrium.

4.1.9 Congestion Routing: Further Dual Interpretation

To get some intuition for the dual optimization problem, we note the following. The dual problem is the same as the non-congestion version of the Problem (8) except that c is an optimization variable (as opposed to constant), and there is also the additional sum penalty term in the objective.

The same intuition applies when we are trying to increase the term $v^\top Sm$. Since the source-sink vector, S, and mass on nodes, m, are fixed, the value of this term increases when the difference of v between the source and the sink becomes larger. Now, rather than c_e providing a hard limit on how much the elements of v can differ along edge e, we are allowed to increase the value c_e. We will, however, pay a penalty term in the objective determined by the term $\int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds$.

Since the variable c must satisfy the equality constraint (8b), we can first increase the value of v by

decreasing μ as much as possible. However, for the edges where $\mu_e = 0$, we have to start increasing c_e to further increase the difference in v along that edge and pay the penalty defined by the integral term above. Intuitively, we can see that for positive $\ell_e(\cdot)$ this integral term gets larger as the upper bound increases, i.e., $\int_{\ell_e(0)}^{\ell_e(0)} \ell_e^{-1}(s)\, ds = 0$ and $\int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds > 0$ for $c_e > \ell_e(0)$. Thus, if we are to increase c_e above its non-congestion value $\ell_e(0)$, we may be able to increase the difference in the elements of v, but we will pay a penalty determined by $-\int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds$.

At optimum, a balance is achieved where we increase the difference defined by $v^\top Sm$ as much as possible while increasing the values of c_e on as few edges as possible. This process has the effect of spreading out which edges are used across the graph. From the KKT analysis above, we can see that this balance is precisely such that the costs on the edges where $\mu_e = 0$ form constant cost optimal paths from the source to the sink. Therefore, intuitively we can think of the purpose of the dual problem as maximizing the lower bound of the total travel cost from the source to the sink in the primal problem while minimizing the congestion impact on each edge's latency cost.

4.2 Matching Linear Program Formulation

The following linear program with $\mathcal{A}$ as a set of agents and $\mathcal{T}$ as a set of tasks is the convex relaxation that solves the linear sum assignment problem, where the numbers of agents and tasks are equal. The variables in the dual problem correspond to the potentials from the well-known Hungarian algorithm [5]. The section presents the primal linear program and shows the derivation and formulation of the dual linear program with the meaning of dual variables and the dual problem setup provided. Finally, the proofs of this formulation based on the paper [35] are discussed.

4.2.1 Matching: Primal Formulation

The primal formulation for the matching problem can be given as follows.

Primal:

$$\min_{M} \quad \text{Tr}(C^\top M) \tag{12a}$$

$$\text{s.t.} \quad M\mathbf{1} = \mathbf{1},\ M^\top \mathbf{1} = \mathbf{1}, M \geq 0 \tag{12b}$$

where,

- $M \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$: a doubly stochastic matrix showing the probability, M_{ij}, of agent i performing task j.
- $C \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$: the assignment cost matrix between agent i and task j.

The primal linear program is a minimization problem with the objective function (12a) being the overall assignment cost by taking the trace of the product of the cost matrix and the matching assignment matrix. The constraints (12b) ensure M matrix is doubly stochastic in Definition 7 with non-negative elements. On top of that, the constraints confirm each agent is assigned to only one task. Consequently, the solution gives an optimal one-to-one matching between agents and tasks with the minimum overall assignment.

4.2.2 Matching: Dual Derivation

The dual objective function in this formulation allows us to find the lower bound on the primal minimization problem, and we derive the dual problem by switching the min and max and enforcing the minimization condition for M as a constraint as follows:

$$
\begin{aligned}
\min_{M}\max_{u,w,U\geq 0}\ \mathcal{L} &= \min_{M}\max_{u,w,U\geq 0}\ \text{Tr}(C^\top M) - u^\top(M\mathbf{1}-\mathbf{1}) - w^\top(M^\top\mathbf{1}-\mathbf{1}) - U^\top M \\
&\Rightarrow \max_{u,w,U\geq 0}\min_{M}\ \text{Tr}(C^\top M) - u^\top M\mathbf{1} + u^\top\mathbf{1} - w^\top M^\top\mathbf{1} + w^\top\mathbf{1} - U^\top M \\
&\Rightarrow \max_{u,w,U\geq 0}\min_{M}\ \text{Tr}(C^\top M) - \text{Tr}(u^\top M\mathbf{1}) - \text{Tr}(w^\top M^\top\mathbf{1}) - U^\top M + (u^\top\mathbf{1} + w^\top\mathbf{1}) \\
&\Rightarrow \max_{u,w,U\geq 0}\left[\min_{M}\text{Tr}(C^\top M) - \text{Tr}(\mathbf{1}u^\top M) - \text{Tr}(\mathbf{1}w^\top M^\top) - U^\top M\right] + (u^\top\mathbf{1} + w^\top\mathbf{1})
\end{aligned}
$$

where we have used the cyclic property of the trace, ie. $\text{Tr}(ABC) = \text{Tr}(CAB) = \text{Tr}(BCA)$.

By solving for the inner minimization problem explicitly for M, i.e., taking the derivative with respect to M and setting it equal to 0, the dual objective function (13a) and constraints (13b) are obtained.

$$
\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial M} = C - u\mathbf{1}^\top - \mathbf{1}w^\top - U = 0 \ \Rightarrow\ C = u\mathbf{1}^\top + \mathbf{1}w^\top + U \\
&\Rightarrow \max_{u,w,U\geq 0}\left[\min_{x}\text{Tr}\Big(C - u\mathbf{1}^\top - \mathbf{1}w^\top - U\Big)^\top M\Big)\right] + (u^\top\mathbf{1} + w^\top\mathbf{1}) \\
&\Rightarrow \max_{u,w,U\geq 0}\Big(\min_{x} 0\Big) + (u^\top\mathbf{1} + w^\top\mathbf{1}) \Rightarrow \max_{u,w,U\geq 0} u^\top\mathbf{1} + w^\top\mathbf{1}
\end{aligned}
$$

The dual problem is then given by the following.

Dual:

$$\max_{u,w,U} \quad u^\top \mathbf{1} + w^\top \mathbf{1} \tag{13a}$$

$$\text{s.t.} \quad C = u\mathbf{1}^\top + \mathbf{1}w^\top + U,\ U \geq 0 \tag{13b}$$

where,

- $u \in \mathbb{R}^{|\mathcal{A}|}$: a vector of row potential as the minimum cost each agent needs to perform at least one task.
- $w \in \mathbb{R}^{|\mathcal{T}|}$: a vector of column potential as the additional minimum cost on top of u each task required.
- $U \in \mathbb{R}_+^{|\mathcal{A}|\times|\mathcal{T}|}$: a matrix that denotes the inefficiency of pairing agent i with task j.

4.2.3 Matching: KKT Conditions

The rest of the subsection contains the KKT conditions to prove the solution obtained is optimal for the matching linear problem formulation.

KKT Conditions:

$$\begin{aligned}
\text{Feasibility:} \qquad & M\mathbf{1} = \mathbf{1},\ \mathbf{1}^\top M = \mathbf{1}^\top,\ M \geq 0 \\
& \left(\frac{\partial \mathcal{L}}{\partial u} = -M\mathbf{1} + \mathbf{1} = 0 \Rightarrow M\mathbf{1} = \mathbf{1}\right) \\
& \left(\frac{\partial \mathcal{L}}{\partial w} = -M^\top\mathbf{1} + \mathbf{1} = 0 \Rightarrow M^\top\mathbf{1} = \mathbf{1}\right) \\
\text{Stationarity:} \qquad & C = u\mathbf{1}^\top + \mathbf{1}w^\top + U,\ U \geq 0 \\
\text{(elementwise)} \qquad & C_{ij} = u_i + w_j + U_{ij},\ U_{ij} \geq 0,\ \forall i,j \\
& \left(\frac{\partial \mathcal{L}}{\partial M} = C - u\mathbf{1}^\top - \mathbf{1}w^\top - U = 0 \Rightarrow C = u\mathbf{1}^\top + \mathbf{1}w^\top + U\right) \\
\text{Complementary Slackness:} \qquad & 0 = \text{Tr}(U^\top M) = \sum_{ij} U_{ij}M_{ij} \iff 0 = U_{ij}M_{ij},\ \forall i,j
\end{aligned}$$

Proof: To verify if the matching assignment obtained is optimal in the primal problem, let $\Xi \geq 0 \in$

$\mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$ be a permutation matrix that defines a particular agent task assignment

$$\Xi_{ij} = \begin{cases} 1 & ;\text{if agent } i \text{ performs task } j \\ 0 & ;\text{otherwise} \end{cases}$$

If $\Xi \geq 0$ is a feasible assignment then $\mathbf{1}^\top \Xi = \mathbf{1}^\top$ and $\Xi \mathbf{1} = \mathbf{1}$. Taking the stationarity condition, multiply it by Ξ and taking a trace, and then substituting in the feasibility condition gives

$$\begin{aligned} \underbrace{\text{Tr}(C^\top \Xi)}_{\text{overall assignment cost}} &= \text{Tr}(\mathbf{1}u^\top \Xi) + \text{Tr}(w\mathbf{1}^\top \Xi) + \text{Tr}(U^\top \Xi) \text{ (Stationarity condition)} \\ &= \text{Tr}(\Xi \mathbf{1}u^\top) + \text{Tr}(w\mathbf{1}^\top \Xi) + \text{Tr}(U^\top \Xi) \\ &= \text{Tr}(\mathbf{1}u^\top) + \text{Tr}(w\mathbf{1}^\top) + \text{Tr}(U^\top \Xi) \text{ (feasibility conditions)} \\ &= \underbrace{u^\top \mathbf{1} + \mathbf{1}^\top w}_{\text{optimal matching cost}} + \underbrace{\text{Tr}(U^\top \Xi)}_{\geq 0, \text{ inefficiency of matching}} \end{aligned}$$

Note here that the term $u^\top \mathbf{1} + \mathbf{1}^\top w$ is independent of the assignment and also that $\text{Tr}(U^\top \Xi)$ is always positive. It follows then that the overall assignment cost is always greater than or equal to $u^\top \mathbf{1} + \mathbf{1}^\top w$ and thus $u^\top \mathbf{1} + \mathbf{1}^\top w$ represents the optimal assignment cost. The term $\text{Tr}(U^\top \Xi)$ then refers to the inefficiency of the assignment Ξ.

Ξ represents an optimal assignment if the support of Ξ is contained in the support of the optimal assignment M, ie. $\Xi_{ij} > 0$ only if $M_{ij} > 0$. Consequently, $\text{Tr}(U^\top M) = 0 \implies \text{Tr}(U^\top \Xi) = 0$, which proves that Ξ is an optimal and efficient matching assignment. Since the optimization problem is linear for generic costs C, the optimal M will lie at a vertex of the feasible set, ie. M will be a permutation matrix.

4.2.4 Matching: Further Dual Interpretation

The goal of the dual objective function (13a) is to maximize the minimum assignment cost of each pairing agent i and task j represented by the sum of the row and column potential and maximize the lower bound of the overall assignment cost. The constraints (13b) describe that the assignment cost equals the sum of the minimum cost and the inefficiency of each pairing agent i and task j, where the inefficiency is non-negative. An inefficient assignment means that the pairing of agent i and task j is not the most optimal. Therefore, the

solution to the dual problem can be found by maximizing the minimum overall assignment cost as much as possible while ensuring that each matching pair is optimal or suboptimal, i.e., having a positive inefficiency.

5 Matching and Routing with Congestion

After defining the primal and dual problems of routing games with or without congestion and balanced assignment problems, we demonstrate our proposed problem formulations for combining the matching assignment problem and routing games in this section. We will show that our formulation finds an optimal matching of agents to tasks when costs for each agent-task pairing are travel times in a congested network. Our optimization formulation solves for a population traffic flow pattern that ensures 1) that members in the population take only routes with minimal travel times in the presence of congestion, i.e., satisfying the Wardrop equilibria condition, and 2) that assignment of agents to tasks minimizes the overall assignment cost. Since our formulation is now convex instead of linear as in the original matching problem, our assignment will take the form of a probabilistic matching where sometimes a single task is split between multiple agents and/or a single agent performs multiple tasks. In this section, we derive both the primal and dual problems with explanations of variables and problem setup meaning and then prove our problem formulation gives an optimal matching for costs determined by a Wardrop equilibrium routing solution.

5.1 Matching and Routing: Primal Formulation

We first present the primal form of our optimization problem.

Primal:

$$\min_{x_{ij}, M_{ij}} \quad \sum_e \int_0^{x_e} \ell_e(s)\, ds \tag{14a}$$

$$\text{s.t.} \quad x_e = \Sigma_{ij} x_{ije},\ \forall e \tag{14b}$$

$$E_{ij} x_{ij} = S_{ij} M_{ij},\ x_{ij} \geq 0,\ \ \forall i, j \tag{14c}$$

$$M\mathbf{1} = \mathbf{1},\ M^\top \mathbf{1} = \mathbf{1},\ M \geq 0 \tag{14d}$$

for edge e and matching between agent i and task j, where

- $x_{ij} \in \mathbb{R}^{|\mathcal{E}|}$: an edge flow vector that gives the portion of traffic flow on each edge corresponding to

agent i performing task j.

- $\ell(x) \in \mathbb{R}^{|\mathcal{E}|}$: a non-decreasing and continuous latency cost function for each edge as a function of x.
- $E_{ij} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$: the incidence matrix of the graph associated with agent i and task j
- $S_{ij} \in \mathbb{R}^{|\mathcal{V}|}$: a source-sink vector indicating where the flow enters and exits the network for agent i performing task j.
- $M \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$: a doubly stochastic matrix showing the probability, M_{ij}, of agent i performing task j.

Constraint (14b) means the flow in edge e is the sum of the flow of all agents traveling edge e. Constraints (14c) and (14d) account for the conservation of flow in the route taken by each matching pair, agent i and task j, and M matrix being doubly stochastic. Besides, the conservation of flow considers the matching assignment by indicating the source and the sink being associated with the locations of agent i and task j. In addition, both the edge flow vector and assignment matrix have non-negative elements.

The primal problem is a minimization problem with the objective function (14a) being the routing game potential function (1). Note that even though we are combining the routing and matching problem, only the routing game objective shows up in the objective of the joint problem. We will see from the KKT conditions below that this formulation will cause agents to choose routes in a similar way to the Wardrop equilibrium case, and the dual variables at equilibrium will transmit cost information about the shortest path to the matching portion of the optimization problem.

5.2 Matching and Routing: Dual Derivation

The dual objective function in this formulation allows us to find the lower bound on the primal minimization problem, and we derive the dual problem by minimizing the Lagrangian function of the primal problem as follows. To simplify some of the analysis, rather than including the constraints $x_e = \sum_{ij} x_{ije}$ in the Lagrangian, we substitute them into the objective first and then get the Lagrangian.

$$\min_{x_{ij},M_{ij}} \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \mathcal{L} = \min_{x_{ij},M_{ij}} \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \left(\sum_e \int_0^{\sum_{ij} x_{ije}} \ell_e(s)ds \right) - \left(\sum_{ij} v_{ij}^\top (E_{ij}x_{ij} - S_{ij}M_{ij}) - \mu_{ij}^\top x_{ij} \right)$$
$$- u^\top (M\mathbf{1} - \mathbf{1}) - w^\top (M^\top \mathbf{1} - \mathbf{1}) - \text{Tr}(U^\top M)$$

Grouping terms, we get

$$\Rightarrow \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \min_{x_{ij}} \underbrace{\sum_e \left[\int_0^{\sum_{ij} x_{ije}} \ell_e(s)\, ds - \sum_{ij} \left(\left[v_{ij}^\top E_{ij} \right]_e + \mu_{ije} \right) x_{ije} \right]}_{\mathcal{L}_1}$$
$$+ \min_{M_{ij}} \underbrace{\sum_{ij} \left(v_{ij}^\top S_{ij} M_{ij} - u_i M_{ij} - w_j M_{ij} - U_{ij} M_{ij} \right)}_{\mathcal{L}_2} + u^\top \mathbf{1} + w^\top \mathbf{1}$$

First, we solve for the first inner minimization problem explicitly for x_{ije} by taking the derivative with respect to x_{ije} and setting it equal to 0 for each x_{ije}. Note that

$$f(x) = \sum_e \int_0^{\sum_{ij} x_{ije}} \ell_e(s)\, ds \;\Rightarrow\; \tfrac{\partial f}{\partial x_{ije}} = \ell_e \left(\sum_{ij} x_{ije} \right) \left[\tfrac{\partial}{\partial x_{ije}} \left(\sum_{ij} x_{ije} \right) \right]$$
$$= \ell_e \left(\sum_{ij} x_{ije} \right) \cdot (1) = \ell_e \left(\sum_{ij} x_{ije} \right) = \ell_e(x_e).$$

Differentiating the $\mathcal{L}_1$ term in the Lagrangian then gives

$$\frac{\partial \mathcal{L}_1}{\partial x_{ije}} = \ell_e(x_e) - \left[v_{ij}^\top E_{ij} \right]_e - \mu_{ije}^\top) = 0 \;\Rightarrow\; \ell_e(x_e) = \left[v_{ij}^\top E_{ij} \right]_e + \mu_{ije}$$

Defining a new dual variable c_e equal to the quantity $\ell_e(x_e)$, i.e., $c_e = \ell_e(x_e) = \left[v_{ij}^\top E_{ij} \right]_e + \mu_{ije}$, we get

$$\begin{aligned}
\min_{x_{ij}} \mathcal{L}_1 &= \min_{x_{ij}} \sum_e \left[\int_0^{x_e} \ell_e(s)\, ds - \sum_{ij} \ell_e(x_e) x_{ije} \right] \\
&= \min_{x_{ij}} \sum_e \left[\int_0^{x_e} \ell_e(s)\, ds - \ell_e(x_e) \left(\sum_{ij} x_{ije} \right) \right] \\
&= \min_{x_{ij}} \sum_e \left[\int_0^{x_e} \ell_e(s)\, ds - c_e x_e \right] \\
&= -\sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds
\end{aligned}$$

where in the last step we have applied Proposition 1 (the Legendre transform).

Second, we solve the second inner minimization problem explicitly for M_{ij} by taking the derivative with

respect to M_{ij} and setting it equal to 0 as well.

$$\frac{\partial \mathcal{L}_2}{\partial M_{ij}} = v_{ij}^\top S_{ij} - u_i - w_j - U_{ij} = 0 \;\Rightarrow\; v_{ij}^\top S_{ij} = u_i + w_j + U_{ij}$$

It follows then that

$$\begin{aligned}
\min_{M_{ij}} \mathcal{L}_\in &= \min_{M_{ij}} \sum_{ij} \left(v_{ij}^\top S_{ij} M_{ij} - u_i M_{ij} - w_j M_{ij} - U_{ij} M_{ij} \right) \\
&= \min_{M_{ij}} \sum_{ij} \left(v_{ij}^\top S_{ij} - u_i - w_j - U_{ij} \right) M_{ij} \\
&= 0
\end{aligned}$$

The results from the two inner minimization problems give us the dual objective function (14a) and constraints (14). Plugging back in we get

$$\begin{aligned}
\max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \min_{x_{ij},M_{ij}} \mathcal{L} = \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \min_{x_{ij},M_{ij}} & \underbrace{\sum_e \left[\int_0^{\sum_{ij} x_{ije}} \ell_e(s)\, ds - \sum_{ij} \left(\left[v_{ij} E_{ij} \right]_e + \mu_{ije} \right) x_{ije} \right]}_{(1)} \\
& + \underbrace{\sum_{ij} \left(v_{ij}^\top S_{ij} M_{ij} - u_i M_{ij} - w_j M_{ij} - U_{ij} M_{ij} \right)}_{(2)} + u^\top \mathbf{1} + w^\top \mathbf{1} \\
= \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} & \left(- \sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \right) + u^\top \mathbf{1} + w^\top \mathbf{1} \\
\Rightarrow \max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} & \left(\sum_{ij} u_i + w_j \right) - \left(\sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \right)
\end{aligned}$$

Finally, the dual problem is given by

Dual:

$$\max_{\substack{c,w_{ij},v_{ij},\\ u_{ij},\mu_{ij},U_{ij}}} \quad \left(\sum_{ij} u_i + w_j \right) - \left(\sum_e \int_{\ell_e(0)}^{c_e} \ell_e^{-1}(s)\, ds \right) \tag{15a}$$

$$\text{s.t.} \quad c^\top = v_{ij}^\top E_{ij} + \mu_{ij}^\top,\ \mu_{ij}^\top \geq 0,\ \forall\, i, j \tag{15b}$$

$$v_{ij}^\top S_{ij} = u_i + w_j + U_{ij},\ U_{ij} \geq 0,\ \forall\, i, j \tag{15c}$$

5.2.1 Matching and Routing: KKT Conditions

The rest of the subsection derives the KKT conditions to prove the solution obtained is optimal for the matching and routing problem formulation with congestion.

KKT Conditions:

$$
\begin{aligned}
\text{Feasibility:} \quad E_{ij}x_{ij} &= S_{ij}M_{ij},\ x_{ij} \geq 0,\ \forall i,j \\
M\mathbf{1} &= \mathbf{1},\ M^\top \mathbf{1} = \mathbf{1},\ M \geq 0 \\
x_e &= \Sigma_{ij} x_{ije},\ \forall e \\
&\left(\frac{\partial \mathcal{L}}{\partial v} = -E_{ij}x_{ij} + S_{ij}M_{ij} = 0 \Rightarrow E_{ij}x_{ij} = S_{ij}M_{ij}\right) \\
&\left(\frac{\partial \mathcal{L}}{\partial u} = -M\mathbf{1} + \mathbf{1} = 0 \Rightarrow M\mathbf{1} = \mathbf{1}\right) \\
&\left(\frac{\partial \mathcal{L}}{\partial w} = -M^\top \mathbf{1} + \mathbf{1} = 0 \Rightarrow M^\top \mathbf{1} = \mathbf{1}\right)
\end{aligned}
$$

$$
\begin{aligned}
\text{Stationarity:} \quad \ell(x)^\top &= c^\top \\
c^\top &= v_{ij}^\top E_{ij} + \mu_{ij}^\top,\ \mu_{ij}^\top \geq 0,\ \forall i,j \\
C_{ij} &= v_{ij}^\top S_{ij},\ \forall i,j \\
C &= u\mathbf{1}^\top + \mathbf{1}w^\top + U,\ U \geq 0 \\
\text{(elementwise)} \quad v_{ij}^\top S_{ij} &= u_i + w_j + U_{ij},\ U_{ij} \geq 0,\ \forall i,j \\
&\left(\frac{\partial \mathcal{L}}{\partial x_{ije}} = \ell_e(x_e) - v_{ije}E_{ije} - \mu_{ije} = 0\ \Rightarrow \ell(x)^\top = v_{ij}^\top E_{ij} + \mu_{ij}^\top\right) \\
&\left(\frac{\partial \mathcal{L}}{\partial M_{ij}} = v_{ij}^\top S_{ij} - u_i - w_j - U_{ij} = 0 \Rightarrow v_{ij}^\top S_{ij} = u_i + w_j + U_{ij}\right)
\end{aligned}
$$

$$
\begin{aligned}
\text{Complementary Slackness:} \quad 0 = \mu^\top x \quad &\iff \quad 0 = \mu_e x_e,\ \forall e \\
0 = \text{Tr}(U^\top M) = \sum_{ij} U_{ij}M_{ij} \ &\iff \ 0 = U_{ij}M_{ij},\ \forall i,j
\end{aligned}
$$

Proof: We now show that our optimization problem finds an optimal matching (in the relaxed matching space) where the costs for each agent-task pair are determined by the shortest path in the presence of con-

gestion. We will show that at optimum, agents only take optimal routes from the origins to destinations and that the overall agent-task assignment is done to minimize the overall matching cost.

Let $\Xi \geq 0 \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$ be a permutation matrix that represents a specific agent-task assignment and let $\xi_{ij} \in \mathbb{R}^{|\mathcal{E}|}$ for each i, j be indicator vectors for the routes chosen by each agent in order to complete their task. If $\Xi \geq 0$ is a feasible assignment, then $\mathbf{1}^\top \Xi = \mathbf{1}^\top$, and $\Xi \mathbf{1} = \mathbf{1}$. If $\xi_{ij} \geq 0$ is a feasible route for task pairing i, j, then $E_{ij}\xi_{ij} = S_{ij}$. As in the simple routing game case, right multiplying the routing stationarity condition by ξ_{ij} for a feasible route, and substitute in the feasibility condition gives

$$\underbrace{\sum_{e\in r} \ell_e(x_e)}_{\text{total travel cost}} = \ell(x)^\top \xi_{ij} \text{ (objective function) } = v_{ij}^\top E_{ij}\xi_{ij} + \mu_{ij}^\top \xi_{ij} \text{ (Stationarity condition)} \tag{16a}$$

$$= \underbrace{v_{ij}^\top S_{ij}}_{\text{optimal cost from source to sink}} + \underbrace{\mu_{ij}^\top \xi_{ij}}_{\text{route inefficiency}} \text{ (feasibility condition)} \tag{16b}$$

As in the original routing problem for each agent-task pairing i, j, $\mu_{ij}^\top \xi_{ij}$ is positive and $v_{ij}^\top S_{ij}$ is independent of the route chosen. Thus, $v_{ij}^\top S_{ij}$ represents the optimal travel time for that pairing, and $\mu_{ij}^\top \xi_{ij}$ represents the inefficiency of the route.

Multiplying the matching stationarity conditions by Ξ, taking the trace, and substituting in the feasibility condition, we get

$$\text{Tr}(C^\top \Xi) = \text{Tr}(\mathbf{1}u^\top \Xi + w\mathbf{1}^\top \Xi + U^\top \Xi) \tag{17a}$$

$$= u^\top \mathbf{1} + \mathbf{1}^\top w + \text{Tr}(U^\top \Xi) \tag{17b}$$

$$\Rightarrow \sum_{ij} \underbrace{C_{ij}\Xi_{ij}}_{\text{matching cost}} = \underbrace{u^\top \mathbf{1} + \mathbf{1}^\top w}_{\text{optimal matching cost}} + \underbrace{\text{Tr}(U^\top \Xi)}_{\substack{\geq 0, \text{ inefficiency of} \\ \text{of assignment } i \text{ to } j}} \tag{17c}$$

The key connection between the routing and matching portions of the problem is given by the stationarity condition $C_{ij} = v_{ij}^\top S_{ij}$ for each i, j. This condition ensures that the costs for the matching problem are indeed the shortest path costs in the congested network. Combining this with equation (16b) we get

$$C_{ij} = \ell(x)^\top \xi_{ij} - \mu_{ij}^\top \xi_{ij}$$

and then finally substituting into (17c) we get

$$\sum_{ij} \ell(x)^\top \xi_{ij} = \underbrace{u^\top \mathbf{1} + \mathbf{1}^\top w}_{\text{optimal matching cost}} + \underbrace{\text{Tr}(U^\top \Xi)}_{\substack{\geq 0, \text{ inefficiency of} \\ \text{of assignment } i \text{ to } j}} + \sum_{ij} \underbrace{\mu_{ij}^\top \xi_{ij}}_{\geq 0, \text{ inefficiency of route } r}$$

As in the original matching problem, $u^\top \mathbf{1} + \mathbf{1}^\top w$ is independent of the matching, $\text{Tr}(U^\top \Xi)$ and $\sum_{ij} \mu_{ij}^\top \xi_{ij}$ are positive, and thus $u^\top \mathbf{1} + \mathbf{1}^\top w$ represents the optimal matching cost. $\text{Tr}(U^\top \Xi)$ is any inefficiency due to a suboptimal matching; $\sum_{ij} \mu_{ij}^\top \xi_{ij}$ is any inefficiency due to sub-optimal routing.

If ξ_{ij} represents an optimal route for agent i with task j, i.e., the support of ξ_{ij} is contained in the support of the optimal flow x_{ij}, then $\xi_{ij} > 0$ only if $x_{ij} > 0$. Consequently, $\mu_{ij}^\top x_{ij} = 0 \implies \mu_{ij}^\top \xi_{ij} = 0$, which proves that ξ_{ij} is an optimal and efficient route for agent i and task j. If Ξ represents an optimal assignment, i.e., the support of Ξ is contained in the support of the optimal assignment M, then $\Xi > 0$ only if $M > 0$. Consequently, $\text{Tr}(U^\top M) = 0 \implies \text{Tr}(U^\top \Xi) = 0$, which proves that Ξ is an optimal and efficient matching assignment.

Moreover, this proves that the optimal route, ξ_{ij}, for each optimal assignment pair agent i and task j achieves Wardrop equilibrium based on the Definition 2 and Equation 5, where for paths $P, P' \in \mathcal{P}^i$ with mass from agent i and indicator vectors ξ_{ij}, ξ'_{ij} respectively,

$$\ell_P(x) = \ell(x)^\top \xi_{ij} \leq \ell(x)^\top \xi'_{ij} = \ell_{P'}(x)$$

5.3 Matching and Routing: Further Dual Interpretation

The dual objective function (15a) represents the summed minimum assignment cost of each pairing agent i and task j computed by the sum of the row and column potential minus the sum of penalty terms for increasing the edge costs c_e above the uncongested costs $\ell_e(0)$. As in the matching problem, our goal is to increase the row and column potential variables as much as possible in order to find a better lower bound for the optimal matching. Due to constraint (15c), we can increasing u and w by initially by reducing U but eventually we will have to increase the term $v_{ij}^\top S_{ij}$ if we want to increase u and w further. As in the routing game problem due to constraint (15b), increasing $v_{ij}^\top S_{ij}$ can be done initially be reducing μ_{ij} but eventually we will have to increase the values in c incurring a penalty defined by the integral terms in the objective. As

in the routing game formulation, we want to increase this value $v_{ij}^{\top} S_{ij}$ as much as possible while receiving as little penalty as possible for increasing c_e above $\ell_e(0)$. Note that the edge cost vector c must be an upper bound on $v_{ij}^{\top} E_{ij}$ for each agent-task assignment i, j. Thus, increasing the $v_{ij}^{\top} S_{ij}$ term for any task assignment pair could force us to push up an element of c and pay the penalty term $\int_{\ell_e(0)}^{c_e} {\ell_e}^{-1}(u)\, ds$. The dual problem then seeks to maximize the minimum assignment cost between the set of agents and tasks while not paying too much penalty for increasing any given edge cost too much.

6 Algorithms

This section introduces Dijkstra's algorithm [7] and A^* algorithm [8] for solving shortest path problems, the Hungarian algorithm [9] for solving assignment problems, and Frank-Wolfe algorithm [6] for solving constrained convex optimization problems as these algorithms are implemented in our proposed algorithm. Next, the Section presents the structure of the proposed algorithm for finding the optimal assignment and the corresponding optimal route between agent i and task j while considering congestion simultaneously.

6.1 Dijkstra's Algorithm

Dijkstra's algorithm aims to find the shortest path between nodes in a weighted graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with nodes $\mathcal{V}$, weighted edges $\mathcal{E}$, and weight function $\mathcal{W}$. Dijkstra's algorithm first separates all the nodes into two sets, V_E and V_U, where V_E contains a set of nodes with the determined shortest path to v_s, while V_U contains a set of nodes with undetermined shortest paths to v_s. With V_E being initialized to have only v_s, the algorithm finds the shortest path between v_s and a node in V_U and then moves the node from V_U to V_E sorted by the corresponding shortest path length until there are no nodes in V_U. The shortest path length is the sum of the corresponding weights in the shortest path from v_s and any node in V_U. Consequently, we obtain the shortest paths from v_s to any nodes v_i as well as the shortest path from v_s to the destination node, v_d. The flowchart of Dijkstra's algorithm is denoted in Algorithm 1.

6.2 A* Algorithm

The A^* algorithm is based on the approach of Dijkstra's method but eliminates unnecessary searches by adding a heuristic function to accelerate the rate of finding the shortest path from the source node, v_s, to the destination node, v_d. Consider a weighted graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with nodes $\mathcal{V}$, weighted edges $\mathcal{E}$, and weight function $\mathcal{W}$. V_O contains the set of discovered nodes that may need to be expanded, while P_s records the shortest path from v_s to v_d. In addition, the evaluation function is defined as $f(v_i) = g(v_i) + h(v_i)$, where $g(v_i)$ is the cost of the path from v_s to v_i while $h(v_i)$ is selected to be the Euclidean distance from v_i to v_d. The algorithm chooses the lowest cost nodes to extend the path starting from v_s until reaching v_d as demonstrated in Algorithm 2.

Algorithm 1 Dijkstra's Algorithm

Input: An edge-weighted graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with nodes $\mathcal{V}$, weighted edges $\mathcal{E}$, weight function $\mathcal{W}$. Let the source node be v_s and the destination node be v_d.
Output: Shortest path and the length, $d(v_d)$, from the source node v_s to the destination node v_d in $\mathcal{G}$.
Initialization: $V_E = \{v_s\},\ V_U = V - V_E,\ d(v_s) = 0$.

```
for v_i in V do
    if v_i is a neighbor of v_s then
        d(v_i) = min(W(v_s, v_i))
    else
        d(v_i) = ∞
    end
end
while V_U is not empty do
    w_i = arg min_{w_i ∉ V_E} {d(w_i)}
    V_U = V_U / {w_i}
    V_E = V_E ∪ {w_i}
    for v_i ∉ V_E do
        if v_i is a neighbor of v_s then
            d(v_i) = min(d(v_i), d(w_i) + W(w_i, v_i))
        end
    end
end
return V_E, d(v_d)
```

6.3 Hungarian (Kuhn-Munkres) Algorithm

Here, we give a description of the classical Hungarian or Kuhn-Munkres algorithm for matching in a weighted bipartite graph. The input is the assignment cost matrix, C, between the agents and tasks, while the output is the optimal matching assignment matrix, M, where both C and M are square matrices with non-negative values. In Algorithm 3, we follow the algorithmic form given in [36].

6.4 Frank-Wolfe Algorithm

Suppose the objective function f is convex and continuously differentiable, and the domain $\mathcal{D}$ is a compact convex set in a vector space. Frank-Wolfe algorithm described in Algorithm 4 solves such optimization problems by considering the linearization of the objective function and moving towards a minimizer of this linear function.

Algorithm 2 A^* Algorithm

Input: An edge-weighted graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with nodes $\mathcal{V}$, weighted edges $\mathcal{E}$, weight function $\mathcal{W}$.
Let the source node be v_s and the destination node be v_d.
Output: Shortest path, P_s and the length, $f(v_d)$, from the source node v_s to the destination node v_d in $\mathcal{G}$.
Initialization: $V_O = \{v_s\}, P_s = \{\},\ g(v_i) = \infty,\ f(v_s) = g(v_s) + h(v_s) = 0 + h(v_s) = h(v_s)$.

```
while V_O is not empty do
    v_i = arg min_{v_i ∈ V_O} {f(v_i)}
    if v_i = v_d then
        P_s = P_s ∪ {v_i}
        return P_s
    end
    V_O = V_O / {v_i}
    for w_i such that (v_i, w_i) ∈ E do
        g(w_i)_tentative = g(v_i) + W(w_i, v_i)
        if g(w_i)_tentative < g(w_i) then
            P_s = P_s ∪ {v_i}
            g(w_i) = g(w_i)_tentative
            f(w_i) = g(w_i)_tentative + h(w_i)
            if w_i ∉ V_O then
                V_O = V_O ∪ {w_i}
            end
        end
    end
end
return Path is not found
```

Algorithm 3 Hungarian (Kuhn-Munkres) Algorithm

Input: Cost Matrix: $C \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{T}|}, C \geq 0$
Output: Assignment Matrix: $M \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{T}|}, M \geq 0$
Compute $C_{ij} - u_i$ for each row i, where u_i is the row minima
Compute $C_{ij} - w_j$ for each column j, where w_j is the column minima
Let C' be the resulting matrix, where $C' = C_{ij} - u_i - w_j$
Let n be the minimum number of horizontal and vertical lines needed to cover all zeros in C'

```
while M is not found do
    if n = |A| then
        return M is found
    else
        Let k be the smallest element in C'
        Subtract k from all uncovered elements: C' = C'_uncovered − k
        Add k to all covered elements: C' = C'_covered + k
        Update n
    end
end
```

Algorithm 4 Frank-Wolfe Algorithm

Input: $x^{(0)} \in \mathcal{X}$
Output: $x^{(t)} \in \mathcal{X}$
for $t = 0...T$ **do**
  Compute $s^{(t)} = \arg\min_{s \in \mathcal{X}} \langle s, \nabla f(x^{(t)}) \rangle$
  Update $x^{(t+1)} = (1-\gamma)x^{(t)} + \gamma s^{(t)}$ for $\gamma = \frac{2}{2+t}$
end

6.5 Frank-Wolfe for Matching and Routing with Congestion

Our proposed algorithm follows the Frank-Wolfe algorithm framework where we linearize the objective function of our proposed congestion-aware routing game and assignment problem formulation In order to facilitate solving the linearized subproblem, we decompose it into two subproblems (19) and (20), where subproblem (19) is for solving routing game with congestion and subproblem (20) is for solving balanced assignment problems. This decomposition allows us to use Dijkstra or A* to solve the optimal routing portion of the problem (19) and the Hungarian algorithm to solve the optimal matching portion (20). This leads to our proposed algorithm shown in Algorithm 5.

6.5.1 Linearized Routing with Congestion and Matching

To apply Frank-Wolfe to our proposed problem, we linearize (14) to get the following optimization problem.

$$\xi_{ij}^{(k)}, \mathbf{M}_{ij}^{(k)} = \arg\min_{x'_{ij}, M'} \quad \sum_{ij} \sum_{e} \ell_e(x_e^{(k)}) x'^{(k)}_{ije} \tag{18a}$$

$$\text{s.t.} \quad E_{ij} x'^{(k)}_{ij} = S_{ij} M'^{(k)}_{ij},\ x'^{(k)}_{ij} \geq 0,\ \forall i, j \tag{18b}$$

$$M'^{(k)} \mathbf{1} = \mathbf{1},\ M'^{(k)\top} \mathbf{1} = \mathbf{1},\ M'^{(k)} \geq 0 \tag{18c}$$

Many methods could be used to solve this linear program; however, with a small amount of analysis, we can show how this joint LP can be solved using a combination of Dijkstra's algorithm and the Hungarian algorithm. Note that the cost (18) does not depend explicitly on M', just x'. Given the nature of the constraints in (18b), scaling ${M_{ij}}'^{(k)}$ only changes the magnitude of the optimal ${x_{ij}}'^{(k)}$, not its direction. As

a result, we can solve for the optimal ${x_{ij}}'^{(k)}$ separate from ${M_{ij}}'^{(k)}$. For each i, j, let ${\eta_{ij}}^{(k)}$ solve

$$\eta_{ij}^{(k)} = \arg\min_{\eta'_{ij}} \quad \sum_e \ell_e(x_e^{(k)}){\eta'}_{ije} \tag{19a}$$

$$\text{s.t.} \quad E_{ij}{\eta'}_{ij} = S_{ij},\ {\eta'}_{ij} \geq 0, \tag{19b}$$

for each i, j pair. Note that if $\eta_{ij}^{(k)}$ satisfies (19b), then setting $\xi_{ij}^{(k)} = \eta_{ij}^{(k)}\mathbf{M}_{ij}^{(k)}$ will satisfy (18b). We can then write (18) as

$$\min_{M'} \quad \sum_{ij} \mathbf{C}_{ij}{M'}_{ij}^{(k)} \tag{20a}$$

$$\text{s.t.} \quad M'^{(k)}\mathbf{1} = \mathbf{1},\ M'^{(k)\top}\mathbf{1} = \mathbf{1},\ M'^{(k)} \geq 0 \tag{20b}$$

with $\mathbf{C}_{ij} = \sum_e \ell_e(x_e^{(k)}){\eta^{(k)}}_{ije}$, where each ${\eta^{(k)}}_{ij}$ solves (19). The solutions to (18) are then given by solving (19) for each $\eta_{ij}^{(k)}$, then solving (20) for each $\mathbf{M}_{ij}^{(k)}$, and finally by computing $\xi_{ij}^{(k)} = \eta_{ij}^{(k)}\mathbf{M}_{ij}^{(k)}$. Since (19) is a shortest path problem, it can be solved efficiently using Dijkstra or A^*, and since (20) is a matching problem, it can be solved efficiently using the Hungarian algorithm. We formalize this algorithm in the following section.

Algorithm 5 Frank-Wolfe for Matching and Routing with Congestion

Input: $x^{(0)} \in \mathcal{X}$
Output: $x^{(k)} \in \mathcal{X}$
for $k = 0...T$ **do**
  Compute edge costs: $e_c = Ax^{(k)} + b$ for edge e
  Find the shortest path, P_{ij}, and length, C_{ij}, between agent i and task j with Dijkstra's or A^* algorithm
  Let $\eta_{ij}^{(k)}$ denote the optimal path indicator vector
  Construct the assignment cost matrix C
  Solve for the optimal assignment matrix $\mathbf{M}^{(k)}$ with the Hungarian algorithm
  Compute edge flow update vectors $\xi_{ij}^{(k)} = \eta_{ij}^{(k)}\mathbf{M}_{ij}^{(k)}$
  Compute the total edge flow update vector $\xi^{(k)} = \sum_{ij}\xi_{ij}^{(k)}$
  Update $x^{(k+1)} = (1-\gamma)x^{(k)} + \gamma\xi^{(k)}$ for $\gamma = \frac{1}{1+k}$
end

At each iteration k, Algorithm 5 first computes the current edge costs with congestion based on a defined latency function and then calculates the shortest path length with the updated edge costs between agent i and task j by using either Dijkstra's or A^* algorithm to populate the cost matrix, C, for the assignment problem. After that, it solves for the optimal matching between agent i and task j with the Hungarian algorithm and finds each agent's associated edge flow vector, ${\xi^{(k)}}_{ij}$, with $\mathbf{M}_{ij}^{(k)}$ amount of mass added to edges in the shortest paths. Finally, the algorithm performs the Frank-Wolfe update to obtain the approximated overall edge flow vector, $x^{(k)}$, and repeats the whole process until x converges.

Comment on non-compactness of (19)**:** In the routing game literature, Frank-Wolfe is often used to solve for Wardrop equilibria, and Dijkstra/A* are used to solve the linearized shortest path problem as in Problem (19). Technically, Problem (19) may not be compact since we can add any element in the nullspace of E_{ij} (i.e., any cyclic flow) to any feasible ${\eta'}_{ij}^{(k)}$ and remain feasible. For positive latencies, however, this is not a problem since we know that any cyclic flow will increase the value of the objective, and thus the optimal set for (19) contains no cyclic flows. Since we know that the optimizer will lie in a compact set (${\eta'}_{ij}^{(k)}$ with no cyclic flows), we can still apply Frank-Wolfe. For further details, we refer the reader to discussions in [34].

7 Simulation Results

This section presents simulation results of our proposed Frank-Wolfe-based algorithm for solving congestion-aware routing and matching problems in the context of the street network of Seattle City and the multi-robot warehouse problem. We solve for balanced and unbalanced assignments while varying the number of agents and tasks and latency function with convergence plotted.

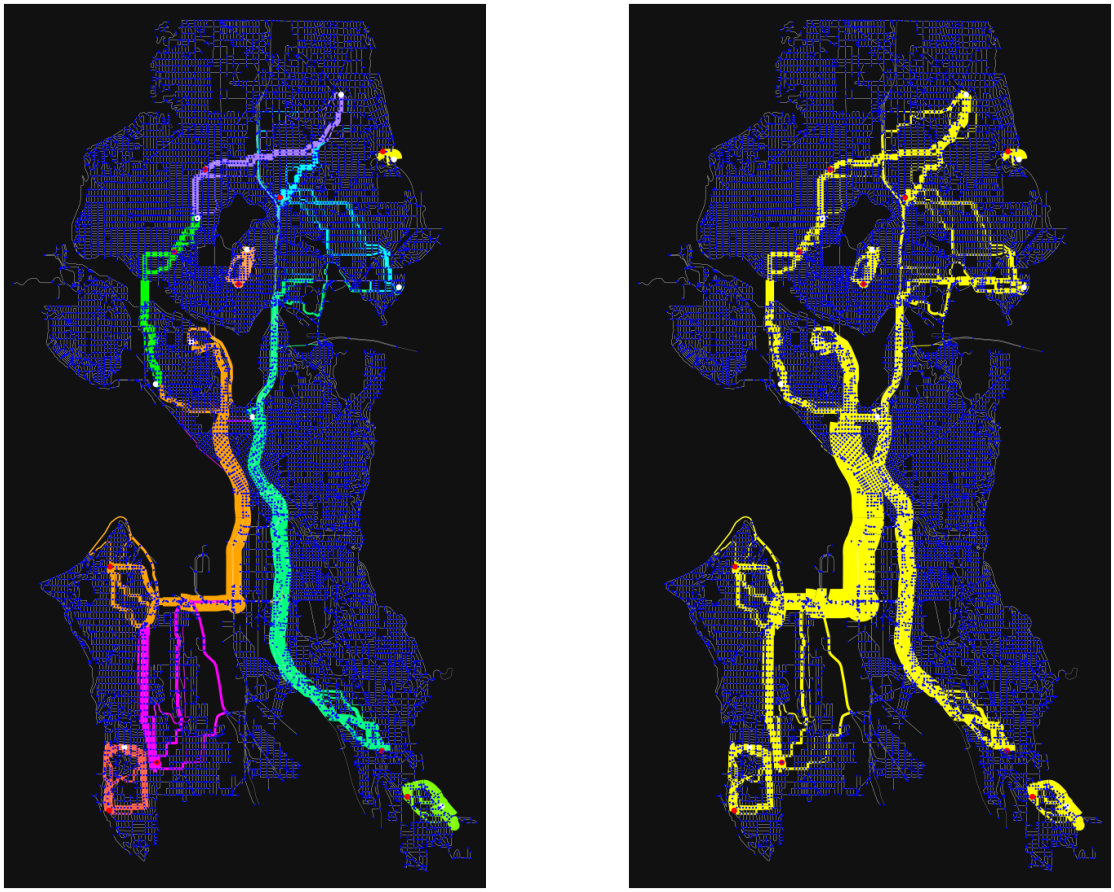
7.1 Street Network Problem

We built our simulation in Python using NetworkX [37], a package for implementing complex networks. Then, we simulated 10 agents and 10 tasks with a latency function of

$$\ell(x) = Ax + b \tag{21}$$

where $A \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is a diagonal matrix, $b \in \mathbb{R}^{|\mathcal{E}|}$ is a constant vector, and $x \in \mathbb{R}^{|\mathcal{E}|}$ is the overall edge flow vector. The traffic congestion is encoded in the diagonal elements of A while the constant edge costs are indicated in b. Based on Algorithm 5 at each iteration, we compute the edge costs according to the latency function (21) and find the shortest path between agent i and task j with Dijkstra's algorithm [7] to construct the assignment cost matrix, $C \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$. We used Dijkstra's algorithm because of the more efficient implementation than A^* algorithm from NetworkX. Next, the optimal assignment matrix, $M \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$, is found with the Hungarian algorithm through the linear sum assignment function from SciPy [38]. after summing up each agent's edge flow vector, the Frank-Wolfe update is performed with a step size of $\frac{1}{k+1}$.

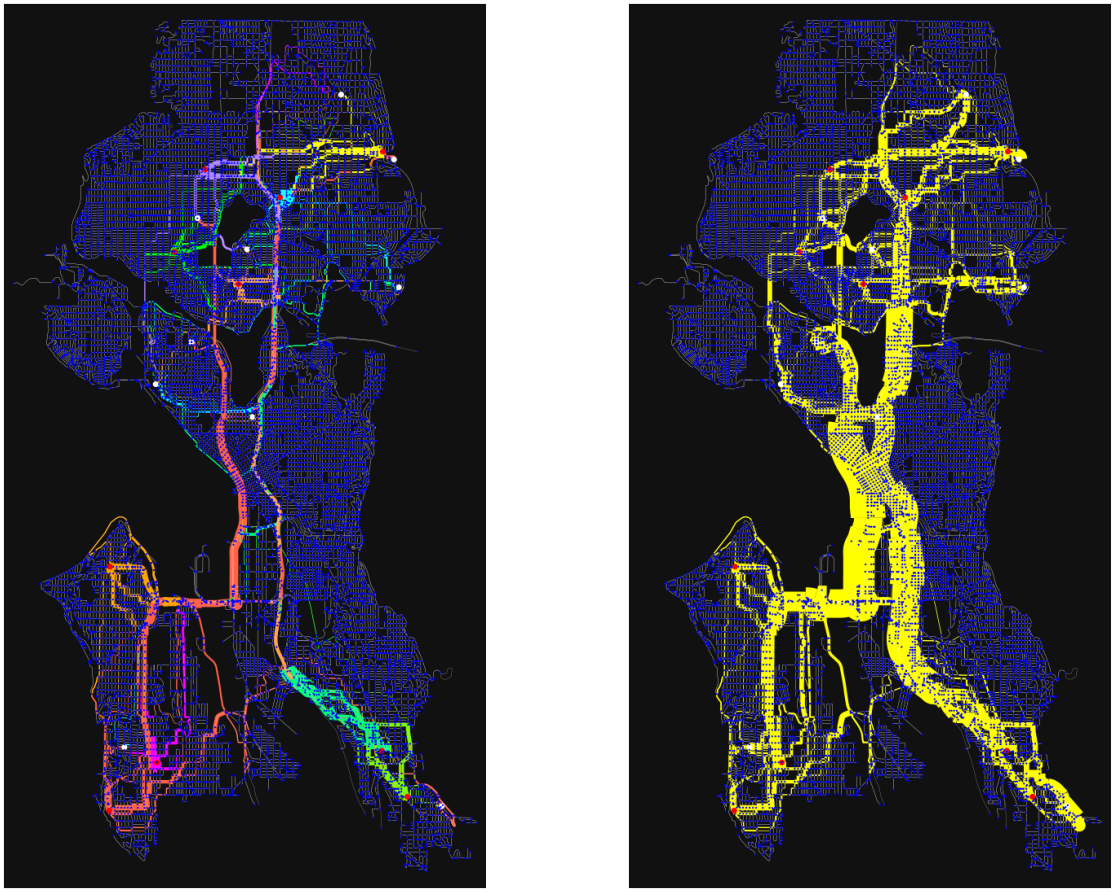
Figure 3 shows the optimal matching and routing result for 10 agents and 10 tasks simulated with A being an identity matrix and b being a vector of ones. The locations of the agents in red and the locations of the tasks in white are randomly picked. Particularly, Figure 3a plots each agent's edge flow by highlighting each optimal assignment and the corresponding shortest paths in different colors, with the widths of the edges reflecting the amount of flow going through each edge. In addition, Figure 3a demonstrates that there may be multiple shortest paths for each agent to travel to its assigned destination, with each path having a different amount of traffic flow distributed at equilibrium to avoid congestion in traffic and achieve the minimum total travel cost. In Figure 3b, we can see the optimal overall edge flow for the network at equilibrium, where the width of each edge includes all the agent's edge flow.



(a) Individual Edge Flow (b) Overall Edge Flow

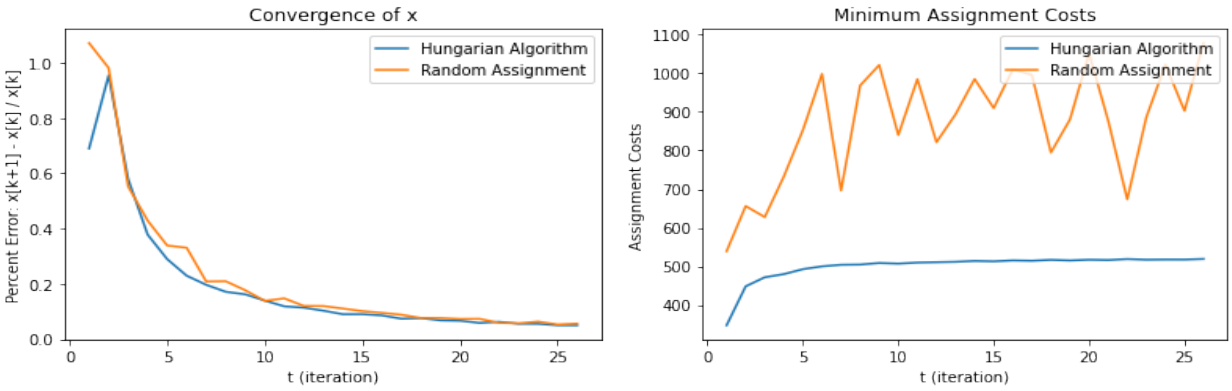
Figure 3: Optimal Matching and Routing Result with Balanced Assignments

Figure 4 shows the matching and routing result simulated by choosing assignments between the agents and the tasks randomly instead of using the Hungarian algorithm. On top of that, we provide Figure 5 to compare the convergence rate and minimum costs achieved by our algorithm and random assignment. In Figure 5, our algorithm converges at iteration 27, while the random assignment algorithm converges at iteration 29. Therefore, our algorithm not only converges at a faster rate but also reaches a lower minimum cost than the random assignment algorithm. Apart from that, we can see Figure 4a reflects that random assignment allows agents to explore non-optimal matching and routing, and Figure 4b further demonstrates the overall edge flow is more spread out in the network, which results in a slower convergence rate and higher total travel cost. As a result, we demonstrate that our algorithm achieves a minimum total travel cost and gives the optimal matching and routing for the given set of agents and tasks effectively and efficiently.



(a) Individual Edge Flow (b) Overall Edge Flow

Figure 4: Random Assignment Matching and Routing Result



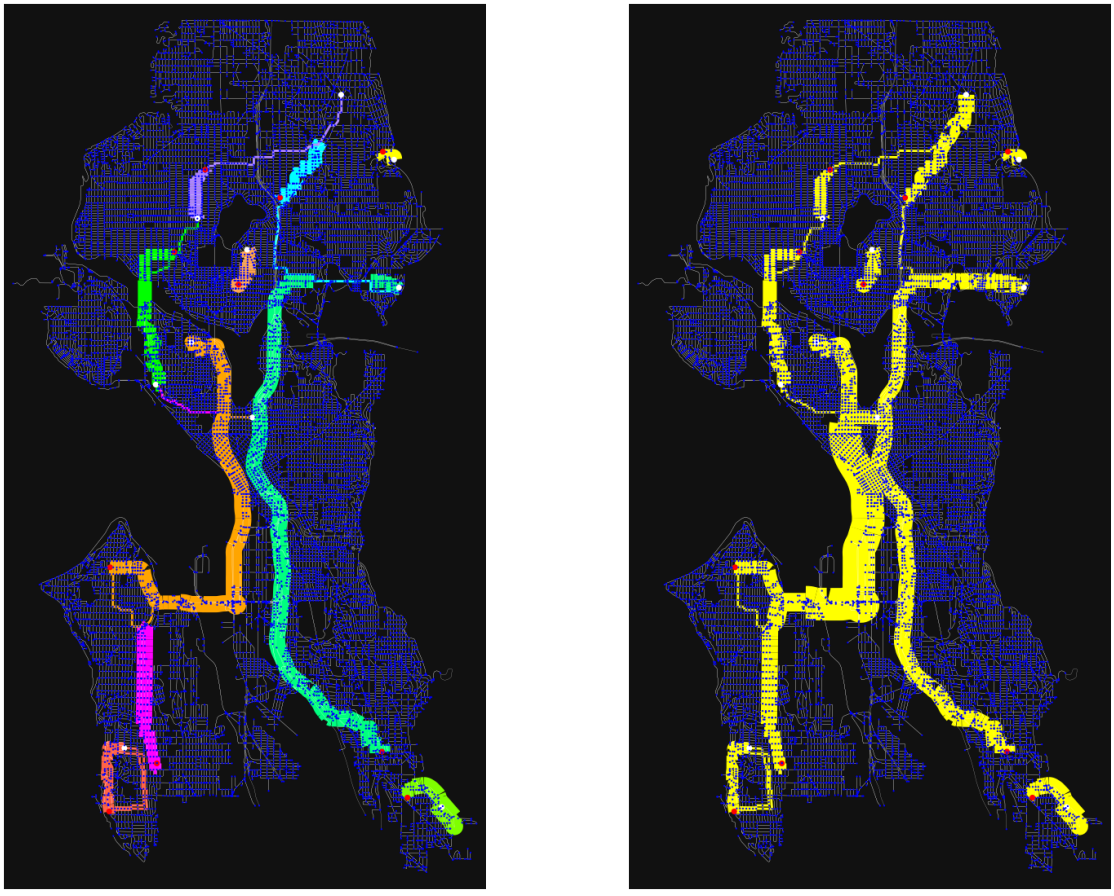
(a) Convergence of x (b) Minimum Assignment Costs

Figure 5: Comparison between Hungarian Algorithm and Random Assignments

We vary the values of A and b to study the contribution of the congestion and constant edge costs terms to the optimal solution, as shown in Figures 6 and 7. Figure 6b shows that when b is large compared to A, the

congestion does not affect the optimal solution much because the shortest path computations are dominated by the values of b. With little congestion in the network, we can observe that each agent is assigned to the optimal task quickly, and most mass clumps up on the assigned shortest paths without much exploration in Figure 6a. Consequently, the optimal solution obtained at equilibrium with b dominating is close to the shortest paths the agents would get if they only consider the edge costs given by b.

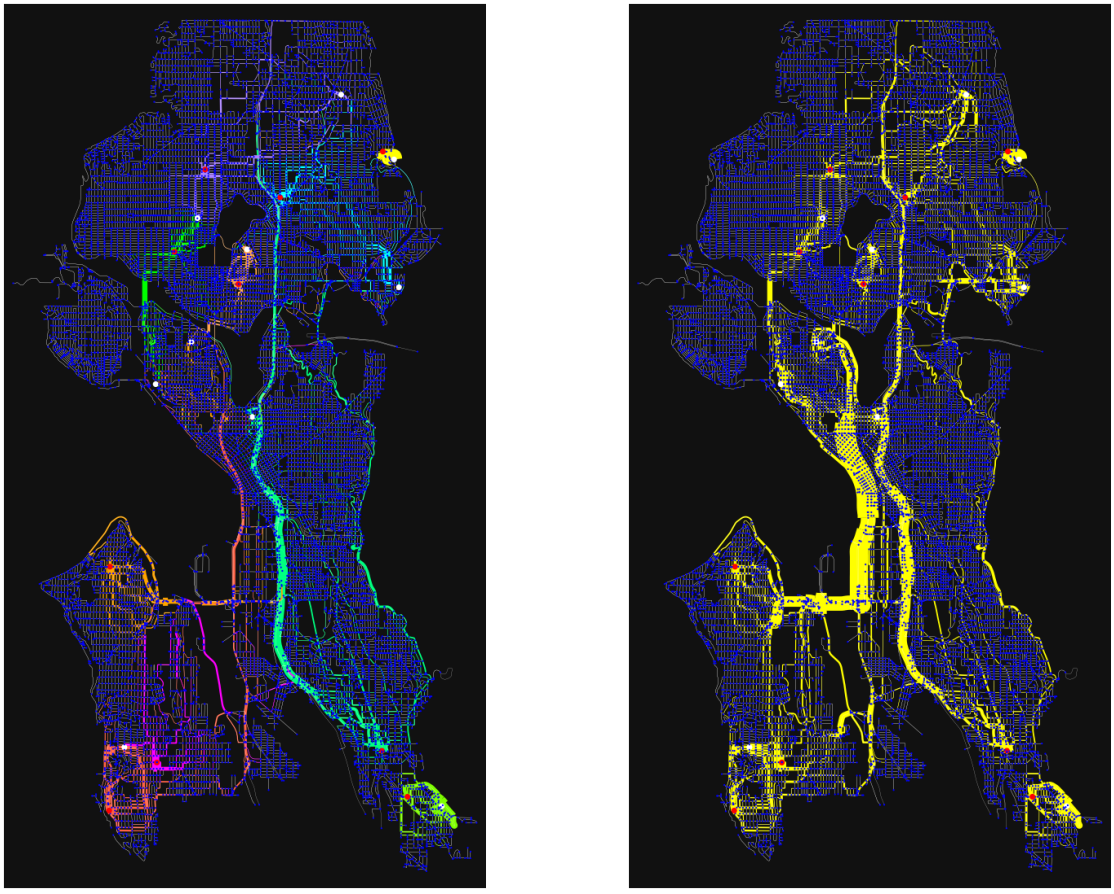
Figure 7b demonstrates that when A is larger than b, most of the latency on each edge comes from the congestion term and the mass of x shifts away from the shortest paths. With heavy congestion in the network, it is shown in Figure 7a that the mass of each agent is distributed across multiple paths, and the optimal assignment between agent i and task j varies more than the results in Figure 6a. As a result, the mass no longer clumps up on a single shortest path to avoid increasing the cost of that path significantly, so the equilibrium optimal solution tends to spread out over multiple paths with various matching assignments.



(a) Individual Edge Flow

(b) Overall Edge Flow

Figure 6: Optimal Matching and Routing Result with Little Congestion



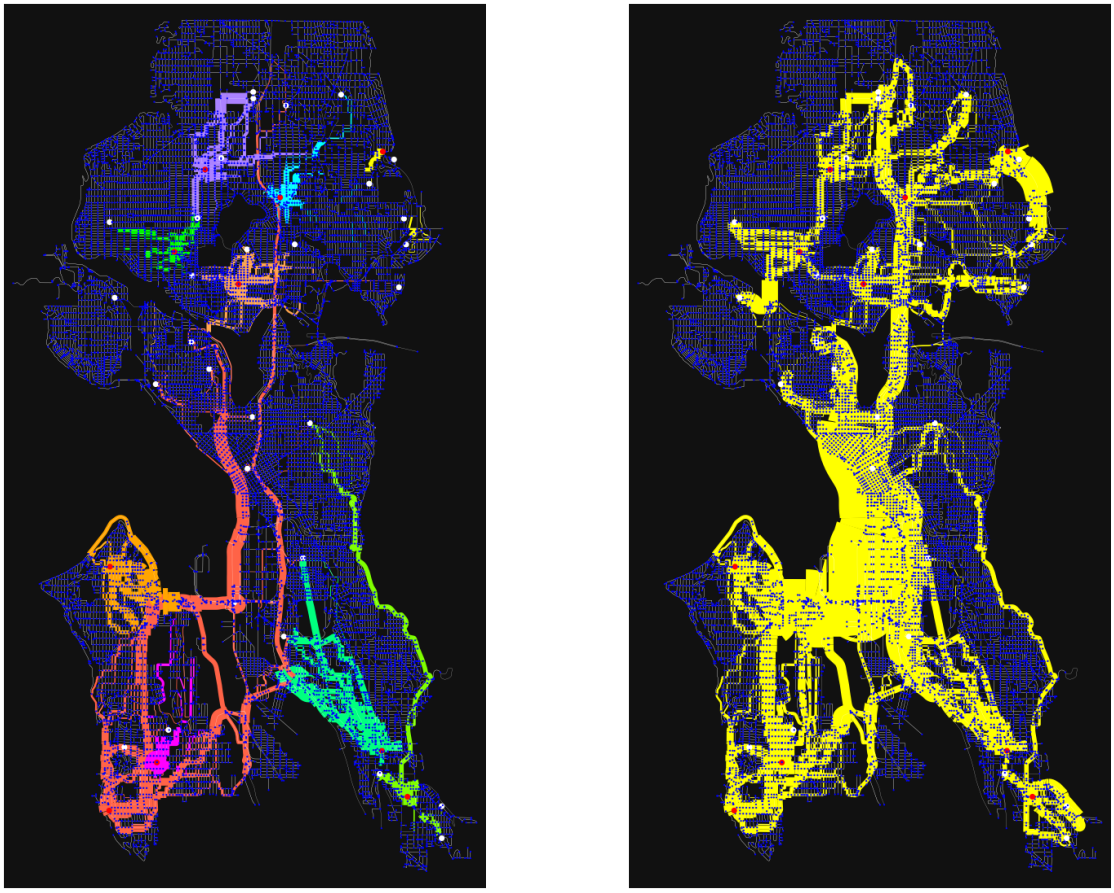
(a) Individual Edge Flow

(b) Overall Edge Flow

Figure 7: Optimal Matching and Routing Result with Heavy Congestion

Finally, we simulated unbalanced assignment with 10 agents and 30 tasks by modifying the constraint (14d) to $M\mathbf{1} = \mathbf{1} \cdot \frac{n}{m}$ and $M^\top \mathbf{1} = \mathbf{1}$, where m is number of agents and rows in M and n is number of tasks and columns in M. The constraints for the M matrix are the columns sum to 1, meaning that a task is completed by all agents, and the rows sum to the ratio of the number of tasks divided by the number of agents, meaning that the agents equally split the tasks. On top of that, we replace the Hungarian algorithm with a linear program to solve for optimal assignment with the modified constraint using the CVXPY package [39, 40]. Figure 8a highlights the shortest paths each agent traveled to each optimally assigned task in the same color. Each path has a different amount of traffic flow distributed at equilibrium to avoid congestion in traffic and achieve the minimum total travel cost. As for Figure 8b, we see the overall optimal edge flow at equilibrium in the network after combining each agent's edge flow. In addition to Figure 8, we provide Figures 9a and 9b to illustrate that our algorithm successfully finds the optimal matching and

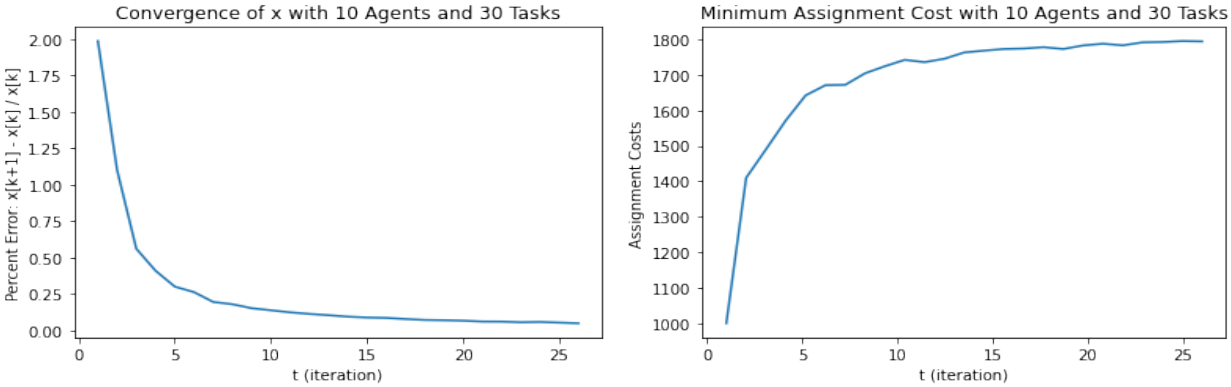
routing for unbalanced assignments as well with x and the minimum cost converging at iteration 26.



(a) Individual Edge Flow

(b) Overall Edge Flow

Figure 8: Optimal Matching and Routing Result with Unbalanced Assignments



(a) Convergence of x

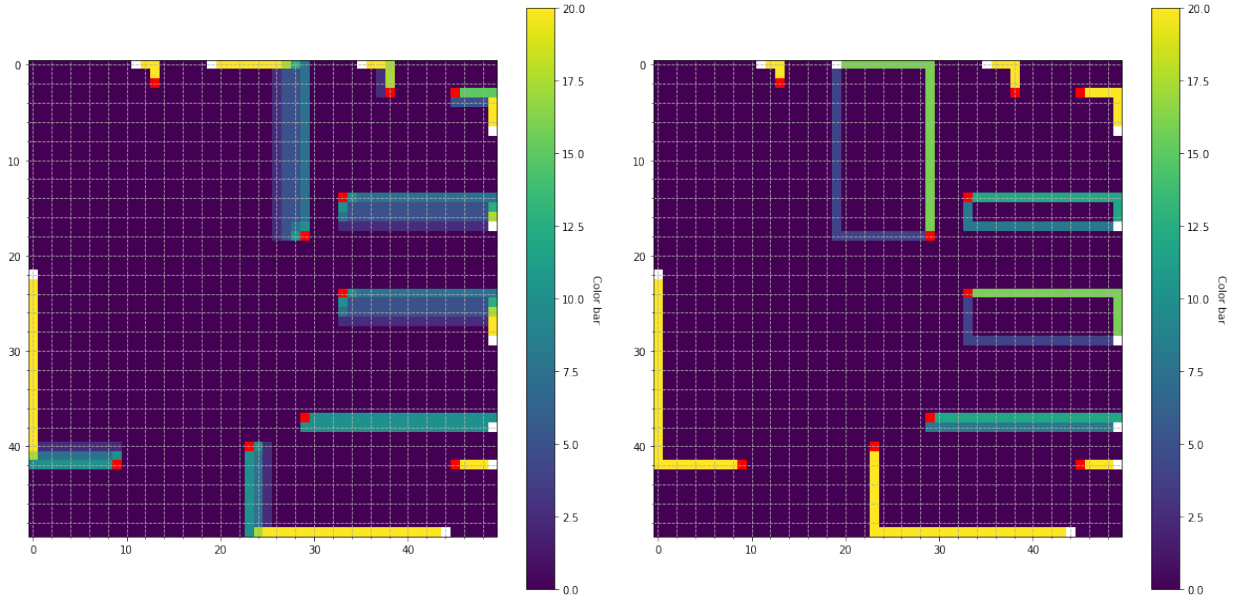
(b) Minimum Assignment Costs

Figure 9: Unbalanced Assignment Convergence of x and Minimum Assignment Costs

7.2 Multi-Robot Warehouse Problem

Similarly, we simulate the multi-robot warehouse problem in Python using NetworkX [37] by considering 10 agents and 10 tasks randomly placed in a two-dimensional 50×50 grid graph with a latency function in Equation 21. The difference in this problem is to keep track of the number of turns robots make and incorporate the information with the matching and routing problems. The goal of avoiding turns is to not only allow robots to move more efficiently, as changing their direction requires extra time but also avoid collisions, where two robots enter the same node. In our implementation, we expand all the nodes in the grid graph in five directions and expand the edges to connect nodes based on the additional direction information. Next, a turn penalty cost is included in the latency function to prevent agents from making many turns when navigating to the delivery locations.

In Figure 10, the robots are highlighted in red, and the delivery locations are highlighted in white around the boundary of the warehouse floor. Also, Figure 10 shows the optimal assignment between the agents and tasks with the corresponding shortest paths indicated by the overall edge flow value tied to each node. After comparing Figures 10a and 10b, we confirm that including the turn penalty term in the latency function allows the robots to travel to assigned delivery locations through shortest paths with fewer number of turns.

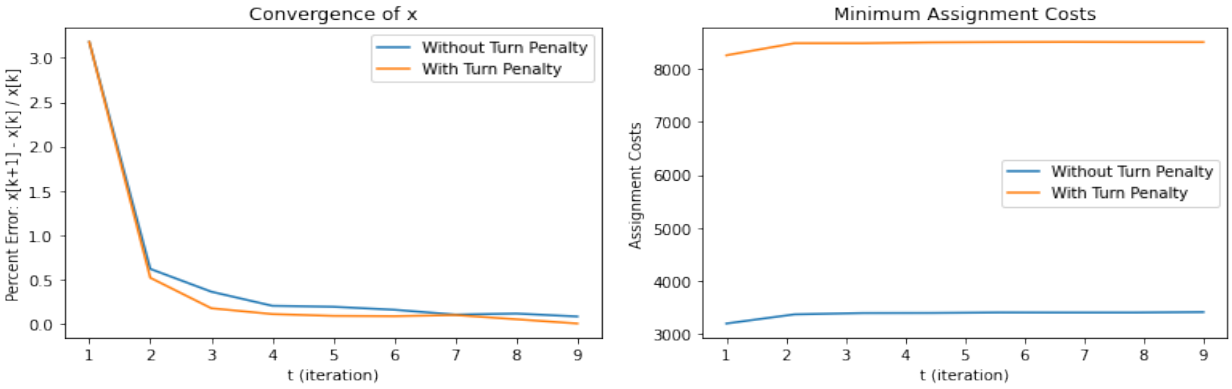


(a) Overall Edge Flow without Turn Penalty

(b) Overall Edge Flow with Turn Penalty

Figure 10: Optimal Matching and Routing Result with Balanced Assignments

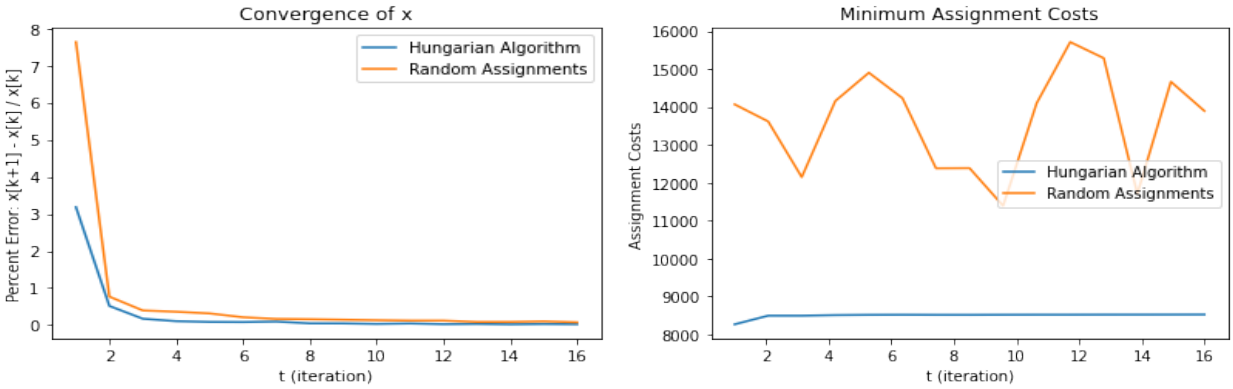
Moreover, Figures 11a and 11b show that the convergence of x and the achieved minimum assignment costs with and without penalty turns. This demonstrates that our proposed algorithm with turn penalty effectively solves the optimal matching and routing with congestion avoidance for the given agents and tasks while ensuring the robots navigate to the delivery locations in consistent directions as much as possible.



(a) Convergence of x (b) Minimum Assignment Costs

Figure 11: Comparison of Turn Penalty in Convergence of x and Minimum Assignment Costs

Same case studies are conducted where we vary the values of A and b to understand the effect of congestion and constant edge costs terms and replace the Hungarian algorithm with random assignments to provide a benchmark illustrating the effectiveness of our algorithm. Figure 13 plots the matching and routing result with random assignments and shows the robots explore non-optimal routes to assigned delivery locations, leading to the slower convergence rate of x and the higher minimum assignment costs compared to the results from our algorithm, as shown in Figure 12.



(a) Convergence of x (b) Minimum Assignment Costs

Figure 12: Comparison between Hungarian Algorithm and Random Assignments

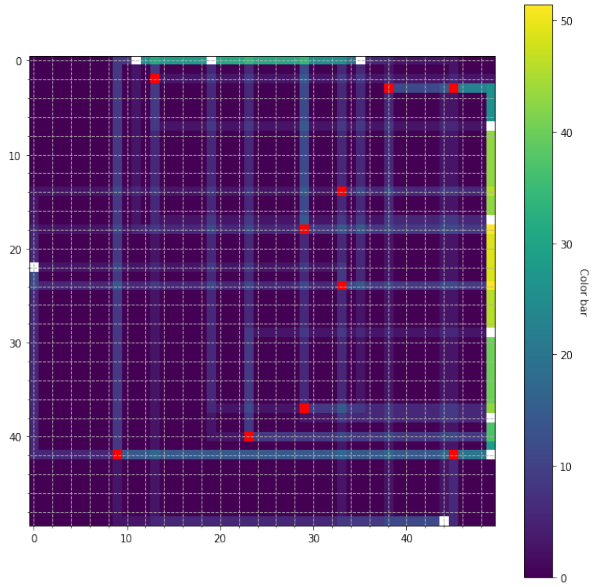
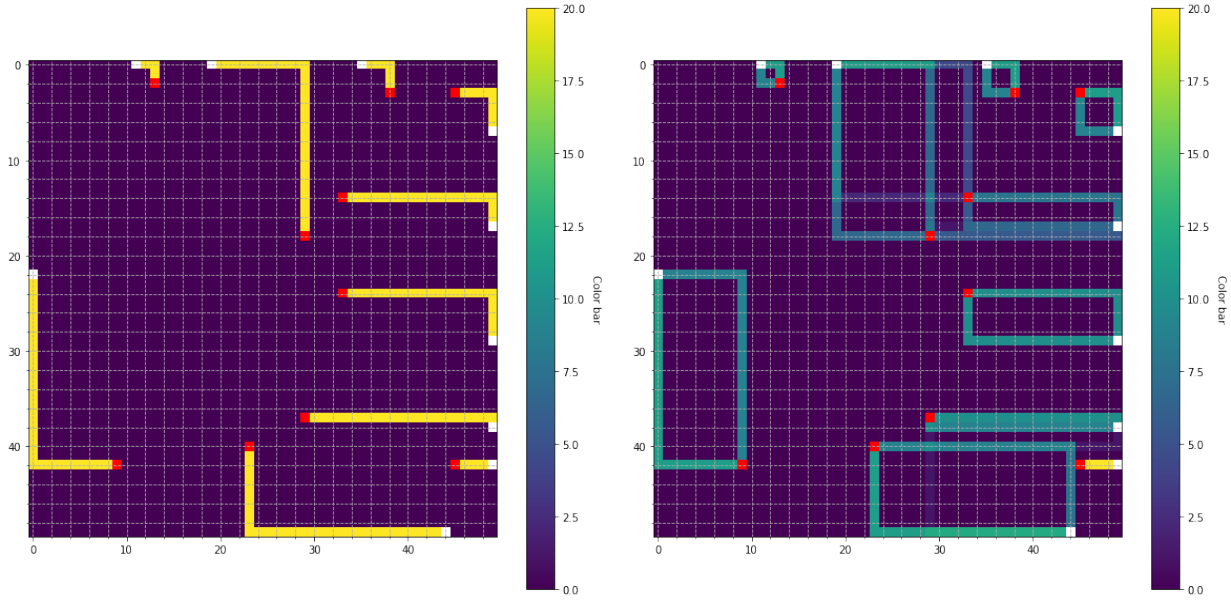


Figure 13: Overall Edge Flow with Random Assignments

Here, we observe the same behaviors, where each agent is assigned to the optimal task quickly, and most mass clumps up on a sign shortest path when little congestion is present in the network, as illustrated in Figure 14b. On the other hand, each agent gets assigned to a different task more often, and the mass no longer clumps up on a single shortest path to avoid increasing the cost of that path significantly when there is heavy congestion present, as shown in Figure 14b.



(a) Overall Edge Flow with Little Congestion

(b) Overall Edge Flow with Heavy Congestion

Figure 14: Optimal Matching and Routing Result with Balanced Assignments

Finally, Figure 15 illustrates unbalanced assignments simulated with 10 agents and 30 tasks and turn penalty included. Additionally, each robot travels through the shortest paths to the optimally assigned tasks with a minimal number of turns. Each path has a different amount of flow distributed at equilibrium to avoid congestion and achieve the minimum total travel cost. Also, Figure 16 verifies that our algorithm successfully finds the optimal matching and routing for unbalanced assignments, where x and the minimum assignment costs converge at iteration 6.

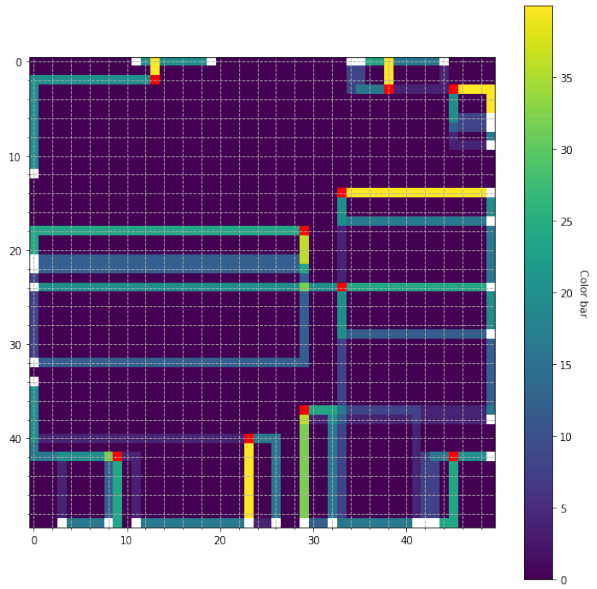
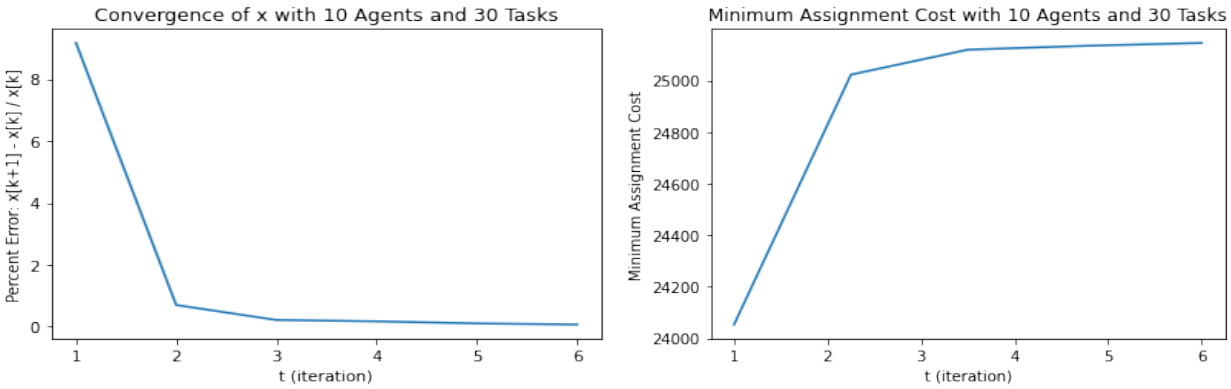


Figure 15: Overall Edge Flow for Unbalanced Assignments



(a) Convergence of x

(b) Minimum Assignment Costs

Figure 16: Unbalanced Assignment Convergence of x and Minimum Assignment Costs

8 Conclusions

In this thesis, we successfully formulated an optimization problem tackling traffic assignment, task assignment, and congestion avoidance simultaneously. By understanding the connection between the routing and matching components in our problem formulation, we developed a Frank-Wolfe-based algorithm to solve for the optimal assignment between the given agents and tasks and the associated optimal routes while considering congestion in the system. Finally, we demonstrated simulation results in the street network of Seattle City and the multi-robot warehouse system to effectively and efficiently deal with congestion, turn penalties, and unbalanced assignments on top of finding the optimal assignment and associated optimal routes. Some directions for future work would be studying the time complexity of our algorithm and reducing the computation cost. Next, our problem formulation and algorithm provide an average solution for the system at equilibrium, so we would like to extend our current framework to consider dynamic collision avoidance for multi-robot systems, which aims to avoid collisions at each time iteration. Last but not least, the current framework built acts as a centralized planner for the whole system, while there has been work focusing on fully distributed path planning and task assignment algorithms applied in the multi-robot systems. Therefore, we would like to explore combining our current algorithm with distributed algorithms implemented on each agent to increase the performance and reduce processing time for finding the optimal solutions.

References

[1] A. C. Andersen, K. Pavlikov, and T. A. Toffolo, "Weapon-target assignment problem: Exact and approximate solution algorithms," *Annals of Operations Research*, vol. 312, no. 2, pp. 581–606, 2022.

[2] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention in multirobot task allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 889–900, 2015.

[3] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472–1483, 2009.

[4] M. W. Levin, "Congestion-aware system optimal route choice for shared autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 82, pp. 229–247, 2017.

[5] F. Wu, V. S. Varadharajan, and G. Beltrame, "Collision-aware task assignment for multi-robot systems," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 30–36.

[6] M. Frank, P. Wolfe *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[9] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[10] J. G. WARDROP, "Road paper. some theoretical aspects of road traffic research." *Proceedings of the Institution of Civil Engineers*, vol. 1, no. 3, pp. 325–362, 1952. [Online]. Available: https://doi.org/10.1680/ipeds.1952.11259

[11] X. Di, R. Ma, H. X. Liu, and X. J. Ban, "A link-node reformulation of ridesharing user equilibrium with network design," *Transportation Research Part B: Methodological*, vol. 112, pp. 230–255, 2018.

[12] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research." *Proceedings of the institution of civil engineers*, vol. 1, no. 3, pp. 325–362, 1952.

[13] M. Beckmann, C. B. McGuire, and C. B. Winsten, "Studies in the economics of transportation," Tech. Rep., 1956.

[14] X. J. Ban, M. Dessouky, J.-S. Pang, and R. Fan, "A general equilibrium model for transportation systems with e-hailing services and flow congestion," *Transportation Research Part B: Methodological*, vol. 129, pp. 273–304, 2019.

[15] Z. Liu, H. Wei, H. Wang, H. Li, and H. Wang, "Integrated task allocation and path coordination for large-scale robot networks with uncertainties," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2750–2761, 2021.

[16] A. Prorok, "Redundant robot assignment on graphs with uncertain edge costs," in *Distributed Autonomous Robotic Systems: The 14th International Symposium*. Springer, 2019, pp. 313–327.

[17] P. M. Shiroma and M. F. Campos, "Comutar: A framework for multi-robot coordination and task allocation," in *2009 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2009, pp. 4817–4824.

[18] Z. Liu, H. Wang, H. Wei, M. Liu, and Y.-H. Liu, "Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1705–1717, 2020.

[19] Y. Fan, F. Deng, and X. Shi, "Multi-robot task allocation and path planning system design," in *2020 39th Chinese Control Conference (CCC)*. IEEE, 2020, pp. 4759–4764.

[20] R. Jonker and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," in *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*. Springer, 1988, pp. 622–622.

[21] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *International conference on machine learning*. PMLR, 2013, pp. 427–435.

[22] G. Laporte, "The veshicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[23] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation," *Mathematical programming*, vol. 73, no. 2, pp. 129–174, 1996.

[24] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0899825696900445

[25] M. Englert, T. Franke, and L. Olbrich, "Sensitivity of wardrop equilibria," *Theory of Computing Systems*, vol. 47, pp. 3–14, 2010.

[26] A. O'sullivan, S. M. Sheffrin, and K. Swan, "Economics: Principles in action," 2003.

[27] G. Sharon, S. D. Boyles, S. Alkoby, and P. Stone, "Marginal cost pricing with a fixed error factor in traffic networks." in *AAMAS*, 2019, pp. 1539–1546.

[28] L. Olbrich, "Aspects of wardrop equilibria," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:19365931

[29] A. Pigou, *The economics of welfare*. Routledge, 2017.

[30] R. Burkard and E. Çela, *Linear assignment problems and extensions*, 1st ed., ser. Supplement Volume A. Netherlands: Kluwer Academic Publishers, 1999, pp. 75–149.

[31] S. Martello and P. Toth, "Linear assignment problems," in *North-Holland Mathematics Studies*. Elsevier, 1987, vol. 132, pp. 259–282.

[32] R. T. Rockafellar, *Convex analysis*. Princeton university press, 1997, vol. 11.

[33] G. Birkhoff, "Three observations on linear algebra," *Univ. Nac. Tacuman, Rev. Ser. A*, vol. 5, pp. 147–151, 1946.

[34] M. Patriksson, *The traffic assignment problem: models and methods*. Courier Dover Publications, 2015.

[35] G. Carpaneto and P. Toth, "Primal-dual algrorithms for the assignment problem," *Discrete Applied Mathematics*, vol. 18, no. 2, pp. 137–153, 1987.

[36] J. Goldberger and T. Tassa, "A hierarchical clustering algorithm based on the hungarian method," *Pattern Recognition Letters*, vol. 29, no. 11, pp. 1632–1638, 2008.

[37] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[39] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[40] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.