RISK-AWARE MULTI-OBJECTIVE OPTIMIZATION FOR GRAPH TRAVERSALS

A REPORT

submitted in partial fulfillment of the requirements

for the award of the dual degree of

Bachelor of Science-Master of Science

in

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

by

RISHABH BHONSLE

(17217)

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH BHOPAL
BHOPAL - 462066

April 2022

CERTIFICATE

This is to certify that **Rishabh Bhonsle**, BS-MS (Electrical Engineering and Computer Science), has worked on the project entitled **'Risk-Aware Multi-Objective Optimization for Graph Traversals'** under my supervision and guidance. The content of this report is original and has not been submitted elsewhere for the award of any academic or professional degree.

April 2022 **Dr. Sujit PB**

IISER Bhopal

Committee Member	Signature	Date

ACADEMIC INTEGRITY AND COPYRIGHT DISCLAIMER

I hereby declare that this project is my own work and, to the best of my knowledge, it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution, except where due acknowledgement is made in the document.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright Act (1957) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless IISER Bhopal from any and all claims that may be asserted or that may arise from any copyright violation.

April 2022 **Rishabh Bhonsle**
IISER Bhopal

ACKNOWLEDGEMENT

First and foremost, I express my sincere gratitude to my mentor and thesis supervisor Dr Sujit PB for providing me with such an exciting and challenging problem to work on and guiding me through it. I thank him for believing in my naive self, who had approached him for academic opportunities, and for introducing me to the world of robotics, where I discovered my gusto amidst its endless possibilities. Through his teachings, patience and empathy, he instilled a sense of confidence in my academic capabilities. I also thank him for his subtle humour and lab gatherings over coffee breaks that made working at MOON Lab ever so enjoyable.

I express my unfeigned appreciation toward Dr Snigdha Thakur and Dr Aditya Medury for mentoring, guiding and paving my undergraduate journey at IISER Bhopal and never letting me lose my focus on its way. I also thank Dr Sujata Ghosh, who kindled the spirit of research in me during my first ever internship at ISI Chennai.

In light of my scientific growth and collaborations with this project, I foremost thank Agamdeep Singh for his enthusiastic discussions and contributions to the project. I extend this gratitude to Rishab Balasubramanian for helping me with the basis of the problem's implementation and to Kshitij Srivastava for providing me with the custom dataset required for modelling the problem. I am also grateful to my colleagues turned friends at MOON Lab - Prithvi, Prakrit, Kasi, Prateek, Rajat, Kartikeya, Manav and Swadhin - for creating and maintaining a fun, flexible, collaborative and co-learning environment.

Outside the world of academia, I have been fortunate to have a plethora of friends which have supported and motivated me through the various walks of life during my half a decade at IISER Bhopal. Naming all of them and the many ways in which they impacted my life would require a thesis in its own. Amongst those however, Shreyas, Mansi, Anyesh, Tanvi, Tayade and Rajnish hold a special place in my heart for always being there in my

happiness, setbacks and everything in between.

Last but not least, I express my heartfelt gratitude toward my parents and my brother for their unconditional love, trust and support, which enabled and motivated me to push my limits. Their rationale approach to life has been pivotal in developing my ability to face challenges confidently and to enjoy life and science in its entirety.

Rishabh Bhonsle

ABSTRACT

Decision making through combinatorial optimization is widely studied and applied for planning and operations research. This thesis aims to incorporate risk in this decision-making process under the uncertainty of real-world data. The problem is formulated using risk-aware submodular optimization based on the Conditional-Value-at-Risk (CVaR) risk metric. The proposed solution to this problem is inspired by the Risk-aware Greedy Algorithm (RAGA) that was used to approach the risk-aware submodular optimization for the stochastic travelling salesman problem.

In recent times, many government organizations and public-facing companies have started incorporating Open Data Platforms for transparency and open-source collaboration for collective development of society. In a similar expedition, this thesis models a case to fulfil its aim using the Open Transit Data Platform of Delhi. This case model allows users to choose one or more transit objectives like travel time, fare cost, user safety, etc. This goal is fulfilled by incorporating the optimization formulation to develop a time and space-constrained risk-aware dynamic pathing algorithm.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

1.1 Motivation and Background

Combinatorial optimization forms a core and important area in computer science and operations research. It aims to search for an optimum object from a finite collection of objects (like a graph), where the set of feasible solutions is or can be reduced to a discrete set [8]. In particular, combinatorial optimization problems over graphs have attracted considerable interest from the theory and algorithm design communities over the years. Of Karp's 21 problems in the paper on reducibility among combinatorial problems [9], 10 are decision versions of graph optimization problems. Most of the other 11 problems can be naturally formulated on graphs.

Problems of combinatorial optimization find a variety of applications in robotics, examples of which as given in [3] include sensor placement for environment monitoring, robot-target assignment and tracking, informative path planning, transportation, social network, telecommunications and scheduling. However, in real-world scenarios where the environment is dynamic, and situations are uncertain, networks can face congestions, sensors can fail or get compromised [10], or robots may not know the exact positions of the targets [11]. This research in this thesis thus focuses on the problem of incorporating uncertainty for combinatorial optimization over graphs.

A particular case study considered in this thesis is to provide the freedom to choose transit objectives to public transit users. Public-transit systems run in the shadows of most developing countries due to their poor deployment even though they continue to upgrade their services. Commercially available mapping systems incorporate congestion for planning optimal routes between

start and end locations but provide congestion data independently in the case of public transits. Moreover, these systems also lack multi-destination tour planning support, which could be a valuable tool for tourists or door-to-door service agencies. Apart from attempting to solve these issues, this case study attempts to take it a notch further by providing multi-objective support to public transit users. Here are a few examples to illustrate this point further:

- Based on traffic data, a shorter but congested path might encounter a longer travel time but a smaller fare over a longer uncongested path.
- A user new to a given transit region might want to avoid travelling through deserted regions for personal safety while maintaining minimum deviation from optimal travel times or fare costs.

In an attempt to solve these decision-making problems, the algorithms developed to search for these multi-objective paths must incorporate uncertainty. While extensive research exists on the decision making part, few include planning under uncertainty. Decisions made in the face of the same are usually penalized by a stochastic cost. The uncertainty is then accounted for using a risk-metric, which fulfils the six axioms for risk in robotic applications [12] on this stochastic cost.

Partly in pursuit of exploring the field of combinatorial optimization problems over graphs and partly out of interest in understanding risk-aware applications that make robotics life-like, this thesis is presented in two parts-

- Re-creation and further generalizations of an existing research problem on risk-aware optimization for combinatorial decisions under uncertainty. [4]
- Modeling, development and implementation of novel applications on real-world data using risk-aware graph-traversal algorithms through a case study.

1.2 Definitions and Interpretations

This section introduces definitions used throughout to build a formal structure to attempt to solve the aim of this thesis. For the purpose of clarity, interpretations of each definition are provided, some of them through examples and illustrations, to understand its context better upon encounter.

To set a standardised base for the definitions, consider a ground set X and a set function $f : 2^X \mapsto \mathbb{R}$ defined on this ground set. The set function thus assigns to each subset S of X a value $f(S)$. Assume $f(\phi) = 0$.

Definition 1.1. Monotonicity [13]
Consider two any two subsets S and S' of X such that $S \subseteq S' \subseteq X$. f is said to be monotone (non-decreasing) if and only if

$$f(S) \leq f(S')$$

Definition 1.2. Submodularity [13, Proposition 2.1]
Consider two subsets S and S' such that $S, S' \in X$. f is said to be submodular if it satisfies

$$f(S) + f(S') \geq f(S \cup S') + f(S \cap S')$$

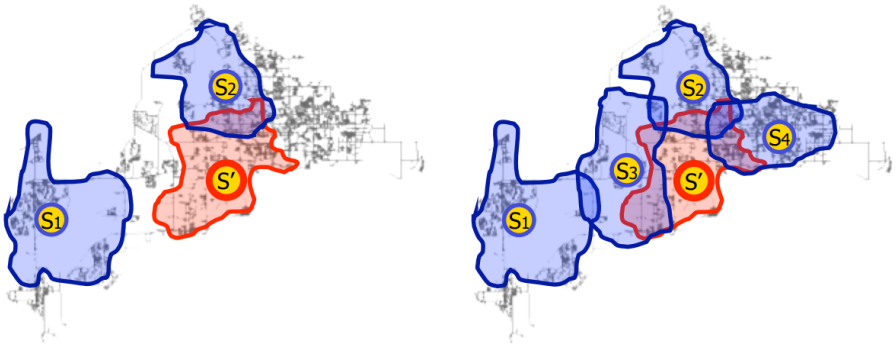
An equivalent definition thus follows that for any element $s \in X$ and $s \notin S'$, f is said to be submodular if it satisfies

$$S \to f(S \cup \{s\}) - f(S) \text{ is non-increasing}$$

$$f(S \cup \{s\}) - f(S) \geq f(S' \cup \{s\}) - f(S')$$

Submodular functions are used in cases where solutions to naturally occurring problems are not tractable and thus require strong guarantees on their approximate solutions. For this, most optimization problems are formulated using convexity or submodularity. Submodularity also possesses the property of subadditivity [13].

Informally, submodularity portrays the principle of diminishing returns.



(a) *Adding s' to set $\{s_1, s_2\}$* (b) *Adding s' to superset $\{s_1, \ldots, s_4\}$*

Fig. 1.1: Visualizing Submodularity [1]

It states that adding an element to a smaller set has more value than adding it to a larger set. This can be seen with the help of Figure 1.1 where s_i represent sensors and the regions marked in blue and red represent the respective regions covered by them. Common examples of submodular monotone functions are entropies, concave functions of cardinality, and matroid rank functions. An in-depth review of the same can be found in [14].

Definition 1.3. Matroid [8, Section 39.1]
Let I be a collection of subsets of X. The pair (X, I) is called a matroid if and only if it satisfies the following conditions:

- atleast one subset of X is independent $I \neq \phi$
- for any set $S \subseteq X$ it must hold that $S \in I$, and for any set $P \subseteq S$ it must hold that $P \in I$ (hereditary property, or the downward-closed property)
- for any sets $S, P \in I$ and $|P| \leq |S|$, it must hold that there exists an element $s \in S \backslash P$ such that $P \cup \{s\} \in I$ (augmentation property or the independent set exchange property)

Matroids are an algebraic structure used to model many real world graph problems of combinatorial optimization. As an illustration of the concept, consider X to be the set of all edges in the un-directed graph shown in Figure

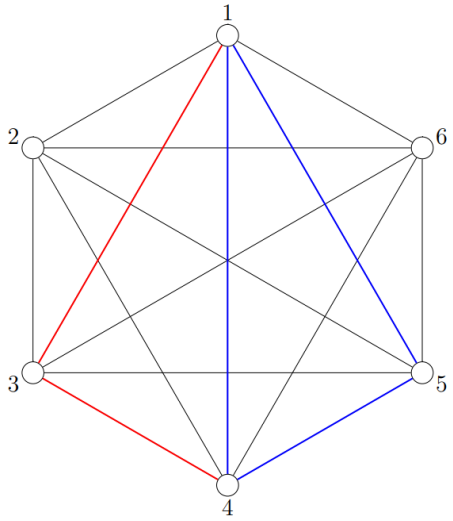


Fig. 1.2: Visualizing Matroids [2]

1.2. A cycle in a graph is a path that begins and ends at a common vertex (sub-graph coloured in blue). Let I be the set of all connected edges that do not form a cycle (sub-graph coloured in red). The set (X, I) is a matroid.

Remark 1.4. For the context of further definitions and the proofs and algorithms built on them, the utility function f is considered to incorporate a noise parameter y, which induces the distribution of the random variable $f(S, y)$.

Remark 1.5. For the purpose of risk-metrics, a risk parameter $\alpha \in (0, 1]$ is defined where $\alpha = 1$ is risk-neutral and $\alpha \approx 0$ is very conservative.

Definition 1.6. Value at Risk ($VaR_\alpha(S)$) [15] [16]
With respect to a specific probability level α, VaR of a given set is the lowest amount τ such that with probability α, the loss incurred will not exceed τ

$$VaR_\alpha(S) = \min_{\tau \in \mathbb{R}} \{\mathbb{P}[f(S, y) \leq \tau] \geq \alpha\}$$

Thus, as shown in Figure 1.3, $VaR_\alpha(S)$ denotes the left endpoint of the α-quantile(s) of the random variable $f(S, y)$.

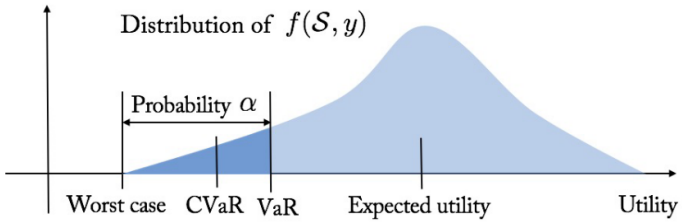


Fig. 1.3: Visualizing VaR and CVaR [3]

Definition 1.7. Conditional Value at Risk ($CVaR_\alpha(S)$) [16] [17]
$CVaR_\alpha(S)$ is the conditional expectation of this set of α-worst cases of $f(S, y)$

$$CVaR_\alpha(S) = \mathbb{E}_y[f(S, y)|f(S, y) \leq VaR_\alpha(S)]$$

Maximizing the value of $CVaR_\alpha(S)$ is equivalent to maximizing the auxiliary function $H(S, \tau)$ [16, Theorem 2]:

$$H(S, \tau) = \tau - \frac{1}{\alpha}\mathbb{E}_y[(\tau - f(S, y))^+]$$

where $[t]^+ = t, \forall t \geq 0$ and 0 when $t < 0$.

Definition 1.8. Travelling Salesman Problem (TSP)
Given a complete graph $G(V, E)$, the objective of the TSP is to find a minimum cost (maximum reward) Hamiltonian cycle.

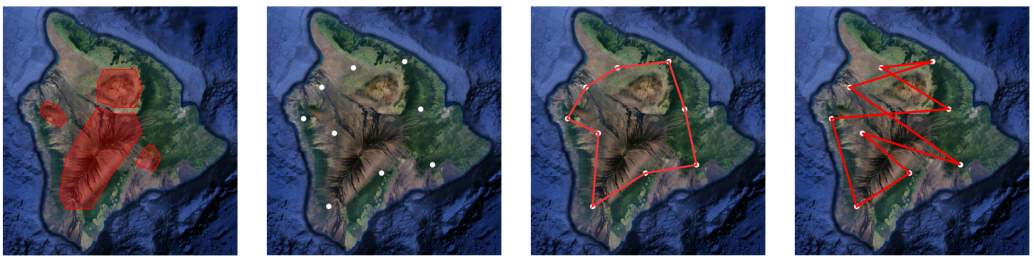
Definition 1.9. A-star Search
A-star (also referred to as A*) search is an informed graph traversal and path search algorithm that uses information about path cost (distance travelled, shortest time, etc.) and heuristics to find the an optimal path starting from a specific starting node of a graph to the given goal node.

1.3 Problem Introduction

1.3.1 Part 1 - Research and Exploration

The work presented in the first part is based and built upon the paper "Risk-Aware Sub-modular Optimization for Stochastic Travelling Salesperson Problem" by Rishab Balasubramanian, Lifeng Zhou, Pratap Tokekar, and P.B. Sujit [4]. This paper introduces a risk-aware variant of the TSP. A touring agent aims to optimize its costs and rewards along the tour simultaneously while exhibiting diminishing marginal gains (are sub-modular) and being subjected to uncertainty (are stochastic) in both. This problem can be better understood by taking an example from the same paper.

Consider an aerial robot tasked to monitor active volcanic regions represented by red polygons over a given terrain, as shown in Figure 1.4 (a). The white dots in Figure 1.4 (b) represent the strategic points of surveillance that the robot must visit to monitor this activity from a safe distance. The robot travels between these sites to collect information and receives a reward. Examples of such travel paths are represented by red segments, as shown in Figures1.4 (c) and (d). The travel path is not constrained by distance or time.



(a) An image of an island, with the locations of active volcanoes highlighted in red

(b) Key sites of surveillance surrounding the volcanic sites

(c) Tour with low cost and low reward

(d) Tour with high cost and high reward

Fig. 1.4: Problem Motivation Example [4]

As the robot travels above the red regions, it has a higher chance of failure or malfunction (higher risk) due to higher volcanic activity but also gains more information (higher reward) about the same. Due to this, the robot can travels through longer paths (higher cost), wanting to collect as

much information as possible, as represented in Figure 1.4 (d). On the other hand, as the robot avoids travel over red regions, it has a lesser chance of failure or malfunction (lower risk) but also loses out on information (lower reward) that can be detected. Due to this, the robot seeks to travel across the outer edges of the volcanic region, avoiding information gain and hence resulting in a shorter path (lower cost). The objective is to find a suitable tour for a single robot while considering the risk threshold and the trade-off between cost and reward for paths independent of time and distance.

The solution to the problem is provided by the risk-aware greedy algorithm (RAGA), which runs in polynomial time [4, Theorem 2] and is within a constant factor of the optimal with an additive term that depends on the value of optimal solution [4, Theorem 1]. The first sub-part of this part thus re-implements the code for this problem by optimizing the code based on calculated assumptions while keeping its approximation guarantee intact. In the second sub-part, this work is leveraged to develop and implement algorithms for its further generalizations and extensions.

Contributions:

- Re-implementation of RAGA based on minor modifications for data generation for a faster run-time complexity.
- Developing and implementing a multi-agent risk-aware greedy algorithm (mRAGA) building on the single-agent variant.
- Generalizing the RAGA algorithm to incorporate a sensing radius for the agent for practical implementations.
- Developing and implementing a multi-depot risk-aware greedy algorithm (MDRAGA) building on the single-depot variant.

1.3.2 Part 2 - Real-world Implementation

The work presented in the second part focuses on developing and implementing algorithms for path-planning in real-world networks for a given case study, as stated in section 1.1. The model uses the transit datasets for buses in Delhi (DIMTS) published by Open Transit Data (OTD), Delhi, an initiative by the Department of Transport (Govt of NCT of Delhi) in association with IIIT-Delhi [18]. The goal is to thus develop path-planning algorithms for agents to travel between given destinations based on user objectives under uncertainty due to traffic.

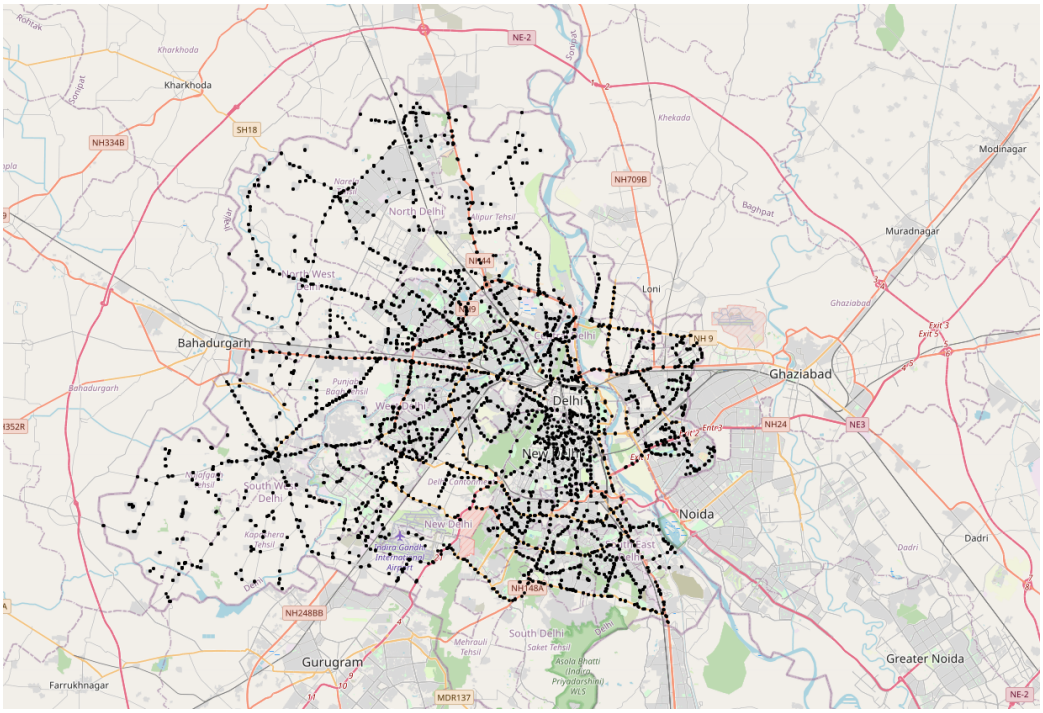


Fig. 1.5: Bus-stop Nodes Distribution by OTD

Figure 1.5 shows a plot of given bus-stops distributed across Delhi. An agent (here passenger) can travel between these stops via busses. These busses, and hence the route determined by them, would depend on the time the agent chooses to be at a particular bus stop. Secondly, every agent is assigned a halt parameter (hp), defined as the amount of time an agent is willing to stay at a particular stop for the possibility of switching between

busses and thus possible bus routes.

Figure 1.6 shows a plot of the traffic heatmap for a half-hour bin. This heatmap changes with time-based on the congestion encountered by the busses on their respective routes. This congestion can cause delays in bus schedules, thus resulting in a long time for congested shorter routes over uncongested longer routes.

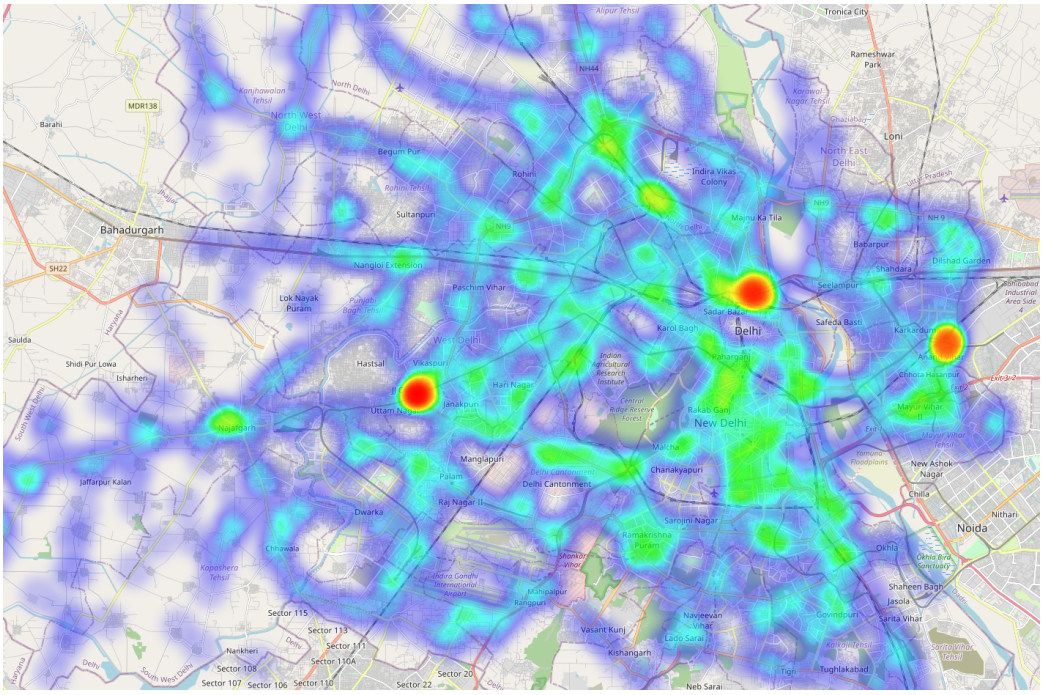


Fig. 1.6: Bus Congestion Distribution Heatmap

This part of the thesis thus aims to develop and implement a time-constrained risk-aware dynamic A-star search to find the best time-dependent route for the agent while incorporating risk due to congestion in a dynamic environment to fulfil the transit user's objectives. These multi-objectives can be modelled by variation in the values of two mathematical objectives associated with the agent for path planning - time-cost of travel based on traffic congestion and fare-cost of travel. The emphasis assigned to each cost by a weighing parameter thus determines the user objective. For example, a user-focused on economic travel over time of travel would desire a fare-cost minimization with disregard for time-cost, while a user wanting to travel

safely through fairly populated areas but also choosing the optimal route would prefer a balance between the two.

Below provided are the limitations of today's mass used mapping systems (like Google Maps) for public transits and the benefits of incorporating risk in these mapping systems.

Mapping Systems Functions	**Risk Incorporation Benefits**
Providing start location to destination location route plans of various public transits by independently displaying estimated journey time and traffic data. Possible options only include upcoming busses/metros for the shortest possible routes and do not consider congestion or user-defined objectives.	Providing start location to destination location route plans for various public transits based on user-defined objectives of reducing fare, avoiding congestion, wanting to traverse through fairly populated routes for safety purposes during odd-hours or at unknown locations, etc.
Traffic data incorporation uses real-time data.	Traffic data incorporation is regardless of real-time data availability (prediction-based).
Option for adding stops for multi-location touring (for sightseeing or delivery services) in order is restricted by user-input which is only available for private transportation.	Option for multi-location touring by optimizing the order of locations to be visited based on user-objective (reducing fare costs, distance, time, etc.) by incorporation of risk-aware path-planning

Contributions:

- Study and analysis of real-world data provided in GTFS format.
- Implementing a time-constrained risk-aware dynamic A-star search algorithm based on RAGA.
- Proposing a risk-aware submodular TSP algorithm based on edges generated by the previous algorithm.

1.3.3 Dataset Overview

Following up on section 1.3.2, this section provides a rough introduction to the data that is used to model the risk-aware optimization problem on real-world networks. As stated previously, the information being used is the transit datasets for buses in Delhi (DIMTS) published by Open Transit Data (OTD) [18]. This open data platform shares transit data in a uniform format that includes transportation schedules, associated geographic information and real-time data via Google Transit Feed Specification (GTFS) [19] and GTFS Realtime [20].

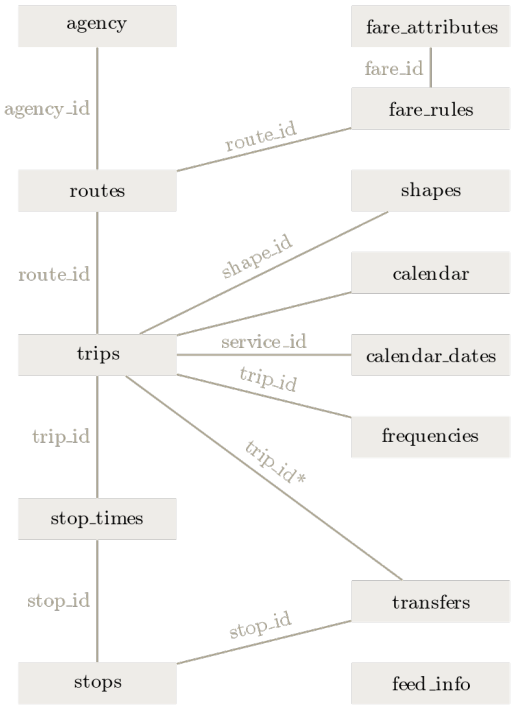


Fig. 1.7: Inter-dependent Fields of GTFS [5]

GTFS Static Feed [19]: This is a collection of inter-dependent tables (CSV files with extension .txt) that describe the transit's scheduled operations. The inter-dependency can be visualized with the help of a simple representation as shown in Figure 1.7. The GTFS tables that OTD Delhi provides are agency.txt, calender.txt, fare_attributes.txt, fare_rules.txt, routes.txt, stops.txt, stop_times.txt and trips.txt. The data represent trips

that busses involved in transit can be a part of. Each trip consists of a sequence of stops and has a corresponding unique bus assigned to it. It approximately traverses these stops at given arrival and departure times corresponding to each stop. A graph network is thus constructed using bus stops (stops.txt) as nodes, and their stop reachability in sequence, as stated in trips.txt, to form corresponding edges between these nodes. fare_attributes.txt, fare_rules.txt and stop_times.txt are used to construct stochastic costs.

	vehicle_id	lat	lng	speed	route_id	trip_id	timestamp	date	time
0	DL1PD5111	28.5534191131592	77.0547790527344	0.0	484.0	484_19_10	1638383337	2021-12-01	00:00:09
1	DL1PD5067	28.5687808990479	76.975830078125	0.0	276.0	276_20_40	1638383339	2021-12-01	00:00:09
2	DL1PD4578	28.7004280090332	77.0348663330078	0.0	749.0	749_19_00	1638383389	2021-12-01	00:00:09
3	DL1PD4652	28.7004108428955	77.0346832275391	0.0	50.0	50_19_10	1638383383	2021-12-01	00:00:09
4	DL1PD5114	28.7004375457764	77.0344619750977	0.0	1232.0	1232_19_00	1638383394	2021-12-01	00:00:09
5	DL1PD4376	28.7004261016846	77.0345077514648	0.0	830.0	830_18_20	1638383390	2021-12-01	00:00:09
6	DL1PD5050	28.7004203796387	77.0347442626953	0.0	881.0	881_19_30	1638383397	2021-12-01	00:00:09
7	DL1PD2289	28.6838817596436	77.330810546875	0.0	371.0	371_18_50	1638383390	2021-12-01	00:00:09
8	DL1PD5343	28.5534152984619	77.0531692504883	0.0	873.0	873_20_40	1638383398	2021-12-01	00:00:09
9	DL1PD4925	28.7005386352539	77.0353164672852	0.0	557.0	557_21_20	1638383394	2021-12-01	00:00:09

Fig. 1.8: GTFS Realtime Feed Database

GTFS Realtime Feed [20]: This is a collection of database files (files with extension .db) corresponding to the number of days the data was acquired (here 31 - December 1, 2021 to December 31, 2021). These database files were acquired in collaboration with Kshitij, who worked with similar files for his thesis [21] at IIIT Delhi. These files contain a recorded database of GPS ping locations, timestamps during transits of different busses for corresponding trip ids, and related fields as shown in Figure 1.8. Static Feed Data represents trips as an ordered sequence of stops, whereas Realtime Feed Data represents trips as a timestamped sequence of GPS ping locations of a bus traversed along the trip for a given day. Realtime Feed Data is independent of the bus-stop locations.

2. PROBLEM FORMULATION

The following section aims to mathematically formulate the objective function for both parts of this thesis based on the [problem introduction in section 1. This section begins by defining the primary objective function for mathematically modelling the problem. It was also stated in section 1 earlier that uncertainty in real-world networks are accounted for using a risk metric. The second sub-section below thus asks and answers the question as to why consider risk for uncertainty? The section concludes with incorporating a risk-metric into the defined objective function and formulating the final base problem upon which algorithmic solutions are attempted.

2.1 The Objective Function

Consider a graph $G(V, E)$ where V represents the set of vertices (like the surveillance sites represented by white nodes in Figure 1.4 (b)) and E represents the set of edges (like paths between the sites represented by red segments in Figures 1.4 (c) and (d)) of the graph G. Let X be the ground set consisting of the set of nodes and edges of a graph G and I be the matroidal constraint. I is notationally defined as $2^{A_1} \cup 2^{A_2} ... \cup 2^{A_n}$ where $A_1, A_2, ...A_n$ each contains edges forming n possible tours. The constraint on these tours is being Hamiltonian in case of part 1 or directed paths in case of part 2.

Let $S \subseteq X$ such that it satisfies the matroidal constraint I be one such required tour. A utility function f of this subset S quantifies the benefit received corresponding to the selection of S represented as $f(S)$.

The optimization problem is thus defined on the choice of the set S that maximizes the utility function f while satisfying the matrodial constraint I

by a common objective function modelled over graph networks similar to the one discussed in [4] as

$$\max_{S \in I, S \subset X} f(S)$$

The traditional way of stochastic optimization using the expected utility as the objective function is considered to account for uncertainty. Thus a random variable $y \in Y$ independent of S is introduced. The utility function thus becomes $f(S, y)$ where f is monotone submodular in $S \in X$ and integrable in y. Thus the objective function is represented as

$$\max_{S \in I, S \in X} \mathbb{E}_y[f(S, y)]$$

As submodular functions have the property of subadditivity, $\mathbb{E}_y[f(S, y)]$ is also submodular and monotone in S.

For the purpose of incorporating multi-objectivity in the utility function, a weighting parameter $\beta \in [0, 1]$ is defined. Let cost objectives and reward objectives be random variables with C as the cost upper-bound per edge. $r(S, y_r)$ is the reward objective for a set of edges S with y_r uncertainty and similarly $c(S, y_c)$ is the cost objective for a set of edges S with y_c uncertainty. Thus a general utility function is defined as

$$f(S, y) = (1 - \beta)r(S, y_r) + \beta(|S|C - c(S, y_c))$$

This utility function is used in the case f Part 1 of the thesis. However, in the case of Part 2, both objectives are cost based. The utility function is thus defined as

$$f(S, y) = (1 - \beta)c_1(S, y_1) + \beta c_2(S, y_2))$$

This utility can thus be maximized or minimized depending on the combination of rewards or costs.

2.2 Why Consider Risk?

Expected utility is a good metric for defining an optimizing function that incorporates uncertainty. However, it has its drawbacks. This is portrayed with the help of an example as cited in [3].

Consider the case of two mobility-on-demand vehicles (red and blue) to pick up a passenger at a demand location, as shown in Figure 2.1

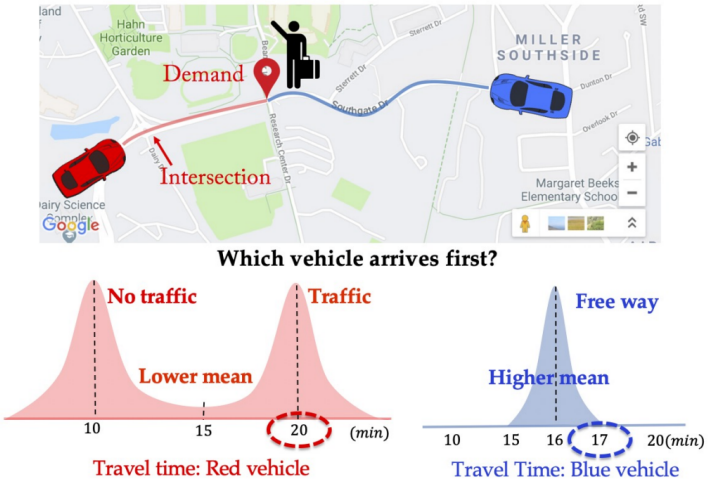


Fig. 2.1: Expected Utility Drawbacks [3]

The blue vehicle is visually further from the demand location than the red vehicle. However, the latter has to cross an intersection which consists of a traffic stop. Depending on the traffic encountered at the stop, the red vehicle will thus have a bimodal arrival time distribution compared to the unimodal time distribution for the blue vehicle. Based on expected utility, the red vehicle (mean = 15mins) beats the blue vehicle (mean = 16 mins). However, the red vehicle has much longer worst-case possibilities ($\approx$ 20 mins) than the blue vehicle ($\approx$ 17 mins).

Thus a balance between the expected utilities and worst-case scenarios must be made. Based on the inspiration taken from stock-market portfolio management, this is taken care of by using risk-aware measures. For the given purpose, the coherent risk-measure $CVaR$, as defined in Definition 1.7 is considered to be a good choice owing to its suitable properties of sub-

additivity and convexity, which other similar risk-metrics lack. An in-depth explanation of the choice of $CVaR$ over other risk measures can be found in [16]

2.3 Incorporating Risk

Incorporating the utility function, by Definition 1.7, $CVaR$ is thus defined as

$$CVaR_\alpha(S) = \mathbb{E}_y[f(S,y)|f(S,y) \leq VaR_\alpha(S)]$$

which is equivalent to the auxiliary function $H(S,\tau)$.

Part 1 of the thesis aims to maximize its objective function. Thus the optimization problem as stated in [4] is thus formally defined as

$$\max_{S\in I,\tau\in[0,\Gamma]} \tau - \frac{1}{\alpha}\mathbb{E}_y[(\tau - f(S,y))^+]$$

where Γ is the upper bound on the value of τ.

In the case of Part 2 of the thesis where the utility function is cost based, the expected utility and thus $CVaR$ needs to be minimized. The optimization problem then is formally defined as

$$\min_{S\in I,\tau\in[0,\Gamma]} \tau - \frac{1}{\alpha}\mathbb{E}_y[(\tau - f(S,y))^+]$$

where Γ is the upper bound on the value of τ.

3. METHODOLOGY

The following section describes the algorithms, methods and approaches to solve the objectives as defined in Section 1 using their mathematical formulations as described in Section 2. Hereon, both parts of the Thesis are discussed separately, unlike the simultaneous comparative discussion thus far.

3.1 Part 1 - Research and Exploration

3.1.1 Re-implementation of RAGA

Introduction: This section begins by re-implementing the Risk-aware Greedy Algorithm (RAGA) as stated in [4] under calculated assumptions in an attempt to understand and recreate the results of the paper. This re-implementation also provides the base for building algorithms for consequent generalizations of this algorithm. Here, a single agent unconstrained by distance or time tries to optimize its objectives of collecting rewards along its travel path (TSP) while minimizing the cost of travel under the risk of failure while gathering rewards. This algorithm attempts to provide a tour for this agent balancing its objectives as desired while also accounting for risk.

Original Algorithm [4]: The risk-aware greedy algorithm as stated in Algorithm 2 in the Appendix begins by initializing the decision set S (Hamiltonian Cycle) as an empty set, a degree vector D to store degree vertices in order and variables $\hat{H}_{max} = 0$ and $\hat{H}_{curr} = 0$ to store the maximal and current values of $\hat{H}(S, \tau)$. Thus for every possible τ value, the algorithm iterates for the edges $e \in E \backslash S$ sequentially and adds that edge that maximizes the marginal gain $\hat{H}(S \cup \{e\}) - \hat{H}(S)$ (measured approximately using a sampling-

based oracle function [22]) to S if it forms a valid tour element. The validity of the edge is maintained by ensuring that D has no array element greater than two and running a Depth First Search (DFS) on the current partial tour doesn't form a sub-tour as long as the length of the current tour is less than the total number of vertices in V. The value of the auxiliary function $\hat{H}$ for each tour is thus stored in $\hat{H}_{curr}$ and the one with the highest value $\hat{H}_{max}$ of $\hat{H}_{curr}$ is chosen.

Run-time and Analysis: By [4, Theorem 2], RAGA has a polynomial running time of

$$\mathcal{O}(\lceil\frac{\Gamma}{\gamma}\rceil(|V^3|(|V|n_p + n_s + 2\log|V| + 1)))$$

According to the analysis of the algorithm, it shows that for given values of τ, varying α and β values gives corresponding approximate greedy TSP routes.

- Lower value of α indicates a small risk level. Thus a low risk, low utility path is chosen. This results in a smaller mean value of information gain but also a lesser variance indicating some amount of certainty.
- Higher value of α indicates a high risk level. Thus a high risk, high utility path is chosen. This results in a higher mean value of information gain but also a higher variance indicating a higher amount of uncertainty.
- When β is close to 0, the tour the robot takes will focus on maximizing rewards, whereas when it is close to 1, the tour the robot takes focuses on maximizing cost. In both these cases, a path with high risk and high utility is chosen.
- As β is increased from 0, RAGA chooses paths with lesser risk and lesser utility with the most conservative paths around its average as it now chooses to optimize both simultaneously.

Assumptions for Re-implementation: Assuming that nodes (like real-time surveillance sites in Figure 1.4) that the agent travels through would have sparse placements (far away from each other and avoiding small angles between adjacent edges), any overlaps between tour edges are considered to be negligible. The information collected across every edge is considered to be independent, thus maintaining submodularity. It is also assumed that the information data, even though dynamic, changes very slightly during a tour computation's time complexity.

Re-implementation Approach: Based on the above assumptions, edge data was computed for every edge using the oracle function [22] for auxiliary function sampling approximation similar to RAGA. This edge data were then arranged in decreasing order of magnitude of its corresponding value, thus completing the data generation process.

For the run-time analysis, these edges were added greedily to the tour of the risk-aware TSP while maintaining the degree vector D and checking validity using DFS by eliminating edges that would result in the formation of sub-tours.

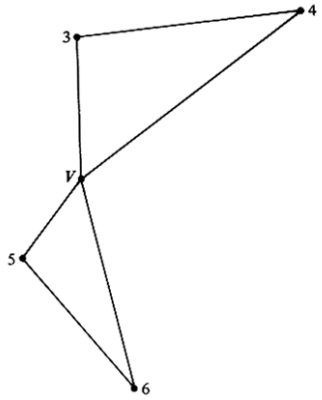
3.1.2 Multi-agent risk-aware greedy algorithm (mRAGA)

Introduction: Multiple agents have an advantage over single agents in that they can complete the allocated tasks in less time. If the robot agent has some constraint like fuel, it would have to revisit the depot for refuelling. A multi-agent approach would solve this problem as each agent would have to travel a shorter path without halt for refuelling. Multi-agent TSP is by definition a single origin-destination problem and can easily be converted into a classical TSP problem [6, Section 6.4.11]. For simplicity, two agents are considered here. The implementation can be easily generalised to n agents.

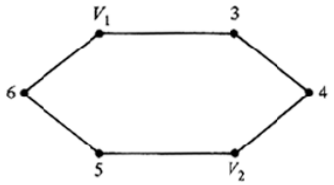
Assumptions: The assumptions used for the algorithm are considered to

be similar to the re-implementation of RAGA in Section 3.1.1.

Approach: The approach used in this algorithm is to convert the multi-route problem into a TSP-like problem that can be then solved by the RAGA approach used in section 3.1.1. This is incorporated using the m-travelling salesmen problem (m-TSP) proposed in [6]. The m-TSP has a single origin-destination V as seen in Figure 3.1 (a). The problem for two agents covering all sites simultaneously can be converted to a single agent travelling across the two independent Hamiltonian paths sequentially to complete its tour. This is achieved by duplicating the origin and destination into two vertices V_1 and V_2 as seen in Figure 3.1 (b). Thus instead of a single agent sharing paths across a linear time reference, two agents can share time while having independent tours to cover all map sites.



(a) Two Agent Path



(b) Conversion to a Single Agent Path

Fig. 3.1: Multi-agent risk-aware greedy algorithm (mRAGA) Example [6, Section 6.4.11]

Consider m agents that visits n sites with all tours beginning and ending at a common origin V. The m-TSP can thus be reduced to a classical TSP by replacing the origin with m instances of it, each corresponding to an individual agent. The nodes thus include a set of additional m vertices $V_1, V_2, ...V_m$, each connected to the other n nodes exactly in distance and positions as the original origin. Thus if x is any one of the n points to be

visited, then

$$d(V_1, x) = d(V_2, x) = ... = d(V_m, x) = d(V, x)$$

In addition to this, each of the m copies of V are connected to each other with "infinite" length (much greater than other distances) in order to avoid agents treading onto each others paths

$$d(V_i, V_j) = \infty \; \forall \, i, j = 1, 2, ..., m$$

The problem is thus converted into a (m+n)-point classical TSP. A minimum tour will never use a link connecting two copies of the origin. "V_i followed by V_j" will never appear (i, j = 1, 2, . . . , m). Upon finding a solution to this TSP using RAGA 3.1.1, the single-agent tour can be thus folded back along the m copies of V to merge it back into a single origin node. The single-agent tour thus decomposes into m tours as required by the mRAGA. Shown below is an example from [6, Section 6.4.11]

3.1.3 Generalizing the RAGA to incorporate a sensing radius

Introduction: This variant of RAGA incorporates a sensing radius R for detecting information as a reward for an agent. Thus upon travelling through adjacent edges, it is ensured that information collected during an incoming path is not counted twice during an outgoing path through a given node.

Assumptions: The assumption for sparse site placement is relaxed in this case, thus allowing for close overlaps or small angles between adjacent edges. The rest of the premises are retained as in the previous cases. It is still maintained that non-adjacent edge overlaps hardly have mutual information along the intersection region than the mutual information between adjacent edges with a small angle between them.

Approach: Following the basis of computation for edge data used in RAGA, it is seen that the distribution of information along an edge is a linear sum

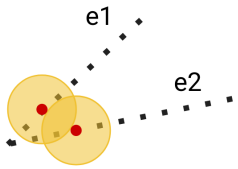


Fig. 3.2: Information Gain Overlaps due to Sensing Radius

of the data as experienced by the points that are obtained by rasterizing the edge. Thus the information of an edge is represented as a function of the information distributed along with its rasterized points. The edge data is converted into points data corresponding to the edge. Given its sensing radius R, the points that the agent "sees", given its sensing radius R, are eliminated from the point list common to other edges from their edge data.

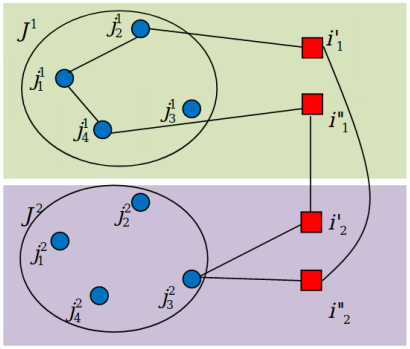
3.1.4 Multi-depot risk-aware greedy algorithm (MDRAGA)

Introduction: Like multiple agents, multiple depots also possess similar advantages for tour planning. Instead of agents returning to the initial origin depot for tasks like refuelling, they could refuel at multiple depots along its path, thus countering the need for multiple agents. This variant of RAGA implements a risk-aware greedy algorithm for a multi-depot TSP considering a single agent and then extends it to Multi-depot risk-aware greedy algorithm (MDRAGA).

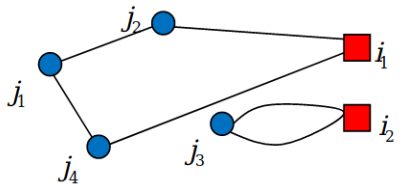
Assumption: Here, the assumptions are reverted back to that in section 3.1.1.

Approach: This problem is approached by a modification of the method of polynomial transformation of MDSMTSP to TSP with GTSP as the intermediate based on [7]. Initially, a more straightforward case for two depots is considered. The method can, however be generalised to the n depots case

easily.



(a) GTSP



(b) MDMTSP

Fig. 3.3: Multi-depot risk-aware greedy algorithm (MDRAGA) Example [7]

Multi-Depot Multiple Traveling Salesman Problem (MDMTSP) is a generalization of the TSP, which consists of determining a set of routes for the salesmen that jointly visit a set of given nodes (like surveillance sites), such that each agent starts from and returns to one depot among a set of available depots and the total cost of the routes is minimized. Let the set of sites be denoted by J and the set of potential depots by I. Thus, $V = I \cup J$, and $E = \{(i, j) : i \in V, j \in J\}$. The cost of any edge $(i, j) \in E$ is denoted by c_{ij}. Costs are assumed to be symmetric, and routes visiting only one site, called return trips, are allowed. Thus the aim is to reduce the MDMTSP to a generalized TSP (GTSP). Given a complete and weighted graph with vertex set V and a partition of V in $m \geq 3$ subsets $C_1, ..., C_m$, called clusters, the GTSP consists of finding a minimum cost cycle that visits each cluster exactly once.

The depot points are duplicated to solve the GTSP, thus preserving the previous edges and forming new ones for the new depots. Let the two depot sites be i_1 and i_2. The duplication thus leaves us with depot points i'_1, i'_2, i''_1, i''_2. The edge $\overline{i''_1 i'_2}$ and edge $\overline{i''_2 i'_1}$ are considered to have zero cost (or very high H value in our case). Whereas edges $\overline{i''_1 i'_1}$ and $\overline{i''_2 i'_2}$ are considered to have infinite cost (or zero H value in our case). The duplication of edges that join a site and a depot maintain their parameters. The duplication can be visualized as two sets of sites for every pair of the depot from which the clusters of GTSP

form (Figure 3.3 (a)). The depot duplication are then collapsed to give a consequent solution of the MDMTSP (Figure 3.3 (b)). The edge parameter can thus be changed to incorporate the auxiliary H values following the separate data generation process of section 3.1.1. The problem thus reduces to MDMTSP [7], thus solving the MDMRAGA.

3.2 Part 2 - Real-world Implementation

3.2.1 Data Cleaning and Analysis

This section aims to model the real-world problem case study as described in 1 using the data as described in section 1.3.3. Real-world data is not accurate and consists of human and calibration errors. Though the performance and usability of the model depend on the accuracy of the data, these can be enhanced by analyzing and cleaning the data to minimize its errors. The section begins by examining and cleaning the GTFS Static and Realtime Feeds independently and then merging them as described in section 1.3.3. This data is then used to model a graph network for implementing the risk-aware optimization problem.

GTFS Static Feed

Initial error identification for Static Data was performed using the gtfs_kit library [23]. The following errors as shown in figure 3.4 were identified:

	type	message	table	rows
0	warning	Unrecognized column agency_id	fare_attributes	[]
1	warning	Route has no trips	routes	[29, 43, 76, 84, 95, 133, 146, 160, 205, 206, ...
3	warning	Repeated pair (trip_id, departure_time)	stop_times	[2153146, 2153192, 2153238, 2153284, 2153330, ...
2	warning	Stop has no stop times	stops	[1713, 1714, 2408, 2409, 3056, 3057, 3058, 305...

Fig. 3.4: Error Identification for Static Feed

- **Unrecognized column agency_id** - agency_id describes the bus agency responsible for the transit services. Here, it is common for all – DIMTS (Delhi Integrated Multi-Modal Transit System). Thus this field is ignored.
- **Route has no trips** - 83/1270 routes don't have any trips associated with it. As trips are a subset of routes, and as the data being used doesn't focus on the transit system but rather the network formed by it, this error too can be ignored.

- **Repeated pair (trip_id, departure_time)** - For a given trip, each stop is identified by a specific departure time whose magnitude corresponds with the sequence of bus stops that trip comprises of. For the corresponding GTFS Static Data, arrival and departure times for a given stop are the same (no halt time considered during time data approximation by OTD). Hence, only departure times are considered. Two stops thus cannot have the exact departure time. However, this is contradicted for 1893/2250290 (stop, departure time) pairs. This error is eliminated by proportionately averaging the time for the stop whose departure time is repeated by its previous and next stops' departure times.

- **Stop has no stop times** - 34/4192 stops do not posses stop_time data. This error cannot be corrected, and the corresponding stops need to be dropped.

GTFS Realtime Feed

Error analysis for this section involved manual methods and strategies developed from scratch for a generalised GTFS Realtime Feed case scenario displayed in a database format as shown in figure 3.5.

	vehicle_id	lat	lng	speed	route_id	trip_id	timestamp	date	time
0	DL1PD5111	28.5534191131592	77.0547790527344	0.0	484.0	484_19_10	1638383337	2021-12-01	00:00:09
1	DL1PD5067	28.5687808990479	76.975830078125	0.0	276.0	276_20_40	1638383339	2021-12-01	00:00:09
2	DL1PD4578	28.7004280090332	77.0348663330078	0.0	749.0	749_19_00	1638383389	2021-12-01	00:00:09
3	DL1PD4652	28.7004108428955	77.0346832275391	0.0	50.0	50_19_10	1638383383	2021-12-01	00:00:09
4	DL1PD5114	28.7004375457764	77.0344619750977	0.0	1232.0	1232_19_00	1638383394	2021-12-01	00:00:09
5	DL1PD4376	28.7004261016846	77.0345077514648	0.0	830.0	830_18_20	1638383390	2021-12-01	00:00:09
6	DL1PD5050	28.7004203796387	77.0347442626953	0.0	881.0	881_19_30	1638383397	2021-12-01	00:00:09
7	DL1PD2289	28.6838817596436	77.330810546875	0.0	371.0	371_18_50	1638383390	2021-12-01	00:00:09
8	DL1PD5343	28.5534152984619	77.0531692504883	0.0	873.0	873_20_40	1638383398	2021-12-01	00:00:09
9	DL1PD4925	28.7005386352539	77.0353164672852	0.0	557.0	557_21_20	1638383394	2021-12-01	00:00:09

Fig. 3.5: Realtime Data Feed Format

- **Checking GPS-Ping Region Bound Constraints** - This error covers the pings that lay outside the region's bounds under consideration

(here, Delhi) for the bus transit system. These pings and their information were dropped (0.05% of 1,49,98,700 pings). An example of the error is shown for day 1 in Figure 3.6 where pings lie in Delhi, but a few also lie near Africa.

Fig. 3.6: GPS-Ping Region Bound Error

- **Eliminating Repeated Pings** - GPS pings for some trips were found to have repeated sub-sequence of pings. Figure 3.7 shows such an instance where repetitions occurred in latitude (lat), longitude (lng) and time (timestamp and time) entries for a given trip.

```
df = get_data_for('601_20_50')
df
```

✓ 1.6s

	vehicle_id	lat	lng	speed	route_id	trip_id	timestamp	date	time
0	DL1PC5847	28.5915508270264	77.2366943359375	0.0	601.0	601_20_50	1638296991	2021-11-30	00:00:06
1	DL1PC5847	28.5915508270264	77.2366943359375	0.0	601.0	601_20_50	1638296991	2021-11-30	00:00:17
2	DL1PC5847	28.5915679931641	77.2367172241211	0.0	601.0	601_20_50	1638297011	2021-11-30	00:00:28
3	DL1PC5847	28.5915546417236	77.2367248535156	0.0	601.0	601_20_50	1638297021	2021-11-30	00:00:39
4	DL1PC5847	28.5915203094482	77.2367324829102	0.0	601.0	601_20_50	1638297031	2021-11-30	00:00:50
...	...	...	...	...	...	...	...	...	...
984	DL1PC5847	28.5916919708252	77.2367706298828	0.0	601.0	601_20_50	1638307786	2021-11-30	03:00:21
985	DL1PC5847	28.5916919708252	77.2367706298828	0.0	601.0	601_20_50	1638307786	2021-11-30	03:00:32
986	DL1PC5847	28.5916919708252	77.2367706298828	0.0	601.0	601_20_50	1638307786	2021-11-30	03:00:43
987	DL1PC5847	28.5916919708252	77.2367706298828	0.0	601.0	601_20_50	1638307786	2021-11-30	03:00:54
988	DL1PC5847	28.5916919708252	77.2367706298828	0.0	601.0	601_20_50	1638307786	2021-11-30	03:01:05

Fig. 3.7: GPS-Ping Repetition Error in Feed Data

This error was frequent for even a single trip over several consecutive pings. This can be visualized via Figure 3.8 where the X-axis shows timestamps and Y-axis shows its frequency in sequence.

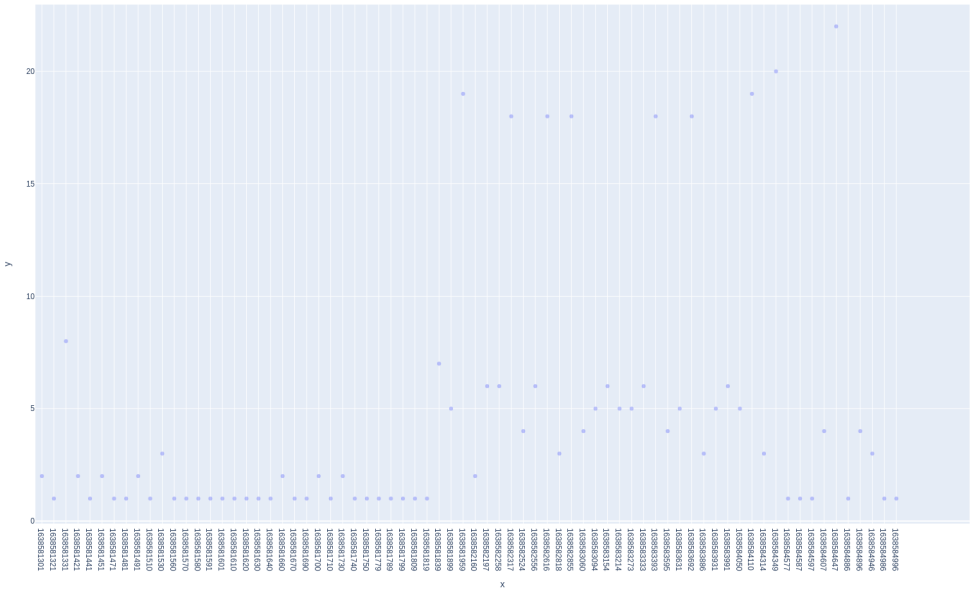


Fig. 3.8: GPS-Ping Repetition Error Visual Analysis

Elimination of these faulty repeated pings left the trip with missing sections of GPS pings in its sequence. This error could either arise due to date-independent causes (lack of network connectivity in a particular region resulting in the unavailability) or date-dependent reasons (junk ping data due to faulty equipment, bus breakdowns, etc.). Repetitions due to the prior were unavoidable. However, the latter could be rectified by interpolating lost data based on the data for the same trip but a different date. The error in this domain of incomplete information was significant (38% of 1,49,23,253 pings).

- **Eliminating Clustered Trips** - The spread of a trip as measured by bus stops (Static Feed Data) was compared with that measured by GPS pings (Realtime Feed Data). Trips whose spreads deviated significantly beyond a determined threshold were eliminated for that

day, Eliminating clustered trips as faulty (0.06% of 92,49,946). As seen in Figure 3.9, the leftmost blue vertical line identifies over 4000 clustered trips.

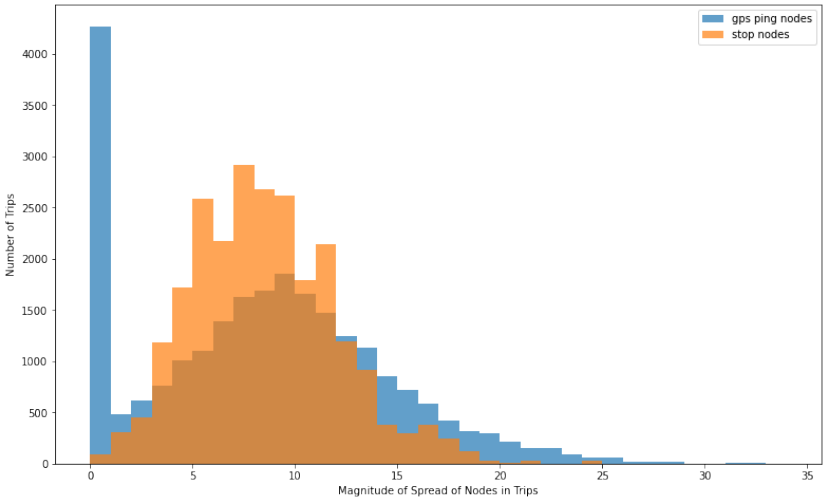


Fig. 3.9: GPS-Ping Clusters Error

- **Eliminating Un-referenced Trips** - Here GPS pings of trips in Real-time Feed Data that did not have their corresponding bus stop data in Static Feed Data and vice-versa were eliminated (0.13% of 86,89,213).

3.2.2 Cost Objective Data Generation

The multi-cost objectives for the optimization problem as defined in section 1.3.2 are as follows -

Congestion Cost: The congestion data along possible bus trips are dynamic across days (here for December 2021) and across different times for a given day. It is computed based on the assumption that changes in congestion are insignificant over a half an hour bin interval. However, this 30 min bin can be changed based on the desired accuracy. Congestion costs are computed per adjacent bus-stop pair for a given bin interval. Congestion between adjacent stop pairs for any trip is calculated by considering the distribution

of the times buses take to traverse the given stop pair across multiple trips on all 31 days during the bin time interval. This time is computed using the time difference between the GPS pings closest to each stop pair considered, identified using the BallTree Algorithm. The distribution is represented as noise y_{cong}

The congestion cost of an edge is thus defined as

$$f_{cong}(S, y) = c(S, y_c)$$

where the function c reduces distribution noise (y_c by converting it into a truncated Gaussian distribution similar to the reward function of section 3.1.

Fare Cost: The fare cost is deterministic and is directly considered by incorporating fare data between stops of the Static Feed Data. An approximation of the fare data can also be made by considering distances between stops. The fare cost is denoted by $f_{fare}(S)$.

Thus the utility function as described in section 2.1 here becomes

$$f(S, y) = \beta \cdot f_{cong}(S, y) + (1 - \beta) \cdot f_{fare}(S)$$

where β is the balancing parameter. Both f_{cong} and f_{fare} are normalised to have comparable means. By [4, Lemma 3], considering f_{cong} to be an analogue of f_r, the utility function $f(S, y)$ is both submodular and monotone increasing in S. Also according to the objective function defined above, a time-constrained risk-aware dynamic A-star search algorithm is defined to minimize $CVaR_\alpha(S)$ which is equivalent to minimizing the auxiliary function $H(S, \tau)$. The optimization part as stated in section 2.3 is thus defined as

$$\min_{S \in I, \tau \in [0, \Gamma]} \tau - \frac{1}{\alpha} \mathbb{E}_y[(\tau - f(S, y))^+]$$

3.2.3 Time-constrained risk-aware dynamic A-star search algorithm

Consider a graph $G'(V', E')$ where V' are all possible bus-stop nodes and E' are the edges between the nodes, given that the end-nodes of an edge $e \in E'$ exist adjacently in a stops sequence for any given trip in the Static Data Feed. According to section 3.2.2, each edge has two cost metrics to optimize - stochastic congestion cost and a deterministic fare cost.

Consider another Graph $G(V, E, t) \subseteq G'(V', E')$ where V are the number of active bus-stop nodes and E are the number of active edges between these nodes at a bin interval t. A bus-stop node is considered active if the departure time associated with the bus-stop node falls within the time interval t. An edge is considered active if the end nodes of the edge are active in t.

A further complexity layer is added due to the involvement of time. Due to this, a "halt patience (hp) is introduced, which is defined as the amount of time an agent in transit is willing to wait at a particular bus stop. hp thus represents the temporal neighbours of a bus-stop node. The temporal neighbours of a node n are therefore defined as a set of active adjacent nodes N that can be reached from n given that the difference between the departure time of n and the departure time of a bus that traverses $n' \in N$ is lesser than or equal to hp.

For the A^* algorithm, a f_{score} value is determined which is the sum of the cost function g_{score} and a heuristic h for given nodes n.

$$f(n) = g(n) + h(n)$$

The problem thus assigns the auxiliary optimization function to g_{score}. The experiment was tried both - with a small distance-based heuristic and without (Dijkstra's Algorithm). Thus a further modification from the traditional A-star Search includes updating the graph G with every iteration of the while loop while maintaining an additional dictionary of features - came from a stop and came from time to support the temporal trace of neighbour-nodes defined above.

The pseudocode that describes the algorithm is as below. Note here that

the stochastic costs of edges are pre-generated by binning. Edge information is stored in the form (start node, end node, start time, end time, costs). Thus the cost of the edge between the start and end node is defined by the bin that corresponds to the beginning and end time. Data generated this way can thus be directly implemented into the A-star search as the neighbours of a node considered at time t, represented above, incorporate activation of nodes at t.

A function reconstructPath() is also considered in the below algorithm to construct the A-star search path from the totalSet variable as described below. totalSet contains information in the form

$$ts = (startnode, endnode, starttime, endtime, costs)$$

The ts in openSet that has the end node as the final node is considered. Its start node and start time are noted. The ts for which this start node is the end node and the start time falls within hp of the end time of that ts is considered next. If there are multiple such entries, the one with the lowest $costs$ is considered. A path is thus generated for a given τ value. The function thus outputs the path with the lowest sum of $costs$ for its edges and returns the corresponding τ value.

Algorithm 1: Time-constrained Risk-aware Dynamic A-star Search

Input: Graph $G'(V', E')$; Risk level $\alpha \in (0, 1]$; Weighing factor $\beta \in [0, 1]$; Start node (s_0); Start time $t(s_0) = t_0$; End node s_f; Upper bound $\Gamma \in \mathbb{R}^+$ on τ; Searching factor $\gamma \in (0, \Gamma]$

Output: A path tour S^G and its respective τ^G.

for $i = \{0,1,2,...,\left\lceil \frac{\Gamma}{\gamma} \right\rceil\}$ **do**

- $\tau_i = i\gamma$
- $openSet(edge)$ = node stop-pair (s_0, s_1) s.t. $t(s_0)$ is closest to t_0
- $openSet(prev\_stop)$ = NAN
- $openSet(prev\_time)$ = NAN
- $openSet(\hat{H}) = f(e_0, y)$
- $openSet(g_{score}) = 0$
- $openSet(f_{score})$ = h(n)
- **while** *openSet is not empty* **do**
  - $currentSet$ = edge (s_i, s_{i+1}) in openSet having lowest f_{score} value
  - **if** *current edge node* $= s_f$ **then**
    - return reconstructPath(totalSet, prev_stop, prev_time)
  - **end**
  - iterSet = temporal neighbours of currentSet within hp
  - $iterSet(prev\_stop) = s_{i+1}$
  - $iterSet(prev\_time) = t(s_{i+1})$
  - $iterSet(\hat{H}) = f((s_i, s_{i+1}), y)$
  - $iterSet(g_{score}) = \hat{H} + currentSet(g_{score})$
  - $iterSet(f_{score}) = iterSet(g_{score}) + h(n)$
  - add iterSet to opeSet and remove currentSet from openSet
  - add iterSet to totalSet
- **end**

end

3.2.4 Time-constrained risk-aware dynamic TSP

Paths generated between two chosen stops in section 3.2.3 are considered as edges of a 2-node TSP set on the graph network of the real-world data. For an n-node TSP, 2^n edges' data for the n chosen depots for touring are obtained using the time-constrained risk-aware dynamic A-star search algorithm. These edges are then arranged in ascending order of their collective costs to greedily choose TSP-edges for the tour path formation similar to RAGA in section 3.1.1.

4. RESULTS

This section discusses the run-time analysis and result analysis of the methodologies proposed in section 3 in order. Each explanation is supported by a visual output generated by the methodologies' successful implementations.

4.1 Part 1 - Research and Exploration

4.1.1 Re-implementation of RAGA

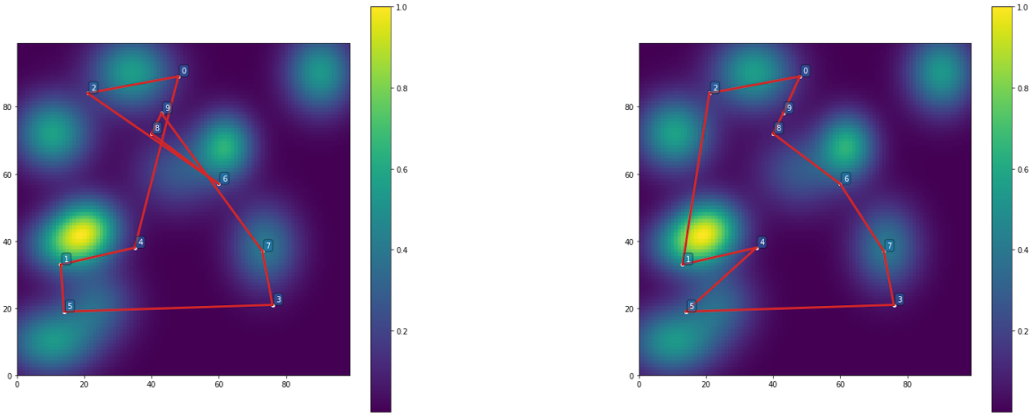
Runtime

The modified algorithm also runs for every value of τ thus having an $\mathcal{O}(\lceil\frac{\Gamma}{\gamma}\rceil)$ outer loop. Further, the algorithm also iterates for every edge thus sustaining the factor of $\mathcal{O}(|E|)$. The DFS for every edge results in a factor of $\mathcal{O}(|E||V|)$. Thus the resultant run-time now reduces to a factor of $\mathcal{O}(\lceil\frac{\Gamma}{\gamma}\rceil|V|^3)$ using the fact that $\mathcal{O}(|E|) = \mathcal{O}(|V|^2)$. The remaining factor of the RAGA run-time is compensated during the data generation process.

Analysis

It was found that the results obtained upon the optimised re-implementation were very similar to the original thus asserting the assumptions made for the algorithm.

In the two subplots in Figure 4.1, it is seen that for (a), $\alpha = 0.1$, a low-risk value, a risk-neutral tour is chosen wherein it tries to avoid short risky paths that procure maximum information gain at risk. At the same time $\beta = 0.5$. Thus that path tries to balance costs and rewards, thus trying to maintain an optimal tour by TSP but choosing a longer route length due to a low-risk



(a) $\alpha = 0.1$, $\beta = 0.5$ **(b)** $\alpha = 0.9$, $\beta = 1$

Fig. 4.1: TSP routes obtained from RAGA Re-implementation

factor. In contrast to this, for (b), $\alpha = 0.9$, a high risk value and $\beta = 1$, which seeks to optimise cost only. Thus, a shorter path is chosen, wanting to minimise cost and travel along paths of more information gain due to a high-risk factor. Both these results and for the remaining cases are observed as expected.

4.1.2 Multi-agent risk-aware greedy algorithm (mRAGA)

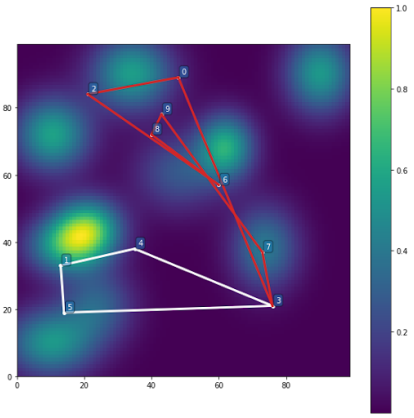
Runtime

The run-time of this algorithms also remains unchanged from that of the previous case. The only difference here is that number of vertices $|V|$ changes to $|V|+m$ and number of edges $|E|$ thus changes to $(|V|+m)^2$. As these would result in smaller or constant factors in the run-time product, the run-time of $\mathcal{O}(\lceil\frac{\Gamma}{\gamma}\rceil|V|^3)$ remains unchanged.

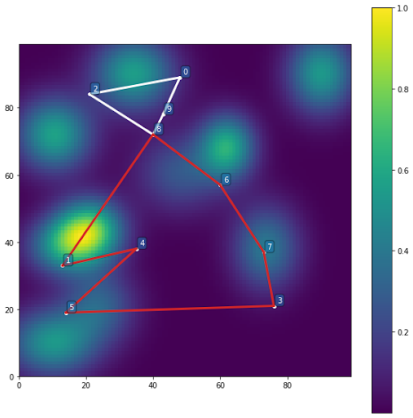
Analysis

The parameters for the paths are maintained to be similar to the previous case to visualise comparisons between similar cases in single-agent RAGA and mRAGA. Thus figure (a) has $\alpha = 0.1$ (risk-averse), $\beta = 0.5$ (balancing

costs and rewards), depot = point 3 and figure (b) has $\alpha = 0.9$ (risky), $\beta = 1$ (cost only), depot = point 8. A similar trend to the previous case while improving on the information gained due to two agents in a similar run-time and risk metric is thus seen. The depot points are chosen randomly as one of the edge's endpoints with the highest H value. Both these results and for the remaining cases are observed as expected.



(a) $\alpha = 0.1$ (risk-averse), $\beta = 0.5$ (balancing costs and rewards), depot = point 3



(b) $\alpha = 0.1$ (risk-averse), $\beta = 0.5$ (balancing costs and rewards), depot = point 3

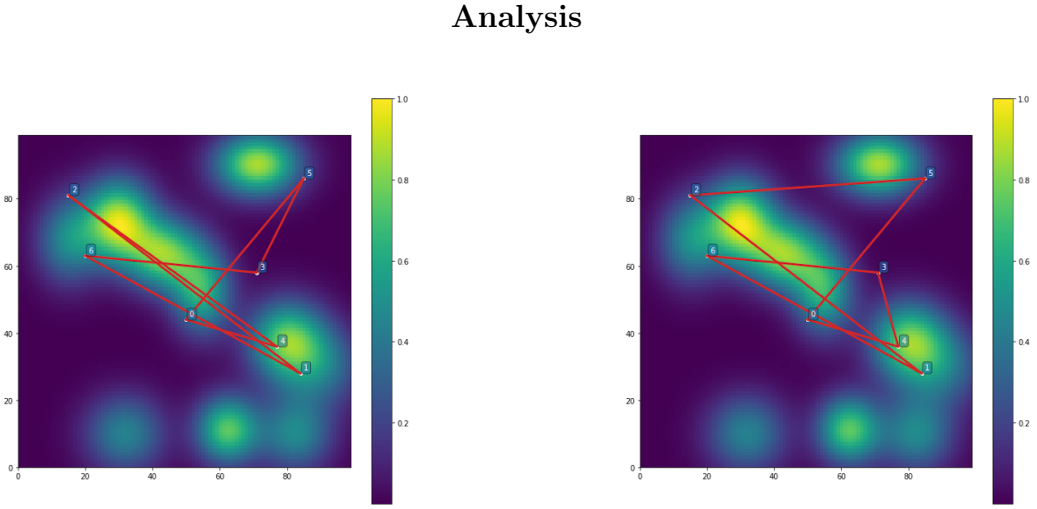
Fig. 4.2: 2-TSP Implementation Result

4.1.3 Generalizing the RAGA to incorporate a sensing radius

Runtime

Consider the run-time analysis of section 3.1.1 of $\mathcal{O}(\lceil \frac{\Gamma}{\gamma} \rceil |V|^3)$. The run-time of the current algorithm will be built upon this as the base. On the addition of every edge, the points on the edge are considered and compared with those of possible adjacent edges. Suppose n_p is the number of raster points of an edge. In that case, the complexity for this particular case increases by a factor of $2|V|n_p^2$ as each point of the current edge is compared to each point of n adjacent edges along each endpoint of the current edge. Thus the new complexity becomes $\mathcal{O}(\lceil \frac{\Gamma}{\gamma} \rceil |V|^4 n_p)$. Note, however, that the actual

complexity will be much lower than this as the sensing radius in a real-life scenario would be very small compared to the distances between the sites and thus having to check for mutual information only around site points. Secondly, as a point attains a value of 2 in its vertex degree array D, no further edges having that point as an endpoint are considered. Thus, as the tour progresses, the adjacent edges of an edge reduce drastically by a factor of n^2.



(a) Path Incorporating Sensing Radius

(b) Path Without Incorporating Sensing Radius

Fig. 4.3: Effects of Incorporating a Sensing Radius

As expected, it is seen that for a given parameter set, the number of overlapping edges or edges with small angles is reduced while keeping certain edges in the route without sensing radius constant. Also, the H value in figure (b) (with sensing radius) reduces as compared to figure (a) (without sensing radius) as overlapping information is not counted twice. We consider the sensing radius $R = 0.05 * mapsize$

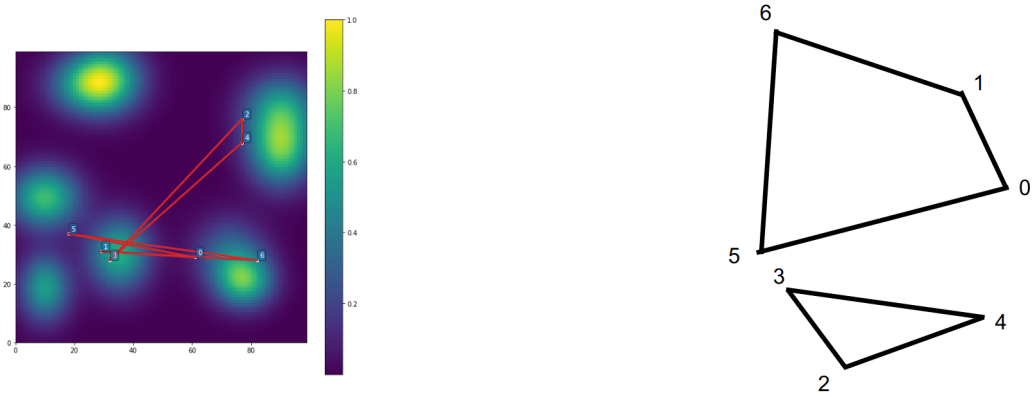
4.1.4 Multi-depot risk-aware greedy algorithm (MDRAGA)

Runtime

The run-time of this algorithms also remains unchanged from that of the case in section 4. Similar to section 5, the only difference here would be that number of vertices $|V|$ changes to $|V|+2$ and number of edges $|E|$ thus changes to $(|V|+2)^2$. As these would result in smaller or constant factors in the run-time product, the run-time of $\mathcal{O}(\lceil\frac{\Gamma}{\gamma}\rceil|V|^3)$ remains unchanged.

Analysis

The MDMRAGA works as expected. Here points 0 and 4 are considered as the depots as they had the least edge priority when arranged in decreasing order of H values, thus being the least likely to get picked. Figure (a) below shows the graph representation of the solution, whereas figure(b) shows a simplified view of the same, which is not to scale and purely for visualisation. The tour output is of the form $[[5,6],[0,5],[1,6],[0,1],[2,3],[3,4],[2,4]]$ which forms the two resultant sub-routes.



(a) Actual Tour

(b) Visualization

Fig. 4.4: Tour Generated by MDRAGA

This can be easily expanded to n depots with variable agents on every depot by finding the cluster loops per depot and applying mRAGA imple-

mented previously on each cluster. Further, incorporating sensing range by the algorithm defined previously would lead to the optimal selection of route edges in practical situations given k agents distributed across m depots.

4.2 Part 2 - Real-world Implementation

4.2.1 Time-constrained risk-aware dynamic A-star search algorithm

Runtime

Data Generation: As stated in section 3.2, alterations on Static (size = n) and Realtime (size = m) Feed data is made sequentially on each data-set thus giving a linear time complexity except in two cases. When both datasets are merged for stochastic cost computation, the complexity thus turns out to be $O(n \cdot m)$ for a constant number of days. Given the average degree centrality l for the nodes of the graph formed by the Static Data Feed, there are n^l stop pairs on average. For generation of stochastic cost, a BallTree Algorithm gives an order of $O(nl \log(nl))$ for the search and $O(nl(\log(nl))^2)$ for its construction. Thus the total time complexity for data generation thus is $O(n(m + l(\log(nl))^2))$

Algorithm: The run-time complexity of the A-star depends on the heuristics, thus giving it $O(b^d)$ in unbounded search space where b is the branching factor and d is the depth of the solution. For the time constrained-risk aware dynamic A-star search, this bound is further restricted based on hp, which defines the number of neighbour branches per node. Consider the average centrality of the nodes in the network given a $hp = 300 seconds$ to be l. Thus the time-complexity for the given case of part 2 is bounded to $O(b^l)$

Real-life networks such as the one used in bus transits are super sparse. For example, the bus transit system has one adjacent stop on the main road and three on 4-way crossings. The number of busses that pass through in a given hp (say 5mins) is probably overestimated around 5 (one bus arrives per minute). Still, the worst time complexity is bounded to $O(b^{10})$.

Analysis

Similar to RAGA on TSP 4.1, similar results are shown by the implementation of the time-constrained risk-aware dynamic A-star search algorithm for Delhi Bus Transit Systems 1.3.3.

The below two cases vary alpha while keeping beta constant -

- For small alpha (lower risk) and small beta (importance to minimizing congestion cost), the paths computed can be longer but avoid traffic as seen in Figure 4.5
- For small alpha (lower risk) and large beta (importance to conserving fare cost), the paths computed are shorter and can overlap with congestion as seen in Figure 4.6

The below two cases vary beta while keeping alpha constant in contrast to the above two cases -

- For large alpha (higher risk) and small beta (importance to minimizing congestion cost), the paths computed can be longer, but the risk factor motivates the agent to risk a shorter path as seen in Figure 4.7
- For large alpha (higher risk) and large beta (importance to conserving fare cost), the paths computed are all shorter and can risk traffic as seen in Figure 4.8

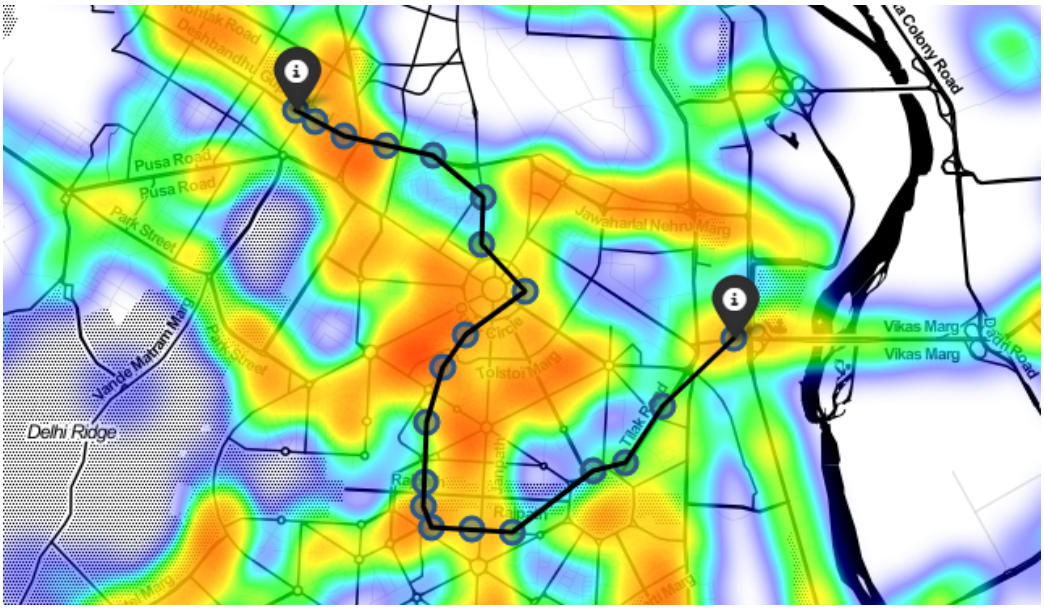


Fig. 4.5: Result1: small alpha ($\alpha = 0.1$), small beta ($\beta = 0.1$)

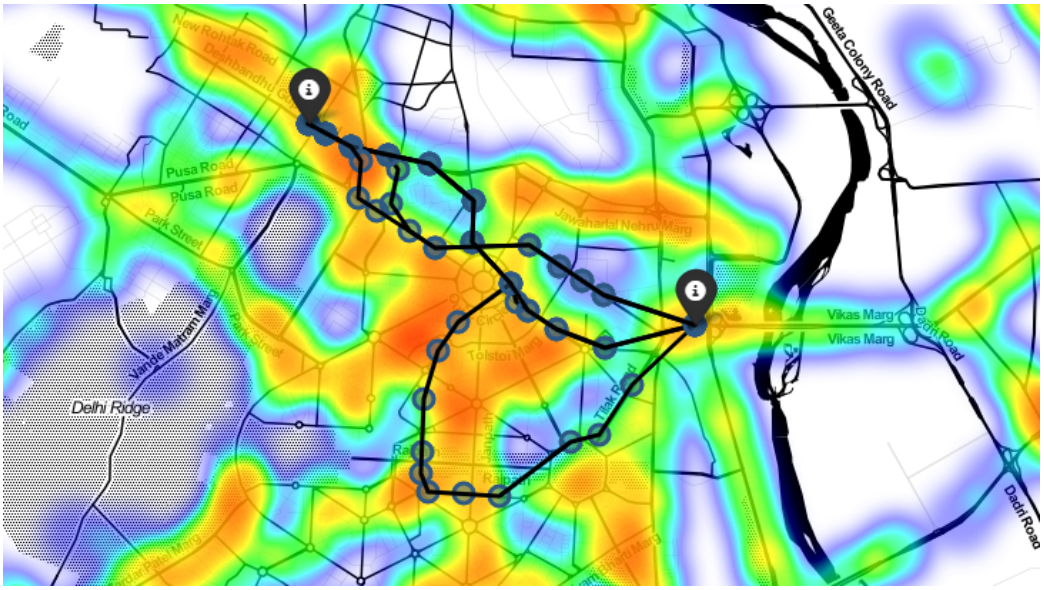


Fig. 4.6: Result2: small alpha ($\alpha = 0.1$), large beta ($\beta = 0.8$)

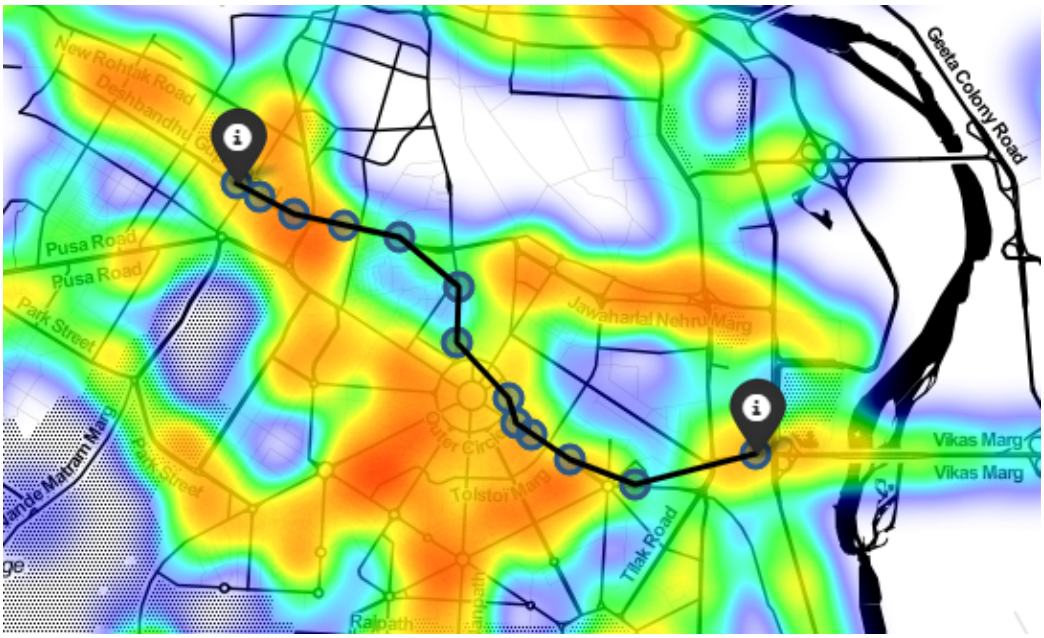


Fig. 4.7: Result3: large alpha ($\alpha = 0.8$), small beta ($\beta = 0.1$)

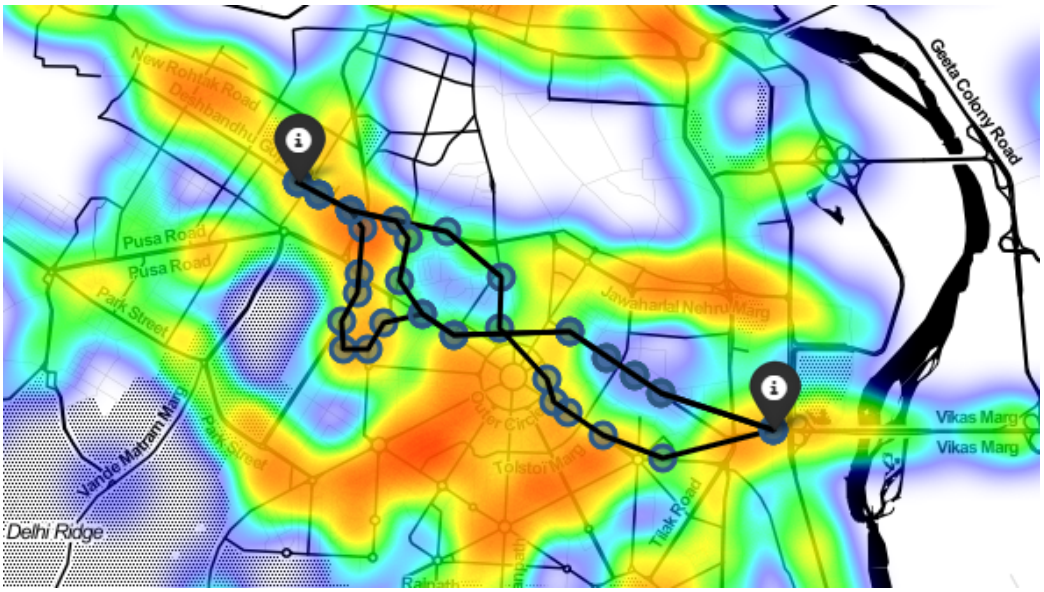


Fig. 4.8: Result4: large alpha ($\alpha = 0.8$), large beta ($\beta = 0.8$)

4.2.2 Time-constrained risk-aware dynamic TSP

Runtime

Similar to the re-implementation of RAGA in section 4.1, the time-constrained risk-aware dynamic TSP has a run-time of $\mathcal{O}(\lceil \frac{\Gamma}{\gamma} \rceil |V|^3)$ using the fact that $\mathcal{O}(|E|) = \mathcal{O}(|V|^2)$. The remaining factor of the RAGA run-time is compensated during the data generation process.

Analysis

Shown below are the results for a 5-depot risk-aware TSP given small alpha ($\alpha = 0.1$) and a balanced beta ($\beta = 0.5$) in Figure 4.9 and balanced alpha ($\alpha = 0.5$) and a large beta ($\beta = 0.8$) in Figure 4.10. The depots are indicated by the black markers and the tour is represented by dashed lines.

- For smaller alpha (lower risk) and a balanced beta (balance between congestion cost and fare cost), the paths computed are seen to be surrounding the region of travel thus avoiding central areas of congestion as seen in Figure 4.9

- For a lesser risk value alpha as compared to the previous case and large beta (importance to conserving fare cost), the paths computed are shorter and can risk traffic. Thus the tour moves through the central locations as seen in Figure 4.10

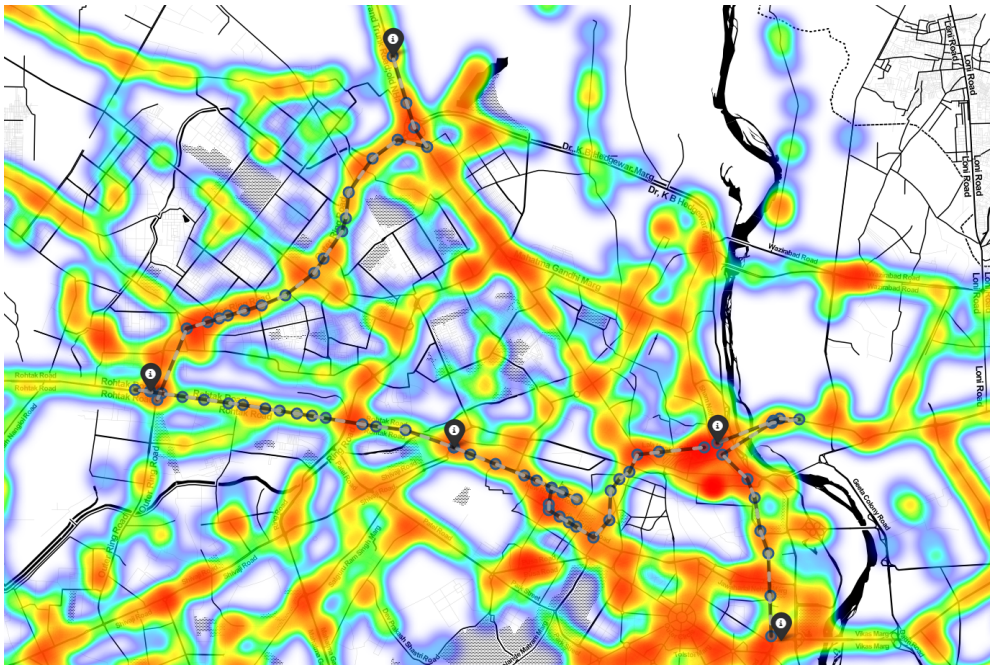


Fig. 4.9: Result1: small alpha ($\alpha = 0.1$), balanced beta ($\beta = 0.5$)

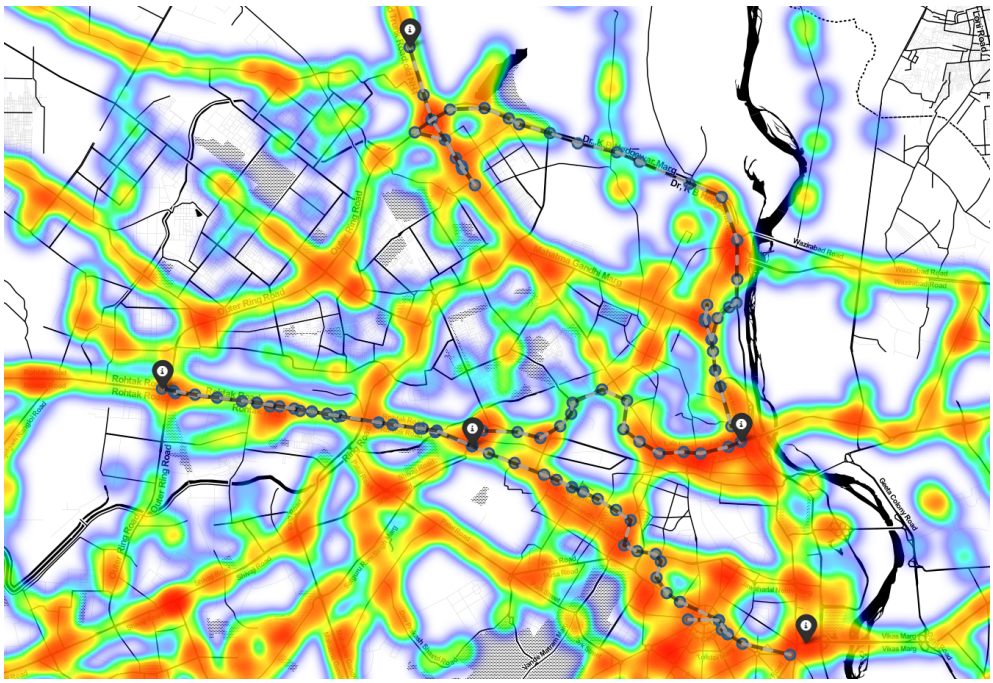


Fig. 4.10: Result2: neutral alpha ($\alpha = 0.5$), large beta ($\beta = 0.8$)

5. CONCLUSION

5.1 Discussions

The goal as described in section 1 is thus fulfilled to a promising extent. The incorporation of risk in combinatorial optimization problems for robotics is a recent field. Its future aims to make robots more life-like; however, this is just the beginning. This is both an optimistic as well as a cautionary statement. Its optimism lies in the fact that this base set-up could help with risk-aware robotic solutions for real-world implementations. But the caution lies in the fact that Open Data for real-world systems is not robust if not freely available. Several errors were observed upon data analysis as performed in section 3.2 which cannot be ignored completely. This, in turn, affected the accuracy of paths generated for traversal constrained by α and β values. This limitation also stunted further applications and possibilities. It, however, sets a basis for future scope in this field in the growing age of data by showcasing its potential.

5.2 Future Work

Future work for this thesis entails using Genetic Algorithms or Reinforcement Learning to better the run-time and accuracy of the current greedy algorithms under risk constraints. This work can be extended to generalize risk-aware algorithms to any combinatorial optimization problems, which form the basis of many real-life applications in operations research. Future work can also include a generalization of the cost function to incorporate more than two objectives on defined levels of stochasticity. Finally, further

work can also entail simulating the obtained solutions in real-life to integrate device and sensor errors, thus setting a base for the initialization of a risk-aware robotics domain.

Appendices

I Algorithms

Algorithm 2: Risk-Aware Greedy Algorithm (RAGA) [4]

Input: Graph $G(V, E)$; Risk level $\alpha \in (0.1]$; Weighing factor $\beta \in [0, 1]$; Upper bound $\Gamma \in \mathbb{R}^+$ on τ; Searching factor $\gamma \in (0, \Gamma]$; Oracle function $\mathbb{O}$ that approximates $H(S, \tau)$ as $\hat{H}(S, \tau)$.

Output: A Hamiltonian tour S^G and its respective τ^G.

for $i = \{0,1,2,...,\left\lceil \frac{\Gamma}{\gamma} \right\rceil\}$ **do**
- $S = \phi; D = 0_{1\times|V|}; \hat{H}_{max} = 0; \hat{H}_{curr} = 0$
- $\tau_i = i\gamma$
- **while** $\varepsilon \neq \phi$ *and* $|S| \leq |V|$ **do**
  - $e^* = argmax_{\forall e \in \varepsilon \setminus S} \hat{H}(S \cup \{e\}, \tau_i) - \hat{H}(S, \tau_i)$
  - $(u, v) \leftarrow V(e^*)$
  - flag = False
  - **if** $D[u] < 2$ *and* $D[v] < 2$ **then**
    - check subtour existence in $S \cup \{e\}$ by running DFS
    - **if** *no subtour present* **then**
      - flag = True
    - **end**
  - **end**
  - **if** *flag == True* **then**
    - $S \rightarrow S \cup \{e^*\}$
    - $D[u] += 1; D[v] += 1$
    - $\hat{H}_{curr} = \hat{H}(S \cup \{e^*\}, \tau_i)$
  - **end**
  - $\varepsilon \leftarrow \varepsilon \setminus \{e^*\}$
- **end**
- **if** $\hat{H}_{curr} \geq \hat{H}_{max}$ **then**
  - $\hat{H}_{max} = \hat{H}_{curr}; (S^G, \tau^G) = (S, \tau_i)$
- **end**
- **if** $\hat{H}_{curr} < 0$ **then**
  - **break**
- **end**

end

BIBLIOGRAPHY

[1] A. Krause and D. Golovin, "Submodular function maximization," *Tractability*, p. 71–104.

[2] "Ucb mathematics." [Online]. Available: https://math.berkeley.edu/

[3] L. Zhou and P. Tokekar, "Risk-aware submodular optimization for multirobot coordination," *IEEE Transactions on Robotics*, p. 1–21, 2022.

[4] R. Balasubramanian, L. Zhou, P. Tokekar, and P. B. Sujit, "Risk-aware submodular optimization for stochastic travelling salesperson problem," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[5] "General transit feed specification," Apr 2022. [Online]. Available: https://en.wikipedia.org/wiki/General_Transit_Feed_Specification

[6] R. C. Larson and A. R. Odoni, *Urban Operations Research.* Prentice-Hall, 1981.

[7] E. Benavent and A. Martínez, "Multi-depot multiple tsp: A polyhedral study and computational results," *Annals of Operations Research*, vol. 207, no. 1, p. 7–25, 2011.

[8] A. Schrijver, *Combinatorial optimization: Polyhedra and efficiency.* Springer, 2003.

[9] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, p. 85–103, 1972.

[10] A. Wood and J. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, p. 54–62, 2002.

[11] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *The International Journal of Robotics Research*, vol. 36, no. 13-14, p. 1540–1553, 2017.

[12] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," *Springer Proceedings in Advanced Robotics*, p. 75–84, 2019.

[13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, no. 1, p. 265–294, 1978.

[14] B. Sankaran, M. Ghazvininejad, X. He, D. Kale, and L. Cohen, "Learning and optimization with submodular functions," 2015.

[15] J. Morgan *et al.*, *Riskmetrics technical document*. New York RiskMetrics, 1996.

[16] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *The Journal of Risk*, vol. 2, 2000.

[17] G. C. Pflug, "Some remarks on the value-at-risk and the conditional value-at-risk," *Nonconvex Optimization and Its Applications*, p. 272–281, 2000.

[18] "Open transit data." [Online]. Available: https://opendata.iiitd.edu.in/

[19] "Gtfs static overview — google developers." [Online]. Available: https://developers.google.com/transit/gtfs

[20] "Gtfs realtime overview — google developers." [Online]. Available: https://developers.google.com/transit/gtfs-realtime

[21] K. Srivastava, "Applications of open transit data," *MS Thesis, IIIT-D-MTech-CS-Mobile Computing-MT18099*, 2020.

[22] N. Ohsaka and Y. Yoshida, "Portfolio optimization for influence spread," *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[23] "Gtfs kit 5.2.1." [Online]. Available: https://mrcagney.github.io/gtfs_kit_docs/