DYNAMIC GROUP TRIP PLANNING QUERIES IN SPATIAL DATABASES

FARHANA AKLAM
Bachelor of Science, Military Institute of Science and Technology, 2016

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

DYNAMIC GROUP TRIP PLANNING QUERIES IN SPATIAL DATABASES

FARHANA AKLAM

Date of Defence: September 24, 2019

Dr. W. Osborn Thesis Supervisor	Associate Professor	Ph.D.
Dr. J. Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. J. Anvik Thesis Examination Committee Member	Assistant Professor	Ph.D.
Dr. H. Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

I dedicate this thesis to my beloved parents for their love, endless support, encouragement and sacrifices.

Abstract

Trip planning queries are considered an integral part of Location Based Services. The advancement of positioning devices and highly available internet facilities enable users to access network information from anywhere at any time. In our research, we investigated Sequential Group Trip Planning (SGTP) queries. Given a set of starting and destination locations and an ordered sequence of Categories of Interests (COIs) for a group of users, a SGTP query returns the route for each user from their respective start and destination locations that minimizes the overall travel distance. We propose two approaches: Dynamic Group Trip Planning (DGTP) and Modified Dynamic Group Trip Planning (M-DGTP). The proposed DGTP approach enables users to plan a group trip in a more flexible manner and the M-DGTP approach optimizes the total travel distance of the group. We compare the results of our proposed strategies with an existing strategy called N-DGTP through experimental evaluation.

Acknowledgments

First of all, I would like to thank almighty Allah for giving me the ability, energy, and patience to conduct this thesis work. I would like to express my sincere gratitude to my supervisor Dr. Wendy Osborn for her continuous support, motivation and enthusiasm. Her guidance, valuable advice and suggestions helped me in all the time of research and writing of this thesis.

Besides my supervisor, I am grateful to my committee members, Dr. John Zhang and Dr. John Anvik for their guidance and valuable suggestions throughout my research.

I would like to express my gratitude to the School of Graduate Studies (SGS) of the University of Lethbridge for their financial support.

I would also like to thank all the people who helped me to make Lethbridge my new home. I am indebted to my Lethbridge family for their immense help, love and care.

Finally, I must express my heartiest gratitude to my father, mother and parents-in-law for their immense love and encouragement and my husband Zulfiqur for his endless support, motivation and patience. Thank you Zulfiqur for believing in me and your encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we discuss our research area and present the research problem with a practical example. Section 1.1 discusses the motivation behind our research, a real-life problem scenario on our research problem and the existing limitations of the current approaches. The contributions of our thesis are discussed in Section 1.2. Section 1.3 presents the organization of the rest chapters of this thesis.

1.1 Motivation

The Geographic Information System (GIS) is a principal technology that is stimulating continuing interests in Spatial Database Management Systems (SDBMSs) [30]. A GIS can be defined as a system designed to capture, store, manipulate, analyze, and present spatial or geographic data.

Nowadays the availability of Global Positioning Systems (GPS) has made it easier to precisely determine the location of any client/user [30]. Location-based services (LBS) have the capability to find the geographical location of a mobile device and provide services based on that location [27].

One of the popular services offered by LBS is *trip planning*. Trip planning queries (TPQs) [19] are considered an integral part of LBS. With the increasing availability of LBS, extensions of TPQs-namely Group Trip Planning (GTP) [15] queries, Sequential Group Trip Planning (SGTP) [2] queries, and Dynamic Group Trip Planning (DGTP) [33] queries-are also being studied.

A Sequential Group Trip Planning Query (SGTPQ) [2] over a road network can be defined as: a group of people having a set of starting locations S and a set of destination locations D, intend to visit some categories of interest (COIs) in an ordered sequence. The goal of a SGTPQ is to find the shortest route for a group of people starting from S, passing through at least one point of interest (POI) from each COI before ending at D.

Sequential group trips can be divided into two categories namely: static and dynamic trip. In a static trip the group size remains constant throughout the whole trip and the order of the COIs can not be changed. In a dynamic trip the group size can vary at any COI location, which means people can join and leave at any COI locations they wish. Additionally, in a dynamic trip the COIs do not need to be pre-determined in advance. In this thesis, we study the problem of DGTP for offering more flexible services to the group members for planning a trip efficiently.

The problem scenario can be demonstrated with an example. Consider some friends decide to visit four places: a coffee shop, a library, a restaurant, and a movie theatre. The type of the visiting places which are called Category of Interests (COIs) can be predetermined or can be changed during the trip. A Point of Interest (POI) is a specific instance of a COI. For example, Figure 1.1 has three coffee shops (C1, C2, C3) which belong to the COI coffee shop (C). Here, C1, C2 and C3 are three POIs that belong to a specific COI (C).

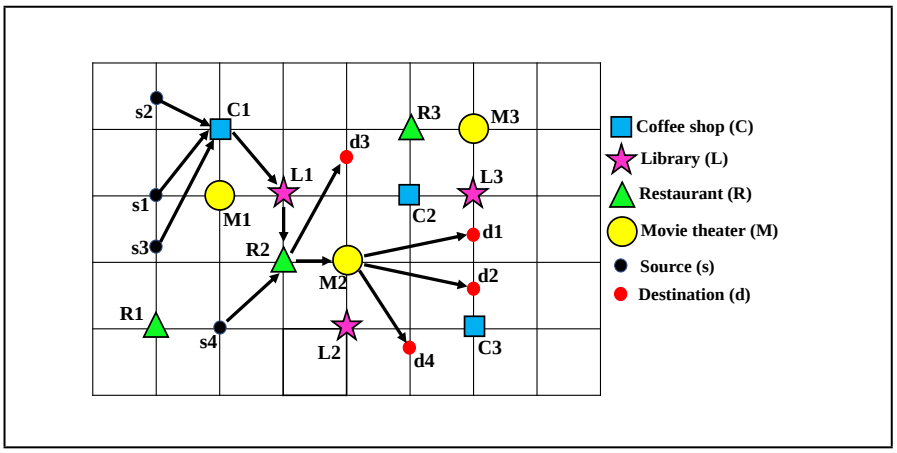

Figure 1.1: An example of DGTP

Let us consider, as the trip has been started there are several things that can happen:

- The group consisting of s1, s2 and s3 already have a predetermined sequence of COIs that they want to visit or they can first meet at the closest POI instance of the first COI (e.g. coffee shop) and decide where they want to go next. From this it can be said that, at every POI they visit the group members have the flexibility to change their mind on where to go next.

- Also, the group can visit as many COIs as they wish without having to predetermine any of them in advance.

- In addition, group members can join and leave the trip at any time. For example, let us assume the group wants to visit next a library, a restaurant and a movie theatre in that order:

  - A friend say s4, who cannot or does not want to visit the coffee shop and library can join the group directly at the restaurant for dinner. So, the trip has the flexibility to have any number of people join for any COI.

  - Alternatively, a friend s3 that does not want to or cannot stay for the movie can directly leave after dinner to their final destination. From this it can be said that, the trip has the flexibility to have any number of group members depart at any COI.

Although some approaches study the problem of DGTPQs in spatial databases [33] they still have the following limitations which motivate us to study them with a more flexible approach:

- They are only considered in Euclidean space,

- They require that the COIs that the group members intended to travel be specified in advance before the start of the trip,

- Finally, the existing approaches are costly as they apply a dynamic programming strategy. The time complexity of a dynamic programming algorithm is $O(n^3)$ [9] which even get worse when the trip is re-calculated again and again when any trip change is made in real-time.

1.2 Contributions

In this thesis, we study DGTPs in a more practical scenario, which makes the following contributions to this field of study:

- The trip is considered truly dynamic where the travellers can change their mind to decide COIs at any point in the trip. We also consider the group is dynamic where any number of travellers can join or leave at any COI,
- The travellers can choose any number of COIs to visit. In addition, the sequence of COIs does not have to be pre-determined before the start of the trip,
- The proposed strategy has been formulated based on a real-world spatial network,
- We conduct extensive empirical evaluations using the real-world California dataset [19] to present the effectiveness of our strategy.

1.3 Organization of Thesis

The rest of this thesis is organized in the following order. Chapter 2 presents background and related work for processing various types of SGTP and other nearest neighbour (NN) queries in spatial databases. Chapter 3 presents our proposed approaches for processing DGTP queries in spatial databases. The implementation details, experiments and results obtained from the proposed strategies are discussed in Chapter 4. Finally, Chapter 5 presents the conclusion and future research directions.

Chapter 2

Background and Related Work

In this chapter, a short introduction on the research area of Sequential Group Trip Planning (SGTP) queries and some of the existing works related to our research problem are summarized. This chapter is organized as follows: Section 2.1 discusses chapter preliminaries and Section 2.2, Section 2.3, Section 2.4 and Section 2.5 discuss related works in the research problem area.

2.1 Preliminaries

A Spatial Database (SDB) [30] can be defined as a database system that is used to store and access spatial data (i.e. the data defining a geometric shape) and their locations in n-dimensional space. In spatial databases, generally spatial data are stored as four categories namely: curve, surface, point and geometry collection [30].

In a Spatial Network Database (SNDB) [30] the data model that is utilized is a spatial network. A Spatial Network is defined as a collection of interconnected elements that have the shape of curves or poly-lines appearing in geographic applications [11]. The instance of spatial network used in location-based services is a road network where edges stand for road segments and nodes stand for intersections of road segments [11].

A road network contains locations on some vertices and edges representing Points of Interest (POIs). In our work, a POI in a road network belongs to exactly one Category of Interest (COI). For example, a COI can be “Airports” where each POI in this COI is a specific instance of an airport.

Location based services (LBS) are applications for users to seek information on the environment around them [27]. An LBS supports query processing for moving users. In addition, the positions of spatial objects being queried exist on spatial networks in many LBS-based applications. In such scenarios, the distance between any two objects is the length of the shortest path connecting them. Therefore in LBS, especially when dealing with SNDBs, the network distance has more importance over Euclidean distance as the Euclidean distance may not exactly approximate the real road distance [27]. These LBS applications require more complex query types that consider user trajectories on road networks [23].

The k-nearest neighbor (k-NN) search is the problem of efficiently finding object(s) (i.e. nearest neighbor (NN)) in a dataset that are closest to a given query point [35]. The number of NNs is usually defined as k, which can be a user-defined constant or vary for other reasons such as on the local density of points. The distance computation technique can generally be any metric measure, such as the standard Euclidean distance measure which historically has been the most common choice. However, k-NN searching strategy is becoming more frequent in road networks. Concepts such as Trip Planning Query (TPQ) [19], Group Trip Planning Query (GTPQ) [15] and Sequenced Group Trip Planning Query (SGTPQ) [2] are introduced and widely studied in SNDBs.

A TPQ [19] is defined as a query returning trip for a single user starting from a point S (i.e. the start), passing through at least one POI from each COI on the trip and ends at E (i.e. the end) A GTPQ [15] returns the best trip for a group and a SGTPQ [2] returns best trip for a group travelling through intended COIs in an ordered sequence.

Consider, some friends decide to meet at a coffee shop. From there they want to visit a library, a restaurant for dinner and a movie theatre. The goal of a SGTPQ is to find the minimal route for the group in such a way that the group can visit one POI from each of the COIs according to their intended order of visit. In the above example, the intended order of visit is a coffee shop, library, restaurant, and movie theatre.

In the upcoming sections some of the existing approaches on TPQs, GTPQs and SGTPQs are summarized. Additionally, some existing works on NN queries and GNN queries are summarized as TPQs, GTPQs and SGTPQs make use of NN and GNN queries. The upcoming sections are organized as: Section 2.2 discusses some of the existing work on query processing in spatial network databases, several nearest neighbor query processing strategies are discussed in Section 2.3, Section 2.4 discusses existing works on group nearest neighbor queries, and in Section 2.5, some of the approaches on group and trip planning queries will be discussed.

2.2 Query Processing In Spatial Network Databases

Papadias et al. [23] propose an index structure and the algorithms for several types of spatial queries including nearest neighbors using the spatial network distance. According to the authors, previous works on SNDB are based on Euclidean spaces only. The authors use a hard disk-based network representation to process spatial queries that use two frameworks: Euclidean restriction and network expansion. To process nearest neighbor queries the authors propose two algorithms, Incremental Euclidean Restriction (IER), and Incremental Network Expansion (INE).

The authors apply IER in high-dimensional similarity retrieval. In IER, first the k Euclidean NNs are obtained using an R-tree [13] and are sorted in ascending order of their network distance to a query point. d_{Emax} is set to be the distance of the k-th point from the query point. All subsequent NNs are retrieved incrementally until a larger Euclidean distance is found. The IER algorihm computes redundant network distances and to overcome this problem the authors propose INE. The INE algorithm examines entities according to the order in which they are encountered.

The authors compare the I/O access and CPU cost, as a function of $\mid S \mid / \mid N \mid$ (i.e. the ratio of entity to segment cardinality) and observe that with the increase of the cardinality the performance of IER improves. For nearest neighbor queries the authors observe that

INE is more efficient and robust when network distance computation is high where IER gives a better performance in dense network where the nodes lie very compact.

2.3 Nearest Neighbor Query Strategies

Chang and Kim [6] present a materialization-based k-NN query processing algorithm (called OMK) for k-NN queries in an SNDB. According to the authors, the existing approaches are only considered in Euclidean space. The architecture proposed by the authors deals with storing and indexing spatial network data, POIs, and moving objects.

First, the authors design a spatial network file organization for the maintenance of nodes and edges and use a node-node matrix file for storing network distances. They also use an edge information file to store the information of the edges and to maintain the POIs lying on an edge. The authors use R-tree [13] indexing for edges and specific POIs lying on the edges. For fast access the information for a specific POI they also use a B+ tree [4] indexing.

The authors compare the performance of their proposed OMK algorithm with INE [23]. They observe that the performance of OMK improves with the increase of the value of k. The authors also observe that for k=1 the performance of OMK is nearly same as INE and for k=100 OMK performs 1.5 times faster than INE.

Lee et al. [18] present a general framework named ROAD (Route Overlay and Association Directory) to evaluate Location-Dependent Spatial Queries (LDSQs) that search for spatial objects in a road network. According to the authors, the existing Euclidean distance-based approaches are not applicable for estimating distances in a road network.

They design a framework to address network traversal and object access. The authors use a dynamic object mapping mechanism and search space pruning technique on road networks which can be efficiently used in an NN search over a large-scale network. The authors claim their contributions as:

i. Present a system framework for efficient processing of LDSQs,

ii. Develop a Rnet (regional sub-networks) hierarchy to reduce index overhead,

iii. Develop an efficient algorithm for nearest neighbor queries,

iv. Develop a maintenance scheme to handle network and object change.

The authors observe that their proposed algorithms significantly minimize I/O cost and search time. They also state that the proposed approach outperforms existing Euclidean distance-based approaches and network expansion-based approaches.

Abeywickrama et al.[1] present a memory-resident method to efficiently process k-NN query on road networks. According to the authors, the existing approaches propose disk-based indexes. However, they are only evaluated and compared in main memory. To overcome this limitation, the authors provide their following contributions including:

i. Optimisation of five (IER [23], INE [23], Distance Browsing (DisBrw) [25], ROAD [18] and G-tree [36]) open-sourced algorithms,

ii. Suggestions on main-memory implementations.

The authors compared the performance of the algorithms considering the parameters k, density, network size and object distance and observed that the G-tree generally outperforms INE, DisBrw and ROAD where IER significantly outperforms every competitor in all.

Chen et al. [8] propose query homogeneity to process k-NN queries in a road network. The authors propose a new query framework named Spatial query by Homogeneity Identification (SHI) to reduce waiting time. The SHI framework combines query processing and queue processing to improve performance.

Sun et al. [32] present a distance pre-computation technique to solve k-NN query processing over moving objects in road networks. The proposed pre-computation approach by the authors utilizes an off-line road network computation and on-line query processing to simplify the computation. According to the authors the proposed pre-computation approach shows excellent performance with good precision. In future, the authors propose to

conduct a comparative study between their pre-computation based approach and Voronoi diagram-based approach [17].

Mouratidis et al. [20] study the problem of monitoring continuous NN in a road network. According to the authors, previous approaches are limited to snapshot queries over static data that utilize the Euclidean distance metric. The authors propose two strategies, namely Incremental Monitoring Algorithm (IMA) and Group Monitoring Algorithm (GMA). The first strategy updates the current NN sets and the second strategy deals with processing time reduction.

The authors use sub-networks of the San Francisco road system to evaluate IMA and GMA and compare their algorithms against the overhaul method (OVH) [20]. For CPU time versus object and query cardinality, GMA outperforms both IMA and OVH, with IMA in second place in all cases. In the case of CPU time versus query agility (the percentage of query performed per timestamp) and speed GMA again outperforms as before. The authors observe that for Gaussian queries GMA works best while IMA works best for uniform queries.

Kolahdouzan and Shahabi [17] propose two ideas for addressing Continuous k-Nearest Neighbor queries (C-kNN) queries in a SNDB namely:

- Intersection Examination (IE)
- Upper Bound Algorithm (UBA)

According to the authors, existing works to process C-kNN queries are limited to the Euclidean distance metric. To overcome this limitation, the authors propose a Voronoi-based Network Nearest Neighbor (VN^3) approach based on the Network Voronoi Diagram (NVD) [16] concept to address k-NN queries in SNDBs.

The authors consider a set of limited number of points called generator points (i.e. POIs), in the Euclidean plane and associate all locations on the plane to their closest generator(s). A Voronoi diagram is the set of Voronoi Polygons (VP) associated with all the

generators, where each VP is the region formed by the set of locations assigned to each generator. The purpose of the region formed for each generator (i.e., the VP) is partitioning a large network into small regions so that distances can be pre-computed both within and across the regions. The Voronoi polygon provides a safe region for the point in the VP. In other words, if a query point remains in the VP, the point in the VP is guaranteed to be the nearest neighbour.

An NVD is a specialized Voronoi diagram defined for spatial networks. According to the authors, in an NVD the locations of the objects are restricted to those lying on the edges of the graph, with the distance between objects being the spatial network distance rather than Euclidean distance. According to the authors, the VN^3 algorithm consists of the major components:

i. Solution space pre-calculation,

ii. Index structure utilization,

iii. Pre-computation of the network distances between the border points of NVP.

The proposed IE approach by the authors finds the k-NNs of the intersections on the path. To achieve a better performance the authors propose the UBA approach which only computes the k-NN for a subset of the nodes of the path.

The authors use real-world data set for their experimental evaluation. They observe that for query response time, UBA outperforms IE. While for sparse points of interest and small values of k, UBA outperforms IE. When UBA performs the computation of k-NN queries only for a subset of the nodes of the path it shows a better query processing time.

Gao and Zheng [12] present a Continuous Obstructed Nearest Neighbor (CONN) search, which deals with the impact of obstacles for determining the distance between two objects in two-dimensional space. According to the authors previous work on spatial queries ignore obstruction. They propose the Incremental Obstacle Retrieval (IOR) algorithm to retrieve

objects in ascending order of their minimal distances. The authors claim the five following contributions:

i. CONN search formalization,

ii. Introduction of control point concept,

iii. Solving quadratic inequalities for split point formation,

iv. Efficient algorithm development to facilitate COk-NN retrieval.

According to the authors both the dataset and the obstacle set is indexed by an R-tree [13]. For each data point, the authors perform the following tasks:

i. Find obstacles that might affect the obstructed distances,

ii. Identify the control points,

iii. Evaluate the impact on the current result list.

The authors measure the performance of the algorithms on both synthetic and real-world datasets. The authors observe the performance of the COk-NN algorithm for the effect of the ratio of $| P | / | O |$ (Where P is data set and O is obstacle set) first decreases then increases between the range of 0.1 to 10. When the authors vary buffer size (% of the tree size) between 1% to 32% of each R-tree [13] size, the authors observe that the I/O performance improves.

2.4 Aggregate Nearest Neighbor Queries

One type of nearest neighbor query in a spatial network is an Aggregate Nearest Neighbor (ANN) query [22]. ANN queries, also known as Group Nearest Neighbor (GNN) queries, apply an aggregate function to optimize the travelling routes for multiple users. An ANN query retrieves the data point with the minimized aggregate function (e.g. distance) with respect to a set of query points [22]. Papadias et al. [21] introduce the concept

of GNN queries as a novel form of NN search. The authors define a GNN query as: for two sets of points P and Q, a GNN query retrieves the point of P such that the sum of cost to all points in Q is minimized [21].

Papadias et al. [21] study the problem of GNN queries and for memory-resident queries. The authors propose three algorithms :

i. Multiple query method (MQM),

ii. Single point method (SPM),

iii. Minimum bounding method (MBM).

The MQM strategy performs an incremental nearest neighbor search for each point in Q and combines the results. The SPM method processes GNN queries by a single traversal to avoid multiple access to the same node, which according to the authors is a problem with the MQM method. The proposed MBM method also performs a single query like SPM but it computes a minimum bounding rectangle of the query point instead of centroid. For disk-resident queries the authors propose:

i. Group Closest Pair method (GCP),

ii. File-Multiple Query method (F-MQM), and

iii. File-Multiple Bounding method (F-MBM).

The proposed GCP method considers query points Q that are indexed by an R-tree [13]. The proposed F-MQM and F-MBM method do not require any indexing on query points.

The authors compare the MQM, SPM and MBM strategies for main-memory queries and observe that MQM is the worst method as the cost increases fast with query cardinality, and that MBM is the most efficient method. For the disk-resident approaches, the authors observe that GCP gives the worst performance. The authors also observe that F-MBM shows the best performance among all three algorithms but in the case of query dataset partitioned in a small number of groups F-MQM is more preferable than F-MBM.

Yiu et al. [34] propose three algorithms to address ANN queries in a road network:

- Incremental Euclidean Restriction (IER),
- Threshold Algorithm (TA),
- Concurrent Expansion (CE).

They consider alternative aggregate functions and techniques that utilize Euclidean distance bounds, spatial access methods, and network distance materialization structure. The proposed IER algorithm by the authors apply two optimization methods to overcome two problems:

i. Minimize the shortest path computation,

ii. Redundant visit to the same edges.

In TA, the network node with the minimum aggregate distance from a set of query points is found by concurrent incremental expansion of the network and performing a top-k aggregate query processing technique [10]. The CE algorithm works similar to TA but it avoids shortest path computations.

Through experimental analysis the authors observe that for different aggregate functions IER has the best overall performance, with the lowest number of page accesses in comparison with other methods. The authors also observe that the effect on the number of page accesses for IER is linear to the network size, where TA and CE show a super linear cost with respect to network density. When the edge weights are proportional to their lengths, the authors observe that IER is the best algorithm.

Safar [24] addresses the problem of Group k-Nearest Neighbors (Gk-NN) queries in SNDBs. According to the author, previous works on Gk-NN queries consider Euclidean distance rather than network distance. The author proposes an approach that uses Network Voronoi Diagram (NVD) properties with a progressive incremental network expansion and the spatial network distance to determine the inner network distances.

Sultana et al. [31] introduce the problem of Obstructed Group Nearest Neighbor (OGNN) queries. Existing algorithms on GNN queries in Euclidean space and road networks ignore the impact of obstacles like buildings and lakes to compute trip distances. To overcome the existing limitation with obstacles the authors propose a novel approach called the Multi Point Aggregate Obstructed Distance (MPAOD) computation technique to process an OGNN query. The authors use geometric properties to prune search space. Through experimental analysis using real and synthetic datasets the authors validate the efficiency of their proposed approach.

2.5 Trip Planning Queries

An Optimal Sequenced Route (OSR) query [29] (i.e. Sequenced Trip Planning Query (STPQ)) is defined as a route for a single user which starts from a given source location and passes through a sequence of locations in such a way that the total trip distance is minimized. Each location, or Point of Interest (POI) is associated with a type of location, or a Category of Interest (COI).

In this section, work on Sequenced Group Trip Planning Queries (SGTPQs) [2] is also presented. SGTPQs [2] are an extension of STPQs [29] where the goal is to find the routes for multiple group members starting from their respective source locations (possibly distinct) to their respective destination locations (possibly distinct) through an ordered sequence of POIs in such a way that the total trip distance is minimized.

This section is organized as follows: Section 2.5.1 discusses some of the existing works on Trip Planning Queries, Section 2.5.2 discusses some of the existing works on Group Trip Planning Queries and some of the existing approaches on Sequenced Group Trip Planning Queries are discussed in Section 2.5.3.

2.5.1 Trip Planning Queries

Li et al. [19] introduce the concept of a new type of query called Trip Planning Queries (TPQ) in SDB. The authors claim their contributions are:

i. Introduction of algorithms with various approximation ratios in terms of total number of categories and maximum category cardinality,

ii. Proposal of two approaches suitable for main memory evaluation,

iii. Various adaptations of the presented algorithms for practical scenarios using spatial index structure and transportation graphs.

The authors present two greedy algorithms suitable for main memory evaluation with tight approximation ratios and with respect to total number of categories:

i. Nearest Neighbor Algorithm (A_{NN}) : In the proposed A_{NN} algorithm a trip is formed by iteratively finding the nearest neighbor of the previous vertex that was added to the trip.

ii. Minimum Distance Algorithm (A_{MD}): The proposed A_{MD} algorithm by the authors first selects a set of vertices (one vertex per category) in such a way that cost of each vertex is minimum among all vertices belonging to the respective category. Then the trip is formed by visiting the selected set of vertices in the nearest neighbor order.

The authors implement proposed TPQ algorithms for both Euclidean space and road networks. For both A_{NN} and A_{MD}, when applied in Euclidean space the authors use R-tree [13] indexing for the spatial dataset. The A_{NN} algorithm for road networks uses a simple extension of the Dijkstra algorithm [9] to locate the nearest neighbor and the R-tree [13] to locate the nearest neighbor of the last stop of the trip. According to the authors for A_{MD}, the Dijkstra algorithm [9] cannot be applied directly to the road networks. According to them, the Dijkstra algorithm does not necessarily minimize cost.

The authors used both real-world datasets and synthetic datasets generated on real road networks to perform their experiments. Experiments in synthetic road network datasets shows that A_{MD} gives a 20%-40% better trip result than A_{NN} in terms of trip length. Experiments in the Euclidean datasets shows that A_{NN} achieves better performance than A_{MD} for higher densities.

Sharifzadeh and Shahabi [28] study the problem of Optimal Sequenced Route (OSR) query in vector spaces. According to the authors, the classic shortest path algorithms like Dijkstra [9] are impractical for real world scenarios because the graph is large. The authors propose a light threshold-based iterative algorithm (LORD) and an extension of LORD named R-LORD which uses an R-tree [13] to locate the threshold values more efficiently. The authors compare the performance of LORD and R-LORD in terms of query response time and I/O access and they observe that in both cases R-LORD is the superior one.

Sharifzadeh and Shahabi [29] also study the problem of Optimal Sequenced Route (OSR) query using Voronoi Diagrams in both vector and metric spaces. According to the authors, their previous approach [28] only focus on vector space. They propose a algorithm named Voronoi-based Optimal Sequenced Route query (OSRV) which uses a precomputation approach to answer an OSR query. The authors compare the performance of their proposed OSRV algorithm with their proposed R-LORD [28] approach in terms of query response time and observe that OSRV always outperforms R-LORD. Additionally, the authors compare the impact of the cardinality of the dataset on the efficiency of OSRV and observe that again OSRV is a superior approach to R-LORD.

Chen et al. [7] propose a multi-rule partial sequenced route (MRPSR) query. According to the authors the MRPSR query includes both the TPQ [19] and OSR [29] queries. The authors prove that MRPSR is NP-hard and present three heuristic algorithms to provide near-optimal solutions for an MRPSR query. The authors observe through simulations that their proposed algorithms can answer a MRPSR query effectively and efficiently. Through experimental evaluation the authors observe that the proposed algorithms perform remark-

ably better in case of response time in comparison with LORD-based brute-force solution. The authors also observe that the resulting route is slightly longer than the shortest route.

2.5.2 Group Trip Planning Queries

Hashem et al. [15] are the first to introduce the concept of a Group Trip Planning (GTP) query. They also propose and evaluate algorithms for their proposed new type of query. The authors define kGTP as: If a group queries for k sets of data points for a GTP then the query is called a k group trip planning (kGTP) query.

To process kGTP queries the authors first consider a method that calculates a trip distance for every user in the group independently. This process continues to determine the sets of data points that minimize the group travel distance. However, the authors find that this strategy incurs a high computation cost as the same data points are accessed by multiple users separately. To overcome this difficulty the authors propose two methods to process kGTP planning queries:

i. Iterative Approach (IA),

ii. Hierarchical Approach (HA).

The IA algorithm makes use of the GNN [21] algorithm to determine the first GNN from the starting points of the group members to the POI of the first COI. Then it finds the nearest POI of the next COI from the first chosen POI. This continues similarly for all the COIs in the trip sequence. Finally, a GNN is determined from the POI of the last COI towards the destinations of the group members.

Once the initial solution is found, the IA algorithm then backtracks to find group trips that are shorter than the initial one for all GNNs and the shortest route is reported. According to the authors, IA does not process an independent trip for every user but it still requires accessing the same data multiple times. The HA method evaluates a kGTP query in a single database traversal incurring less computational overhead. The authors conduct an experimental evaluation considering both the IA and HA algorithms and observe the

effect of group size, query area and data set size. The authors observe that HA performs faster and requires less I/Os than IA.

Hashem et al. [14] study an improvement to their previously proposed GTP [15] strategies and develop both optimal and approximation algorithms to process GTP queries for Euclidean space and road networks. According to the authors GTP queries are computationally expensive because of multiple types of POIs and aggregate trip distance computation. The authors develop two new strategies based on the geometrical properties of the ellipse to process kGTP queries:

i. Range based group trip planning (R-GTP)

ii. Incremental group trip planning (I-GTP)

According to the authors, as the hierarchical approach for group trip planning (GTP-HA) approach is not easily adaptable for road networks they propose a naive approach (N-GTP) to process group trip planning queries. Through experimental analysis the authors show that the proposed strategy outperforms the GTP-HA [15] in the Euclidian space and the N-GTP in road networks.

Samrose et al. [26] also study the problem of Group Optimal Sequenced Route (GOSR) queries in road networks. According to the authors, the proposed GTP [15] algorithms in Euclidean space incurs very high query processing overhead and the algorithm cannot be extended to road networks. The authors utilize elliptical properties to refine POI search space. According to the elliptical property, the distance between two foci of an ellipse via a point outside the ellipse is greater or equal to the length of the major axis. That is why, a POI outside the refined area cannot contribute to minimize the aggregate travel distance. The authors define a kGOSR query as: a query returning k sets of data points for a GOSR query. The authors propose two algorithms:

- Naive approach (kGOSR-NA) : The naive approach proposed by the authors does not apply any pruning technique to select the POIs. They calculate the trip distances for

all the POIs that are intended to be visited and finally the k smallest values for the total trip length are returned.

- Elliptical Approach (kGOSR-EA): The elliptical approach proposed by the authors at first calculate the geometric centroid of the source locations. Then it retrieves nearest POI of C_1 (i.e. First COI) with respect to the source centroid. Starting from C_1 all the NNs are retrieved incrementally with respect to the previous POI until C_{m-1} (m = the number of COIs intended to visit). Then k number of NNs from C_m is identified with respect to the POI P_{m-1}. Finally, the kGOSR query returns the output containing k sets of POIs that have the k smallest trip distances.

The authors observe through experimental analysis that kGOSR-EA outperforms kGOSR-NA in terms of I/O and processing time.

2.5.3 Sequenced Group Trip Planning Queries

Ahmadi and Nascimento [2] also study the problem of Sequenced Group Trip Planning Queries (SGTPQs) in Spatial databases. According to the authors, the Iterative Approach (IA) [15] algorithm for SGTPQs lacks in optimal answer generation and computational efficiency. To overcome the mentioned difficulties the authors propose:

i. A small modification to IA [15] to produce optimal solutions called Revised Iterative Approach (RIA),

ii. Introduction of a new approach named Progressive Group Nearest Neighbor Exploration (PGNE),

The RIA algorithm makes use of the GNN [21] query to return a POI which has the smallest sum of distances to all users in the group. The location of the previously retrieved POI multiplied by the group size is considered while retrieving POI from last COI.

To overcome the drawback of optimal route generation with higher processing time the authors propose the PGNE algorithm which uses a mixed Breadth-Depth first search strat-

egy. The PGNE approach is an iterative process which generates a set of Partial Group Trips (PGTs) with length and corresponding Lower Bound (TDLB) of its group travel distance. A PGT is the sum of the distances of POIs from sources to the last POI multiplied by the group size.

A Lower bound for the group total travel distance is the sum of distance of calculated PGT and distance from its last POI to the destinations of the group members. In each iteration a PGT is stored in a MinHeap based on its TDLB . In each iteration the PGT having smallest TDLB is fetched and two operations are applied:

i. Replacement Operation,

ii. Progressive Operation.

The Replacement Operation takes place by replacing the last POI in the PGT with another POI (GNN towards source and destination centroid) from the same COI. The Progressive Operation takes place by appending a POI (GNN from replaced POI in replacement operation towards destination centroid) from next COI based on the predefined order of the COIs.

The authors use both real and synthetic datasets to compare the results. The POIs are indexed by an R* tree [5]. The parameters varied throughout the experiments are: number of categories to be visited, the group size, and the query area. For varying group size and query area the authors observe that PGNE retrieves a smaller number of POIs and performs better than RIA.

The authors observe PGNE requires more POI retrievals but less response time than RIA for varying number of categories. For synthetic datasets the authors observe that in each case PGNE performs better than RIA. As a part of future work the authors propose to work on avoiding pre-computation of the shortest path and time-dependent SGTPQs.

Ahmadi and Nascimento [3] propose additional strategies for SGTPQs. According to the authors previous works on SGTPQs needs to inspect some POIs repeatedly which incurs

redundant computation. To overcome the problem of redundant computation the authors propose Iterative Backward Search (IBS) which is a depth-first search algorithm allowing efficient reuse of previously computed optimal partial group trips.

According to the authors the IBS works by generating an initial trip by visiting nearest POIs belonging to the succeeding unvisited COIs. The trip is then backtracked in every step to generate a new shorter trip to the current best one. The authors claim that IBS differs from IA [15] in three aspects:

i. While retrieving POIs from different COIs IBS considers the centroids of the source and destination locations,

ii. To shrink the POIs search space IBS applies an efficient pruning strategy,

iii. To overcome redundant computational overhead of same POIs IBS considers a Suffix Optimality Principle which saves a considerable amount of computation.

Through experimental evaluation using real-world and synthetic datasets the authors observe that IBS shows best response time in all cases for varying the parameters like: POI density, query area, group size, number of COIs and POI distribution.

Tabassum et al. [33] introduce the concept of dynamic groups for Group Trip Planning(GTP) queries in Spatial Databases. Previous approaches of GTP queries are limited to a fixed group size during the trip. To overcome this limitation the authors propose a new concept named Dynamic Group Trip Planning (DGTP) queries where the group size can change at any POI. That means any number of new or old members can join or leave the group at any point of the trip which can be either predetermined or in real-time. The authors propose efficient solutions to process DGTP queries in Euclidean space and through experimental study compare their results with the naive [14] approach. The authors claim that the experimental study shows that proposed strategy with dynamic groups outperforms in query processing time and I/O access in comparison with naive approach.

According to the authors, for every change of group members a straightforward application of existing GTP algorithms in a DGTP query will obtain a very high processing overhead as the same POIs will be retrieved multiple times which means repeated computations. The authors claim that the proposed approach to process DGTP queries can handle both real time and predetermined change in group size at any part of the group trip.

The authors first implement the naive [14] approach using dymanic programming which works in two phases:

i. Optimal answer computation by retrieving all POIs from database for initial trips,

ii. Updating existing optimal answer whenever there is a change in group size.

According to the authors, the implemented NaiveDGTP runs the dynamic programming algorithm multiple times over the entire dataset that results in extensive I/O overhead and processing time. The authors claim that this issue has been resolved by their proposed FastDGTP approach which has two key improvements over the NaiveDGTP approach:

i. This approach utilizes the trip information of users to compute a pruning bound that enables to search over a small data space instead of the entire one,

ii. POIs are stored in a cache so that the information can be re-used to overcome same POIs retrieval again and again in the update phase.

The computation of initial optimal answer set is comprised of the following steps:

i. A heuristic answer computation for the DGTP query,

ii. Refining the search space using the computed heuristic answer and trip set T,

iii. A range query operation to retrieve all candidate POIs that lies inside the refined search space,

iv. Dynamic programming algorithm is applied to compute optimal answer set from computed candidate POIs set.

The update phase to update the calculated optimal answer consists of the following steps:

i. Trip set T and POI set PT is updated,

ii. Computation of new pruning bounds for the search region depending on updated trip information and existing optimal answer,

iii. A range query execution to retrieve POIs that exist inside the new bound,

iv. Dynamic programming technique to compute the updated optimal answer set.

For experimental analysis the authors used real-world California road network data set that contains 87635 POIs of 63 different categories. The California road network has 21048 nodes and 21693 edges. The authors also generated two types of synthetic data sets using uniform and zipfian distribution for POIs. The authors carry out the experimental analysis by varying four parameters:

- Total number of users, n
- Number of POI types, m
- The query area, A and
- The size of data set, ds.

The authors used 100 randomly generated DGTP queries and calculated the average processing time and average I/O access for each experiment. The experimental analysis shows that if any of the values of n, m, A and ds increases, the processing time also increases for both real-world and synthetic data sets. The authors observe that the I/O of NaiveDGTP approach is constant as it does not use any pruning bound. The authors claim that their approach is 99.74 % faster and requires 94 % less I/O in comparison with the naive approach.

Utilizing the elliptical properties the authors developed new pruning techniques to refine the POIs search space. Their proposed dynamic programming techniques minimizes the

travel distance for a dynamic group. In future the authors propose to work on handling the dynamic change of POIs types in real time after the start of the group trip.

2.6 Summary

To process trip planning queries various research works have been undertaken. In this chapter, we summarize some of the existing approaches on TPQs, GTPQs and SGTPQs along with some of the existing works on NN and GNN queries as TPQs, GTPQs and SGTPQs make use of NN and GNN queries.In the next chapter, we present our proposed strategies to process DGTPQs (i.e. dynamic group trip planning queries).

Chapter 3

Dynamic Group Trip Planning

In this chapter we propose two approaches for processing DGTPQs. The first approach is called the Dynamic Group Trip Planning (DGTP) approach and the second approach is called the Modified-Dynamic Group Trip Planning (M-DGTP) approach. Before we discuss the proposed strategies in detail, a description of problem statement and a sample of road network is presented in Section 3.1.

The rest of the chapter is organized as follows: Section 3.2 presents a high level overview of DGTP and M-DGTP approach, Section 3.3 discusses the DGTP approach and Section 3.4 discusses the M-DGTP approach. A running example for both the DGTP and M-DGTP approaches is presented in Section 3.5.

3.1 Preliminaries

The Section 3.1.1 introduces the problem statement, terminologies, and symbols used to explain DGTP and M-DGTP. A sample road network with its elements is presented in Section 3.1.2.

3.1.1 Problem Statement

In Table 3.1, given a set of starting and corresponding destination (e.g. home locations) of users (s and H, respectively), a sequence C of m COIs and a set of joining and leaving locations (COIs) of x and y users (J and L, respectively), for a given set of s, H, C, J and L a DGTP query returns a sequence of POIs P (where P_i is an instance of C_i) to be visited by the

Table 3.1: Meaning of symbols used in DGTP and M-DGTP

n	Number of users in beginning
$s = \{s_1, s_2, s_3, ..., s_n\}$	Starting locations of users for current iteration
$H = \{H_1, H_2, H_3, ..., H_n\}$	Destination or home locations of users for current iteration
m	The number of COIs
$C = \{C_1, C_2, C_3, ..., C_m\}$	Sequence of COIs in intended order of visit
$P = \{P_1, P_2, P_3, ..., P_m\}$	A sequence of POIs corresponding to C
P_i	An instance of C_i
x	Number of members that want to join
y	Number of members that want to leave
$X = \{s_1, s_2, s_3, ..., s_x\}$	Joining locations of the x new users
$Y = \{H_1, H_2, H_3, ..., H_x\}$	Home locations of the x new users
$J = \{C_1, C_2, C_3, ..., C_x\}$	Joining COIs of new users
$L = \{C_1, C_2, C_3, ..., C_y\}$	Leaving COIs of users
$S = \{S_1, S_2, S_3, ..., S_n\}$	Source locations of members for the next COI
$\{(u_1, v_1), (u_2, v_2), (u_3, v_3),...., (u_k, v_k)\}$	Set of edges of the network

group members in such a way that the distance of the trip is minimized. Here, we express the minimized distances using two terminology, namely: total distance and trip distance.

- **Total distance:** Total distance is the overall distance travelled by group members starting from their source locations to their respective destination (e.g. home) locations. Mathematically, total distance can be expressed as:

$$\text{Total distance} = \text{source distance} + \text{group distance} + \text{home distance} \tag{3.1}$$

  - source distance = sum of the distances of all members from their respective start locations to the COIs they want to join

  - group distance = sum of partial group distances for trip through C, where,

$$\text{Partial group distance} = \text{current group size at } P_i \times \text{distance from } P_i \text{ to } P_{i+1} \tag{3.2}$$

  - home distance = sum of the distances of all members from their last visited POIs to their respective destination locations

- **Trip distance:** Trip distance is the distance travelled by the group members starting from their source locations to the last POI they visit. Mathematically, trip distance can be expressed as:

$$\text{Trip distance} = \text{Total distance} - \text{home distance} \tag{3.3}$$

  The starting and departing locations and COIs of the members are randomly located for a trip. In some cases, the scenario can arise where the trip distance of members is significantly shorter than the destination or home distance of the members. To keep track of the original trip distance (from start to last visited P_i) with respect to total distance travelled by the members we calculate the trip distance.

3.1.2 Sample Road Network

A road network contains the information on its nodes and edges. A node in a road network contains information like Node ID, Longitude, and Latitude, while an edge contains information like Edge ID, Start Node ID, End Node ID and Edge Length. In other words an edge length is the cost to travel from one node to another associated with the edge. In a road network a node can be connected with multiple nodes. Additionally, a road network contains Point of Interests (POIs) information. A POI consists of information like Category ID (i.e. COI), Longitude and Latitude.

Figure 3.3 is a sample of a road network segment merged with its POIs. The nodes and edges show the ways to move within the network. The nodes are represented by circles

and marked with a Node ID. The edges are represented by straight lines connecting the nodes. For example, in the following figure there is an edge between nodes 0 and 6 where, the travel cost between these two nodes is 2. The colored rectangular boxes lying on the edges represent the POIs. Each of the rectangular boxes belongs to a specific category. Additionally, the rectangular boxes having same colour represent they belong to the same category.

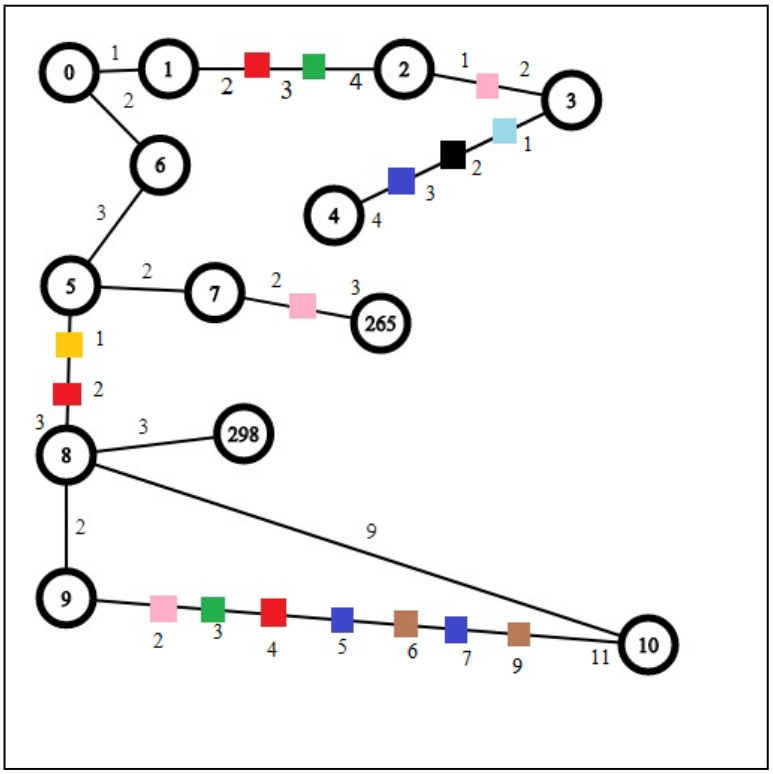

Figure 3.1: A sample road network with POIs

Similarly, to move from nodes 1 to 2 in the sample network the travel cost will be 4. There are two different POIs (Red, Green) lying on the edge (the edge between nodes 1 and 2) belonging to two different categories. The travel cost from node 1 to the POI marked by Red colour is 2 and the travel cost from node 1 to the POI marked by Green colour is 3. The movement can be bidirectional. That means to move from nodes 2 to 1 the total cost will be 4. The travel cost from node 2 to the Green POI can be calculated by subtracting the travel cost of the corresponding POI (from node 1) from total cost. Therefore, the cost

to travel from node 2 to the Green POI is 1 (i.e. $4-3=1$). Similarly, the travel cost from node 2 to the Red POI is 2 (i.e. $4-2=2$).

3.2 Overview of DGTP and M-DGTP Strategy

Before we discuss our proposed DGTP and M-DGTP approaches in detail, a high level overview of both the approaches are discussed in this section. Figure 3.2 present a generalized flow-diagram for both DGTP and M-DGTP approaches. The flow-diagram works as follows:

i. Initial values of group_size [1], Total_distance and Trip_distance is set to zero,

ii. Start and destination (e.g., home) locations (i.e. vertices) of the group members for the current iteration are taken as input,

iii. The updated value of group_size is calculated as: the sum of previous group_size and number of new joining member(s),

iv. The next desired COI that the current members want to visit is taken as input,

v. The POI (P_i) of the desired COI having minimum the aggregate distance from all the source locations of current group members is determined. This POI location will be considered as the one source location for the entire set of current group members for the next POI (P_{i+1}) search,

vi. Checking for the departure of any group member(s):

   (a) If any member(s) want to leave, the distance from their last POI (P_i) to their home and their individual_trip_distance and individual_total_distance are calculated. Total_distance, Trip_distance and current group_size are also calculated,

[1] In beginning, the group_size=0 and this also happens when the trip is finished and everyone has departed for home

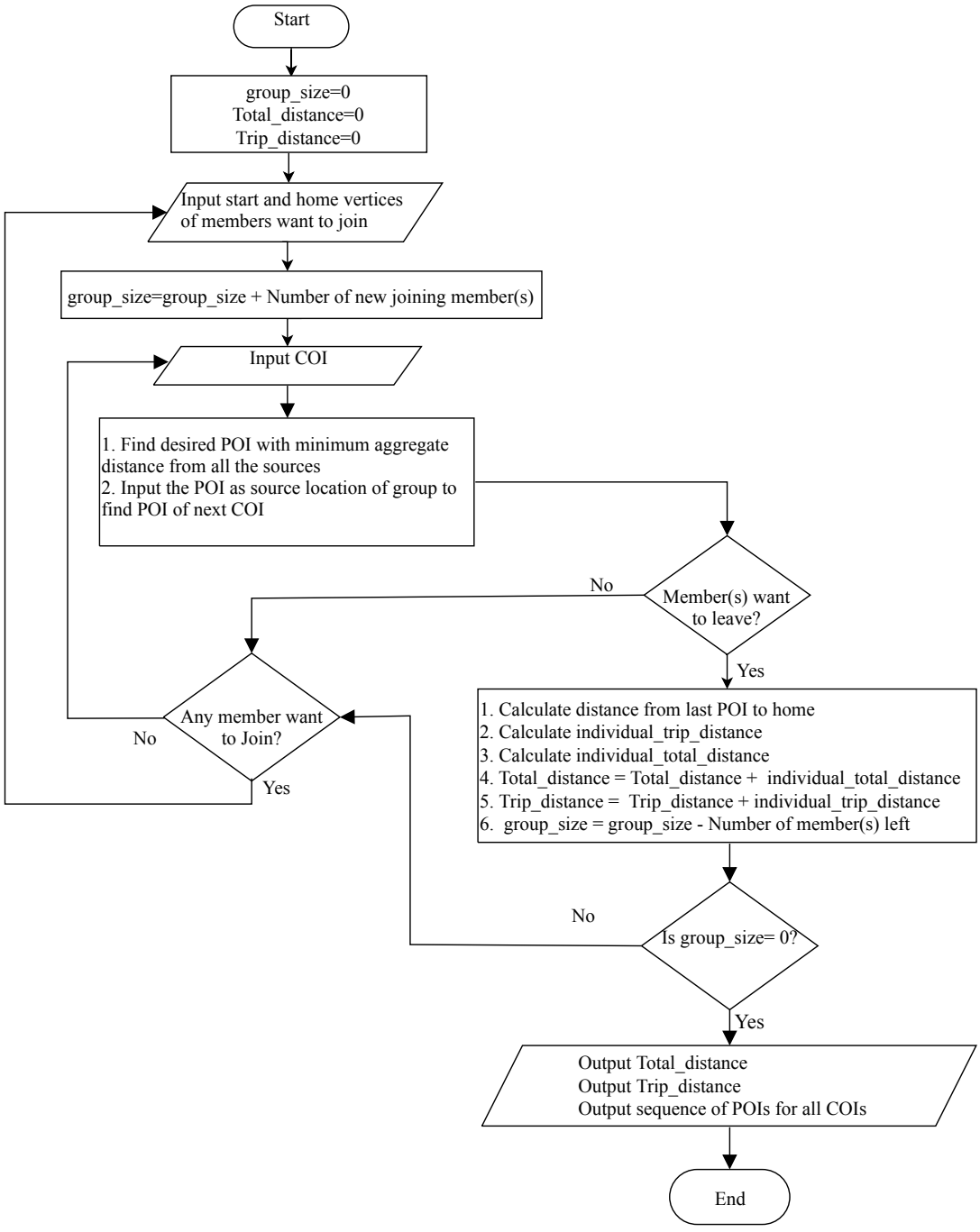

Figure 3.2: Generalized flow diagram for DGTP and M-DGTP

updated and stored for future use. The resulting group_size is checked in step viii below.

(b) If no member wants to leave from the current group location then step vii is performed next,

vii. Checking for the addition of any new group member(s):

(a) If any new member(s) want to join, their start and home locations are taken as input and the algorithm continues from step ii above,

(b) If no new member want to join then the algorithm continues from step iv above,

viii. Checking current group size:

(a) If the group size is equal to zero that means there is no members left in the group and the algorithm continues from step ix below.

(b) If the group size is not equal to zero then the algorithm continues from step vii above,

ix. The Trip distance and Total distance for the entire trip is calculated and the chosen POIs sequence for the trip is returned as output.

3.3 Dynamic Group Trip Planning (DGTP) Approach

In this section, our proposed first strategy called DGTP is discussed in detail. Section 3.3.1 presents the procedure of shortest distance and path computation, while the aggregate distance calculation procedure is presented in Section 3.3.2 and Section 3.3.3 presents the result computation procedure.

3.3.1 Step-1: Shortest Distance and Path Computation

Our intention is to find out the desired target location (POI P_1) which is nearest from current locations of all the members who are travelling the current moment in the trip. In

the road network, all the target locations (POIs) are located on edges $\{(u_1, v_1), (u_2, v_2), (u_3, v_3),...., (u_k, v_k)\}$.

i. Using the Dijkstra algorithm [9], the shortest distances and paths from a specific source location, say S_i to all the vertices of the graph are calculated. At the end of this step, we have the shortest distances from the source location to all the vertices.

ii. After that, using the calculated distances of each vertex from source S_i, the distance to each POI that belongs to the current COI is calculated as follows: if the distance from source S_i to vertex u of an edge (u, v) that is hosting a POI, say P_i is Distance$[S_i \rightarrow u]$ and the distance from u to P_i is Distance$[u \rightarrow P_i]$ then the distance from source S_i to P_i which we call Distance$[S_i \rightarrow P_i]$ is calculated by adding two distances in the following way and consequently the predecessor (previous vertex) of P_i is set to u.

Distance$[S_i \rightarrow P_i]$ = Distance$[S_i \rightarrow u]$ + Distance$[u \rightarrow P_i]$

and, Predecessor$[P_i] = u$

iii. If the P_i is reachable through the other vertex v of the same edge (u, v) hosting the POI and the distance from source S_i to P_i through vertex v is less than the distance through u then, the distance Distance$[S_i \rightarrow P_i]$ is updated with this new minimal distance calculated in the following way and consequently the predecessor of P_i is reset to v.

Distance$[S_i \rightarrow P_i]$ = Distance$[S_i \rightarrow v]$ + Distance $[v \rightarrow P_i]$

and, Predecessor$[P_i] = v$

For all group members $\{s_1, s_2, s_3, ..., s_n\}$ visiting the same COI, the procedure mentioned above is repeated to calculate the distances $(d_1, d_2, d_3, ..., d_n)$ and predecessors from every source locations to all the POIs (P_i) of the first COI (C_1) intended to be visited by the group members. Then, the calculated distances and the corresponding predecessors for all the POIs of C_1 are stored for future use.

At the beginning of the journey, the number of source locations is exactly the same (as they are assumed to be distinct) as the number of members starting for the first COI C_1 as each member starts their journey from a location (S_i). After reaching the chosen POI (P_1) for C_1 all the current members are now considered virtually as a single-member and their current location is considered as a single-source (S_i) for the next desired COI.

3.3.2 Step-2: Aggregate Distance Calculation

Using the calculated distances from step-1, the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ for every P_i of the first COI (C_1) are calculated in adding the distances from the source locations of the group members in S to P_i. Finally, the minimum value from the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ is determined and the corresponding P_i for the minimum value is chosen as the desired POI of C_1.

$$\text{Aggregate\_Dist}[P_i] = \text{Distance}[S_1 \rightarrow P_i] + \text{Distance}[S_2 \rightarrow P_i] + ... + \text{Distance}[S_n \rightarrow P_i] \tag{3.4}$$

For all subsequent COIs the same procedure (i.e. step-1 and step-2) is used for determining onward travel. While searching for any COI after C_1 the current starting location of the current group members is considered as a single source location because as mentioned earlier the group will be virtually considered as a single member. However, some additional members may join the journey at any given time. Between any pair C_i and C_{i+1} of COIs:

- If no new member wants to join to the trip, the next portion of the trip will proceed by considering only one source location (S_1) and the aggregate distance will be equal to the minimum value of Distance$[S_1 \rightarrow P_i]$
- If x new members wish to join the trip at COI C_i then for each P_i, the aggregated distances $(D_1, D_2, D_3, ..., D_n)$ can be calculated in the following way. The set of

starting locations for the x new members is $X=\{s_1, s_2, s_3, ..., s_x\}$.

$$\text{Aggregate\_Dist}[P_i] = \text{Distance}[S_1 \rightarrow P_i] + \text{Distance}[s_1 \rightarrow P_i] + ... + \text{Distance}[s_x \rightarrow P_i] \tag{3.5}$$

After calculation, the minimum value from the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ is determined and the corresponding P_i for the minimum value is chosen as desired POI of C_i.

Since aggregate distance is calculated in this way in DGTP, all the source locations have equal priority when the desired COI is chosen. The size of the existing group at that location is not taken into account, which leads to a P_i being selected which is closer to the locations of the existing group members. However, the total distance travelled by the group members may not be the best in this approach. To overcome this, we propose a small modification in aggregate distance calculation (in the M-DGTP approach) which will be discussed later in this chapter.

3.3.3 Step-3: Final Trip Result Computation

The distance travelled by any individual member is calculated when the member completes their trip by visiting all of their intended P_i and wants to leave for home. When a member wants to leave for home H_i from their respective last POI, say P_n, at any point in the journey, the leaving distance of the member from P_n to H_i is calculated by using the Dijkstra algorithm [9].

i. The distance travelled by the member during their entire trip is calculated by adding distance from their start location (s_i) to last POI (P_n) they visited and the distance from P_n to their respective home H_i.

   Individual_travel_distance[i] = Distance$[s_i \rightarrow P_n]$ + Distance$[P_n \rightarrow H_i]$

   This calculated Individual_travel_distance may only include a subset of the POI set P since the user can join the trip at any time.

ii. The distance travelled by the member from their start location (s_i) to their last visited POI (P_n) is the Individual_trip_distance[i] of the member:

Individual_trip_distance[i] = Distance[$s_i \rightarrow P_n$]

iii. When an individual member finishes their trip, the individual travel distance of the member is added to the total travel distance. Similarly, the individual trip distance of the member is added to trip distance.

Total_Distance = Total_Distance + Individual_travel_distance[i]

Trip_Distance = Trip_Distance + Individual_trip_distance[i]

For the rest of the journey all the remaining COIs are visited by following all the steps from step-1 (Section 3.3.1) until the last POI P_m for all remaining group members is reached.

Trip Completion Calculation

When all the COIs have been visited by the remaining group members, the leaving distances for all remaining group members are calculated by using the Dijkstra algorithm [9]. The cumulative Total_Distance and Trip_Distance is calculated and updated by following the same procedure from the beginning of step-3 (i.e Section 3.3.3). The final value of Total_Distance is calculated which is the the total travel distance of the group at the end of the trip.

At the end of the trip all information such as total distance, trip distance, the chosen list of POIs for the m COIs (i.e $P_1, P_2, P_3, ..., P_m$) and the travel paths for individual members are retrieved and displayed.

3.4 Modified Dynamic Group Trip Planning (M-DGTP) Approach

To address the issue with DGTP mentioned above, the M-DGTP approach is proposed which optimizes the travel route for the whole trip. The M-DGTP approach is only affected

when new members are added during the journey, which leads to a higher group travel distance than necessary. If all the group members start at the beginning of the journey and no new members want to join throughout the trip, then there is no effect on the travel distance. Similarly, if all the group members start at the beginning and only leave during or at the end of the trip (i.e. no new member joins) then there is also no effect on travel distance.

The Step-1: Shortest Distance and Path Computation procedure and Step-3: Result Computation of M-DGTP is exactly the same as the procedure described in Section 3.3.1 and Section 3.3.3 respectively. Please refer to these sections for more details.

The M-DGTP approach has a different procedure for aggregate distance calculation which is discussed below.

3.4.1 Step-2: Aggregate Distance Calculation

Using the calculated distances from step-1, the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ for every P_i of first COI (C_1) are calculated by adding the distances from the source locations of the group members (say, n source locations for n members) to P_i multiplied by respective number of members at that source location. Finally, the minimum value from the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ is determined and the corresponding P_i for the minimum value is chosen as the desired POI. If member_count$[S_i]$ is the number of member present at source S_i then the equation can be expressed as:

$$\begin{aligned}\text{Aggregate\_Dist}[P_i] = \text{Distance}[S_1 \rightarrow P_i] \times \text{member\_count}[S_1] + \text{Distance}[S_2 \rightarrow P_i] \\ \times \text{member\_count}[S_2] + ... + \text{Distance}[S_n \rightarrow P_i] \times \text{member\_count}[S_n] \quad (3.6)\end{aligned}$$

If no new member joins at this point, the P_i of the next C_i is determined by following the same procedure. Otherwise:

- If x new members join the trip at C_i, then for each P_i, S_1 is the location where the exist-

ing group remains at present and member_count[S_1] is the number of group members currently at S_1, and X=$\{s_1, s_2, s_3, ..., s_x\}$ is the starting locations of the x members that are joining the trip. The aggregated distances $(D_1, D_2, D_3, ..., D_n)$ can be calculated by the following:

$$\text{Aggregate\_Dist}[P_i] = \text{Distance}[S_1 \rightarrow P_i] \times \text{member\_count}[S_1] + \text{Distance}[s_1 \rightarrow P_i] \times \text{member\_count}[s_1] + ... + \text{Distance}[s_x \rightarrow P_i] \times \text{member\_count}[s_x] \quad (3.7)$$

After calculation, the minimum value from the aggregate distances $(D_1, D_2, D_3, ..., D_n)$ is determined and the corresponding P_i for the minimum value is chosen as desired POI of C_i.

If the aggregate distance is calculated considering the actual number of members remaining in a source location, the source location where the maximum number of members remain is given priority in choosing the desired COI over the locations of the other members. This leads to a P_i being selected which is closest in terms of distance traversed which results in minimizing the total travel distance of all the members.

3.5 Running Example

Assume four friends want to visit four COIs. The first and second friend want to visit all the COIs. The third friend wants to leave after visiting the third COI and the fourth friend wants to join at third COI. The following information represents the trip information in detail:

Number of members in the beginning, $n = 3$

Set of start locations, $s = \{s_1, s_2, s_3\}$ is $\{10, 2, 7\}$

Set of home locations, $H = \{H_1, H_2, H_3\}$ is $\{298, 7, 3\}$

The number of COIs, $m = 4$

Sequence of COIs to be provided during the trip, $C = \{C_1, C_2, C_3, C_4\}$ denoted by {Red, Green, Blue, Pink} in Figure 3.3

The number of members want to join, $x = 1$

Source, home and joining COI location $\{s_4, H_4, C_3\}$ of new member is {4, 265, Blue}

The number of members that want to leave before the end of the trip, $y = 1$

Source, home and leaving COI location $\{s_3, H_3, C_3\}$ of departing member is {7, 3, Blue}

In this section, we present an example of both DGTP and M-DGTP using the problem scenario mentioned above. The example is explained using the sample road network of Figure 3.3. The contents of the sample road network have been explained previously in Section 3.1.2. For the sake of example demonstration, the sample road network figure from Section 3.1.2 is placed again here.

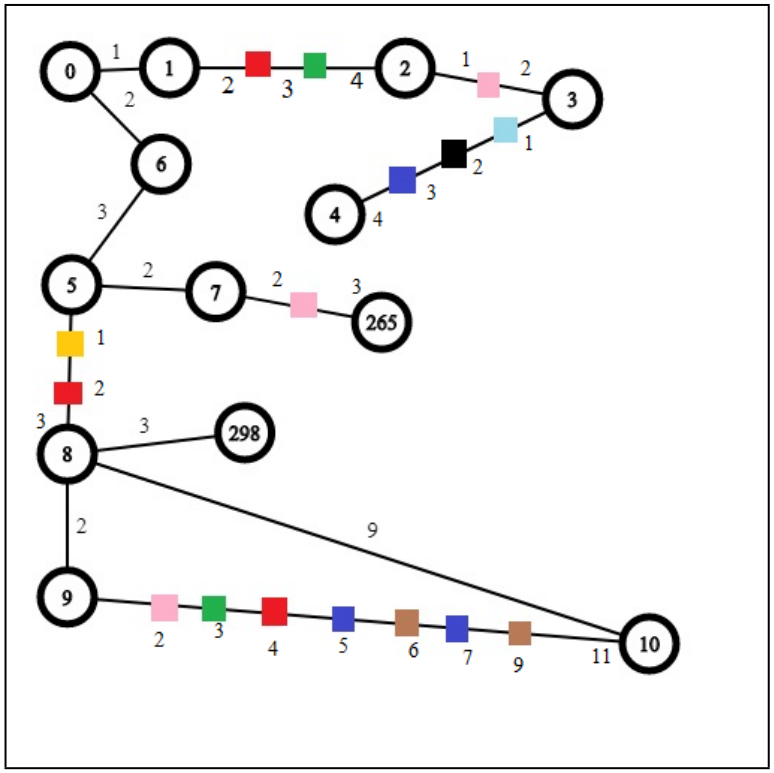

Figure 3.3: A sample road network with POIs

At the beginning, three members with the set of starting vertices s={10, 2, 7} and the set of home vertices H={298, 7, 3} respectively intend to visit one of the three POIs of C_1 which is represented by red coloured square boxes in Figure 3.3. Each POI P_i of C_1 is represented as P_i (u_i, v_i, d_i, wt_i) where, u_i and v_i is the start and end vertex of the hosting

edge respectively and the distance from u to the POI is d_i and wt_i is the edge (u_i, v_i) length. Using the same representation the POIs of C_1 (red) can be represented as: P_1 (1, 2, 2, 4), P_2 (5, 8, 2, 3) and P_3 (9, 10, 4, 11). Using the Dijkstra [9] algorithm the distances from the starting location s_i to u_i and v_i for each P_i are calculated using the procedures Distance[$s_i \rightarrow u_i$] + d_i and Distance[$s_i \rightarrow v_i$] + (wt_i - d_i) respectively and the minimum distance between them is calculated.

Repeating the same procedure the distances of all three POIs of C_1 are calculated from the three starting vertices $\{10, 2, 7\}$. The aggregate distances for all P_i are calculated using both the DGTP and M-DGTP approaches. From calculation of Table 3.2, it is seen that for both DGTP and M-DGTP the minimum value of aggregate distance is 26 and POI P_2 is chosen for COI C_1.

Table 3.2: Shortest distance and aggregate distance calculation for C_1

Start vertex	Member count	Distance to P_1 (1, 2, 2, 4)	Distance to P_2 (5, 8, 2, 3)	Distance to P_3 (9, 10, 4, 11)
10	1	20	10	7
2	1	2	12	19
7	1	10	4	11
Aggregate distance DGTP		$20+2+10=32$	$10+12+4=26$	$7+19+11=37$
Aggregate distance M-DGTP		$20 \times 1 + 2 \times 1+10\times1=32$	$10\times1+12\times1+4\times1=26$	$7 \times 1 + 19 \times 1+11\times1=37$

The same procedure will be repeated while searching the next COI (green) C_2 considering the previously chosen P_2 (5, 8 , 2, 3) of C_1 as the only source location. From the calculation in Table 3.3, it is seen that for DGTP and M-DGTP the minimum value aggregate distances are 6 and 18 respectively for POI P_2. So, P_2 is chosen as the desired POI for COI C_2. The total and trip distances travelled by each members are also calculated and preserved.

A new member joins at COI (blue) C_3 having source and home vertices at 4 and 265 respectively. The previously chosen P_2 (9, 10, 3, 11) of C_2 which represents the existing

Table 3.3: Shortest distance and aggregate distance calculation for C_2

Source vertex	Member count	Distance to P_1 (1, 2, 3, 4)	Distance to P_2 (9, 10, 3, 11)
(5, 8)	3	11	6
Aggregate distance DGTP		11	6
Aggregate distance M-DGTP		$11 \times 3 = 33$	$6 \times 3 = 18$

group, and 4 for the new member are the new source locations for C_3. From the calculation of Table 3.4, for the DGTP approach P_1 will be chosen as the desired COI for C_3 having a minimum aggregate distance of 24. For M-DGTP P_2 will be chosen for C_3 having a minimum aggregate distance of 32. So, after leaving C_2 the travel paths for DGTP and M-DGTP will be different. The distances and path travelled by the members is also calculated, updated and preserved.

Table 3.4: Shortest distance and aggregate distance calculation for C_3

Start vertex	Member count	Distance to P_1 (3, 4, 3, 4)	Distance to P_2 (9, 10, 5, 11)	Distance to P_3 (9, 10, 7, 11)
(9,10)	3	23	2	4
4	1	1	26	28
Aggregate distance DGTP		$23+1=24$	$2+26=28$	$4+28=32$
Aggregate distance M-DGTP		$23 \times 3 + 1 \times 1 = 70$	$2 \times 3 + 26 \times 1 = 32$	$4 \times 3 + 28 \times 1 = 40$

After visiting C_3 a member $\{s_3, H_3\}$, having start (s_3) and home (H_3) vertices 7 and 3 respectively, wants to leave. As the distances and paths for both DGTP and M-DGTP change after C_2 the distance travelled by the member to their home will be different for both the DGTP and M-DGTP approaches.

For DGTP, the travel distance of $\{s_3, H_3\}$ member from vertex 7 to POI P_1 of C_3 via the chosen POIs for C_1 and C_2 is $4+6+23=33$ and the home distance from P_1 (3,4,3,4) to vertex 3 using the Dijkstra[9] algorithm is 3. Therefore, for the departing member individual travel and total distances are 33 and $33+3=36$ respectively.

For M-DGTP, the travel distance of third member from vertex 7 to POI P_2 of C_3 via the chosen POIs for C_1 and C_2 is $4+6+2=12$ and the distance from P_2 (9,10,5,11) to vertex

3 using the Dijkstra[9] algorithm is 22. Therefore, for the third member their individual travel and total distances in M-DGTP approach are 33 and $12+22=34$ respectively.

Table 3.5: DGTP: Shortest distance and aggregate distance calculation for C_4

Start vertex	Member count	Distance to P_1 (2, 3, 1, 2)	Distance to P_2 (7, 265, 2, 3)	Distance to P_3(9, 10, 2, 11)
(3,4)	3	4	19	22
Aggregate distance DGTP		4	19	22

The remaining three group members $\{s_1,H_1\}$, $\{s_2,H_2\}$, $\{s_4,H_4\}$ will travel from C_3 to C_4 denoted by pink colored square in Figure 3.3. From Table 3.5 for the DGTP approach, the previously chosen P_1 (3, 4, 3, 4) of C_3 is considered as the new source for traveling to C_4. For C_4, POI P_1 (2, 3, 1, 2) is chosen as it has the minimum value for the aggregate distance when using the DGTP approach and the value is 4. Like before, the distances and path travelled by each member is also calculated, updated and preserved.

Table 3.6: M-DGTP: Shortest distance and aggregate distance calculation for C_4

Start vertex	Member count	Distance to P_1 (2, 3, 1, 2)	Distance to P_2 (7, 265, 2, 3)	Distance to P_3 (9, 10, 2, 11)
(9,10)	3	21	14	3
Aggregate distance M-DGTP		$21\times3=63$	$14\times3=42$	$3\times3=9$

From Table 3.6 for the M-DGTP approach, the previously chosen P_2 (9,10,5,11) for C_3 is considered as the new source for C_4. For C_4 the value of the minimum aggregate distance is 9 which is found for P_3. Therefore, P_3 (9,10,2,11) is chosen as the desired POI for C_4. For M-DGTP, the distances and path travelled by each members is also calculated, updated and preserved.

As C_4 is chosen by the group to be the last intended COI, the remaining members $\{s_1,H_1\}$, $\{s_2,H_2\}$, $\{s_4,H_4\}$ leave for their respective homes. For the DGTP approach, using the Dijkstra[9] algorithm, the home distances of the remaining members are calculated as

Table 3.7: Result computation of DGTP approach

Start vertex	Home vertex	Sequence of P_i for each C_i from start to home	Distance from source to last C_i	Distance from last C_i to home	Individual total distance
10	298	$10 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_1] \rightarrow C_4[P_1] \rightarrow 298$	$10+6+23+4=43$	17	60
2	7	$2 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_1] \rightarrow C_4[P_1] \rightarrow 7$	$12+6+23+4=45$	13	58
7	3	$7 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_1] \rightarrow 3$	$4+6+23=33$	3	36
4	265	$4 \rightarrow C_3[P_1] \rightarrow C_4[P_1] \rightarrow 265$	$1+4=5$	16	21
DGTP approach Trip distance			$43+45+33+5=126$		
DGTP approach Total distance			$60+58+36+21=175$		

17,13 and 16 respectively. For the M-DGTP approach, the calculated home distances from the last C_i is 7, 9, 12 respectively.

Table 3.7 shows the result computation for the DGTP approach. The value of the trip distance is obtained by adding the distances from the source to the last C_i for each member. The value of the individual total distance is obtained by adding the home distance to individual trip distances for each member. The total and trip distance for the entire group is obtained through summing the individual total and trip distances. For the DGTP approach, the obtained value of the trip and total distances are 126 and 175 respectively.

Table 3.8 shows the result computation for the M-DGTP approach. Following the same procedure as before the trip and total distances are calculated. For the M-DGTP approach, the obtained value for the trip and total distances are 85 and 135 respectively. From the result, the running example also shows that M-DGTP approach reduces the value of trip and total distance in comparison to the DGTP approach.

Table 3.8: Result computation of M-DGTP approach

Start vertex	Home vertex	Sequence of P_i for each C_i from start to home	Distance from source to last C_i	Distance from last C_i to home	Individual total distance
10	298	$10 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_2] \rightarrow C_4[P_3] \rightarrow 298$	$10+6+2+3=21$	7	28
2	7	$2 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_2] \rightarrow C_4[P_3] \rightarrow 7$	$12+6+2+3=23$	9	32
7	3	$7 \rightarrow C_1[P_2] \rightarrow C_2[P_2] \rightarrow C_3[P_2] \rightarrow 3$	$4+6+2=12$	22	34
4	265	$4 \rightarrow C_3[P_2] \rightarrow C_4[P_3] \rightarrow 265$	$26+3=29$	12	41
M-DGTP approach Trip distance			$21+23+12+29=85$		
M-DGTP approach Total distance			$28+32+34+41=135$		

3.6 Summary

In this chapter, procedures of our two proposed approaches: DGTP and M-DGTP are discussed. A running example for both the strategies is also presented. The next chapter presents the performance evaluation of our proposed strategies.

Chapter 4

Experiments and Evaluations

In this chapter we discuss the conducted experiments and the generated results from the experiments. The performance of the proposed DGTP and M-DGTP algorithms is compared with an adapted version of the NaiveDGTP algorithm that is recently proposed by Tabassum et al. [33]. Extensive experiments have been carried out using the real-world road network of California [19] [2].

This chapter is organized as follows: Section 4.1 presents a summary of the NaiveDGTP algorithm and our adaptations to it. Section 4.2 presents the experimental setup used for conducting experiments. Section 4.3 presents on the performance metrics considered for comparison. Section 4.4 and Section 4.5 presents a review of the generated results for static and dynamic groups respectively. Section 4.6 presents an overall discussion of all the conducted experiments.

4.1 Adapted Naive Dynamic Programming Approach

The NaiveDGTP [33] algorithm was proposed for Euclidean Space whereas our proposed DGTP algorithms has been implemented using the spatial network distance in a real road network. Our proposed DGTP, M-DGTP and adapted N-DGTP are compared considering both static and dynamic group scenarios. The static group scenario assumes no group members join or depart during the trip and the dynamic group scenario assumes group members can join or depart at any point of the trip. Our proposed DGTP and M-DGTP

[2] https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm

approaches do not require to pre-specify the visiting COIs before the start of the trip. That means the trip can be changed at any point of the trip.

For the sake of comparison, an adaptation of the NaiveDGTP algorithm which we call N-DGTP has been implemented for a real road network. Like NaiveDGTP, the N-DGTP algorithm uses dynamic programming. For a given pre-specified list of COIs and (start, end) position of each person that starts the trip from the very beginning, the N-DGTP algorithm is implemented as:

- **Initial optimal trip computation:**

  i. Retrieval of all POIs of the COIs that the group intends to visit,

  ii. The dynamic programming algorithm is applied to computing an initial optimal trip. It works as follows:

  (a) For COI C_1, the partial aggregate distances are calculated as: the sum of distances from all the locations of members who want to join at C_1 to each POI of C_1 and distance(s) from all the POIs of P_1 to all the destination of member(s) who want to leave from C_1. The calculated partial aggregate distances are stored for future use.

  (b) For all COIs after C_1, the partial aggregate distances for all the POIs are calculated as: the sum of the partial sums at P_{i-1}, the distances from P_{i-1} to P_i multiplied by the number of users who continue the trip from C_{i-1} to C_i, the distances from source to all POIs of P_i who join at C_i, and the distance from all the POIs of P_i to destination who want to leave from C_i.

  (c) The minimum total aggregate distance after processing the last COI is identified and the corresponding trip is the computed initial trip.

- **Process path:** The path is re-calculated if any trip change occur in real-time (i.e. addition and departure of new group members). The path re-calculation is performed as follows:

i. The first POI on the path is visited,

ii. The starting point of existing group members is considered as the last POI they visited,

iii. If there is any departure of group member(s) the distance to destination is calculated,

iv. If there is any new arrival of additional group member(s) it is added to the the set of group members,

v. The changes in the group is updated and remaining path is re-calculated using the dynamic programming algorithm.

4.2 Experimental Setup

The proposed algorithm has been implemented using the C++ programming language. All the experiments were carried out on a Intel® Core™2 Duo CPU E6750 with 2.66GHz CPU and 1835 MB RAM which is powered by CentOs Linux 7 operating system. The experimental settings that have been used to conduct the experiments are grouped into two categories:

- Static group
- Dynamic group

Section 4.2.1 discuss the California [19] dataset that is used for conducting experiments. Section 4.2.2 and Section 4.2.3 describe the setups that have been used for the static and dynamic groups respectively.

4.2.1 Data Sets

Table 4.1 shows the metadata of the California [19] road network. As most of the POIs lie on the edges of the network, there are many more POIs than vertices in the network. For our experiments, the COIs having less than 1000 POIs have been utilized. There are 45

COIs with this range of POIs, with a total of 10,948 POIs. Table 4.2 summarizes six of the selected categories which have been used for experiments.

Table 4.1: California road network summary

Title	Information
Network (Notation)	California (CA)
Number of vertices / edges	21,048 / 21,693
Number of COIs	62
Number of POIs	87,635

Table 4.2: Categories used in experiment

Category	POIs
Airport	928
Hospital	824
Tower	899
Bar	218
Trail	887
Cemetery	788

4.2.2 Static Group Experimental Setup

The static group scenario assumes no group members begin or depart during the trip (in other words, they begin at the start of the trip and depart at the end). The evaluation for static group has been considered by varying three parameters:

- Number of points in dataset,
- Number of COIs (i.e. trip size),
- Number of group members (i.e. Group size).

For each set of experiments, the value of one parameter is varied while fixing other parameters to their default values. Table 4.3 summarizes the values used for each parameter to conduct the experiments and their default values. For experiment, the number of points are varied in a range from 20%-100% where 20% means first 20% of the number of vertices

Table 4.3: Static group parameters

Parameter	Range	Default
Number of points	20%, 40%, 60%, 80%, 100%	20%
Number of COIs	2, 3, 4, 5	3
Group size	2, 4, 8, 16, 32	5

and associated information with them. The total number of vertices in the California [19] dataset is 21,048. So, the first 20% of the number of points contain approximately 4,200 vertices. Similarly, 40% of the number of points contain approximately 8,400 vertices. In Table 4.3 the 100% number of points means the complete California [19] dataset.

4.2.3 Dynamic Group Experimental Setup

The dynamic group scenario assumes that the group size is changing throughout a trip. It is assumed that, the change in group size can occur by any one of the three incidents:

- Adding new participant to the trip,
- Having participants depart from the trip,
- Random addition and departure of group members.

The four scenarios used for evaluating the addition of new members during a trip assumes that there are a total of 16 members in the group and the group members travel to five different COIs. The four groups Group-1, Group-2, Group-3 and Group-4 present how the participants are added in each trip (i.e. Group Trip-1, Group Trip-2, Group Trip-3, Group Trip-4). For example, in Table 4.4 Group-1 presents the scenario of Group Trip-1 where, the number of people join at COI1, COI2, COI3, COI4 and COI5 are 4, 4, 4, 4, and 0 respectively. For Group-1, equal numbers of members are added to the first four COIs. For Group-2, half of the total group members (i.e. 8) are added to the first COI and rest of the 8 members are equally added to the remaining four COIs. For Group-3, the number of people added to each COIs are in descending order and for Group-4, the number of people added to each COIs are in ascending order. In each case, we use 20% of our whole California [19]

Table 4.4: Dynamic group parameters

COIs	Adding new participants				Dataset
	Group-1	Group-2	Group-3	Group-4	
COI1	4	8	6	1	20%
COI2	4	2	4	2	
COI3	4	2	3	3	
COI4	4	2	2	4	
COI5	0	2	1	6	

Table 4.5: Dynamic group parameters

COIs	Having participants depart				Dataset
	Group-1	Group-2	Group-3	Group-4	
COI1	0	2	6	1	20%
COI2	4	2	4	2	
COI3	4	2	3	3	
COI4	4	2	2	4	
COI5	4	8	1	6	

dataset to run our experiments. Table 4.4 summarize how the group members are added during the trip.

Similarly, four scenarios are used for evaluating the departure of participants from a trip. They also assume that there are a total of 16 members in the group and the group members travel to five different COIs. For Group-1, no member departs at first COI and equal number of members depart from the remaining COIs. For Group-2, half of the total group members equally depart from the first four COIs and remaining half of the group members depart from the last COI. For Group-3, the number of people depart from each COIs are in descending order and for Group-4, the number of people depart from each COIs are in ascending order. Table 4.5 shows the combinations for the four group trips and summarizes how the group members depart during the trip.

The four scenarios used for evaluating random addition and departure of group members also assumes that there are a total of 16 members starting the trip. The group members travel to five different COIs and at the end of the trip the group size also remains at 16. For

Table 4.6: Dynamic group parameters

COIs	Addition and departure of participants				Dataset
	Group-1	Group-2	Group-3	Group-4	
Leave after COI1	4	8	3	5	20%
Join at COI2	4	4	6	1	
Leave after COI2	4	0	3	5	
Join at COI3	4	4	4	5	
Leave after COI3	4	8	4	5	
Join at COI4	4	4	3	5	
Leave after COI4	4	0	6	1	
Join at COI5	4	4	3	5	

Group-1, equal numbers of members join and depart at each COIs. For Group-2, Group-3 and Group-4 the number of members join or leave at each COIs are chosen randomly. Table 4.6 summarizes how the group members are added and departed during the trips.

4.3 Performance Metrics

The proposed three strategies: Dynamic Group Trip Planning (DGTP), Modified Dynamic Group Trip Planning (M-DGTP) and Naive Dynamic Group Trip Planning (N-DGTP) are compared in terms of three performance metrics:

- **Total distance:** The total distance is the sum of the distances for each members, starting from a source location passing at least one POI from each COIs to their respective destination locations.

- **Trip distance:** The trip distance is calculated by deducting the home distance from total distance. Where, Home distance is the sum of distances of each group member from the last visited POI to their respective destination locations.

  Trip distance = Total distance - Home distance

  As the starting and departing locations and COIs of the members are randomly located for a trip, a scenario can arise where the trip distance of members is significantly shorter than the destination or home distance of the members. This is why to

keep the track of the original trip distance (from start to last visited P_i) with respect to total distance travelled by the members we calculate the trip distance.

- **Processing time:** The processing time is measured in seconds and is the time required to generate the trip result for the group.

4.4 Static Group Results

The performance of DGTP, M-DGTP and N-DGTP on static group is compared in terms of total distance, trip distance and processing time. The effect of number of points, number of COIs and group size is observed. Section 4.4.1 discusses the effect of number of POIs in dataset, Section 4.4.2 discusses the effect of number of COIs, and effect of group size is discussed in Section 4.4.3.

4.4.1 Effect of Number of Points in Dataset

Table 4.7 presents the comparison of the DGTP, M-DGTP and N-DGTP approaches in terms of processing time (in sec.) for a varying number of POIs in dataset. Here, the number of POIs are expressed as a percentage of number of vertices in the California [19] road network. All three approaches incur higher processing times with the increase of the number of POIs in dataset, as expected. The experimental results also illustrate that the processing time of DGTP and M-DGTP is almost the same, whereas N-DGTP shows the worst processing time. In each case, it is observed that DGTP and M-DGTP is 26, 78, 208, 249 and 221 times faster than N-DGTP in respect to the increase of number of POIs.

Table 4.7: Processing time in sec.

Number of Points	DGTP	M-DGTP	N-DGTP
20%	0.08	0.08	2.11
40%	0.18	0.18	14.036
60%	0.288	0.286	59.648
80%	0.37	0.368	91.708
100%	0.476	0.474	105.33

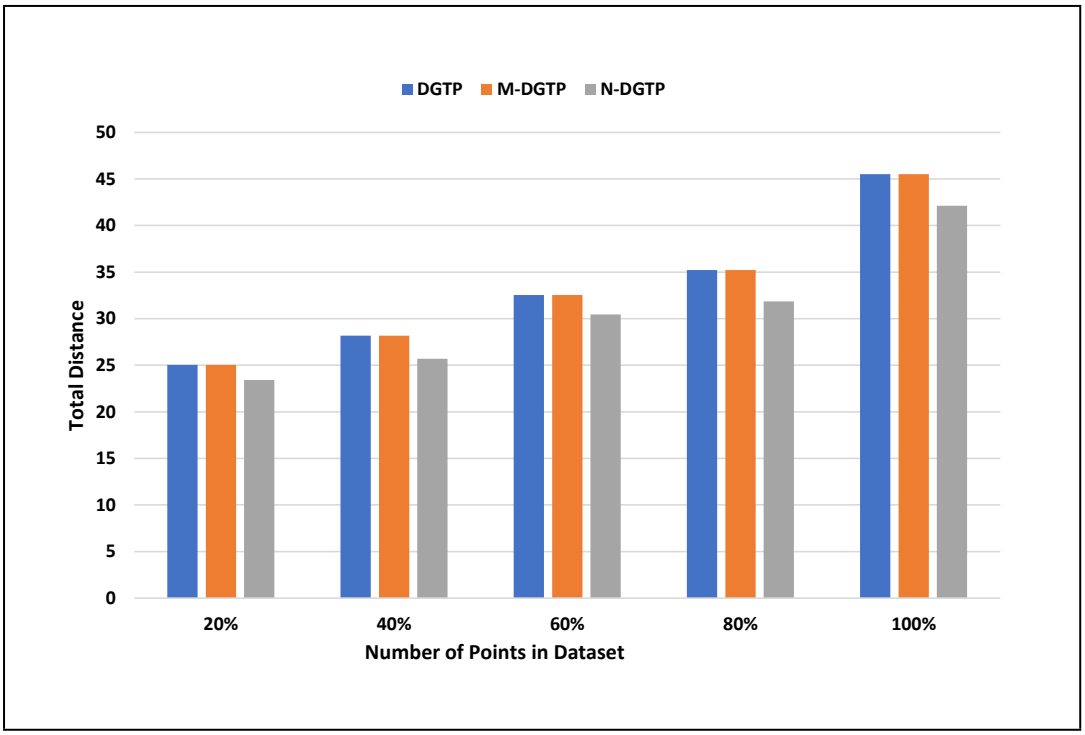

Figure 4.1: Effect of number of POIs on average total distances

This is because, the DGTP and M-DGTP approaches do not re-calculate the trip, when compared to a trip (or partial trip) that needs to be processed for N-DGTP to generate the resultant trip. The N-DGTP approach based on dynamic programming requires a significantly larger number of shortest path computation and partial trip re-calculations with the increase of the number of POIs.

Figure 4.1 shows the effect of the number of POIs on total distance. The results show that DGTP and M-DGTP have exactly the same total distance for each group of varying POIs. This is because, M-DGTP only differs from DGTP in terms of the calculation of aggregate distance for varying addition and departure of group members (discussed in Section 3.4.1). As stated earlier, the static groups considers the situation where all group members start together at the beginning and there is no new addition or departure of group members during the trip. Therefore, the distance is same for both DGTP and M-DGTP approaches.

Figure 4.1 also illustrates that, in each case N-DGTP shows a smaller total distance which is expected as the N-DGTP approach is based on dynamic programming and sup-

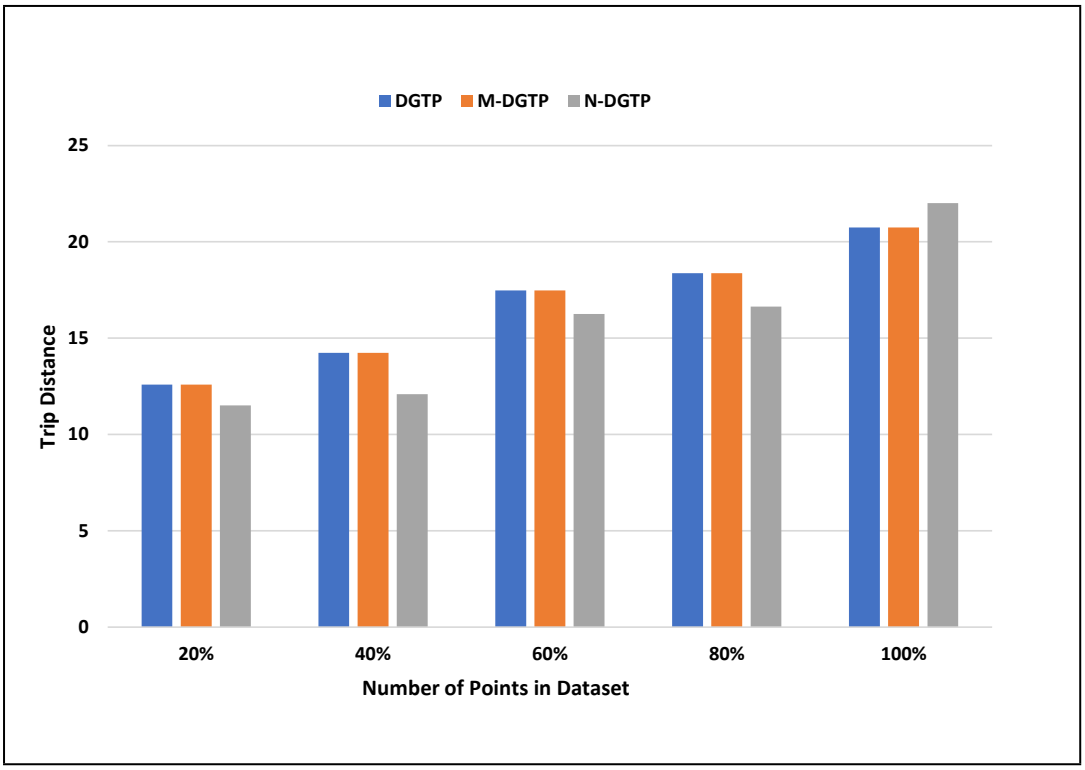

Figure 4.2: Effect of number of POIs on average trip distances

posed to generate the optimal trip result. However, from the experimental results it is seen that DGTP and M-DGTP still being greedy approaches generates on average only an 8% longer trip than N-DGTP in terms of total distance.

Figure 4.2 shows the effect of number of POIs on trip distance and again for the same reason mentioned above both DGTP and M-DGTP shows the exactly same trip distances regardless of the number of POIs. Also N-DGTP generates a shorter trip results than both DGTP and M-DGTP. In case of 100% points in dataset N-DGTP shows an exception generating a little longer trip than both DGTP and M-DGTP approaches. The reason behind this may be: as the DGTP and M-DGTP approaches do not incorporate home or destination distances to optimise the route, there might be the situation where the home distances of the group members for DGTP and M-DGTP approaches is larger than the trip distance. The trip length generated by DGTP and M-DGTP still being greedy approaches is on average of only 8% longer than those generated by N-DGTP.

4.4.2 Effect of Number of COIs

In this section a discussion on the effect of number of COIs on processing time, total distance and trip distance is presented. Table 4.8 presents the results of DGTP, M-DGTP and N-DGTP in terms of processing time. Again, as expected all the approaches show an increase in processing time with the increase of the number of COIs. In case of N-DGTP, where number of COIs increases from 3 to 4 the required processing time is a little bit smaller in case of 4 COIs than 3 COIs. The reason behind this is as the COIs are randomly selected, there remains possibility of selecting four COIs which has comparatively less number of POIs. Less number of POIs means less search space and less computation time required.

Table 4.8: Processing time in sec.

Number of COIs	DGTP	M-DGTP	N-DGTP
2-COIs	0.0620	0.0620	4.4280
3-COIs	0.0800	0.0800	9.7620
4-COIs	0.1000	0.1020	9.6480
5-COIs	0.1200	0.1200	13.2180

The DGTP and M-DGTP comes out ahead of N-DGTP showing 71, 122, 94 and 110 times faster processing than N-DGTP for increasing number of COIs respectively. This is because with the increase of the number of COIs the number of times that a partial trip needs to be re-calculated for N-DGTP also increases which results in a very expensive computational overhead. Again, DGTP and M-DGTP significantly outperforms N-DGTP in terms of processing time because both of the approaches do not require any partial trip re-calculation.

Figure 4.3 illustrates the effect of the number of COIs on total distance. Again, for the static groups, DGTP and M-DGTP generate the same total distances whereas, N-DGTP generates smaller total distances than both DGTP and M-DGTP. For the lower COIs the total distances generated for DGTP, N-DGTP and M-DGTP are almost the same. With the increase in number of COIs DGTP and M-DGTP still being greedy approaches generate

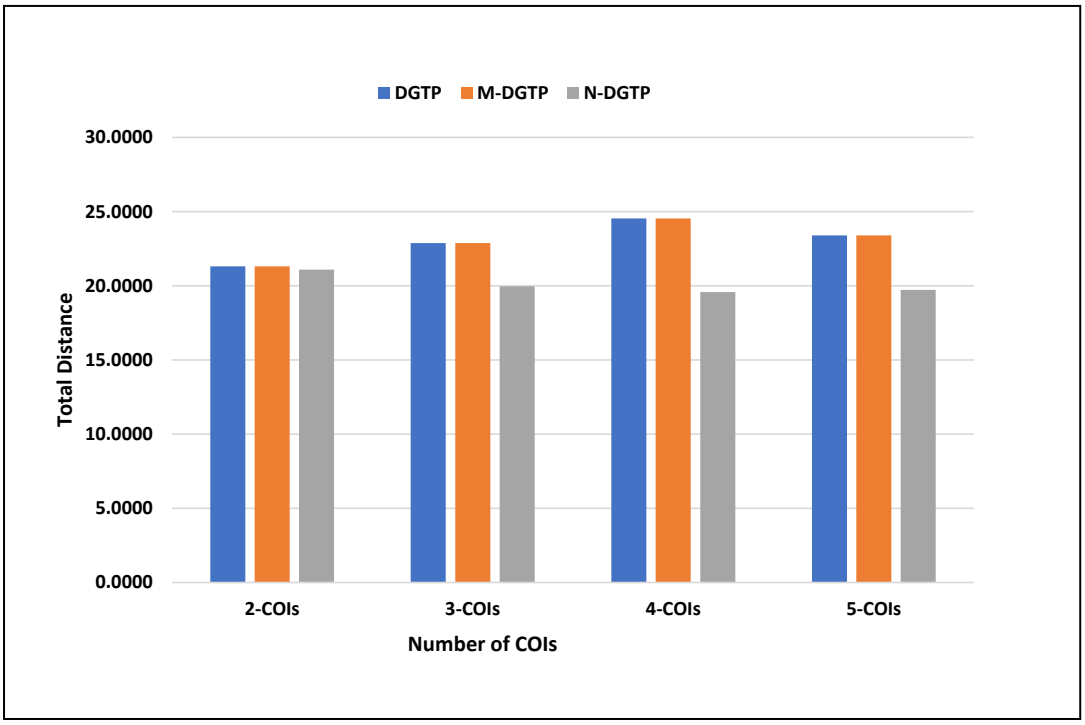

Figure 4.3: Effect of number of COIs on average total distances

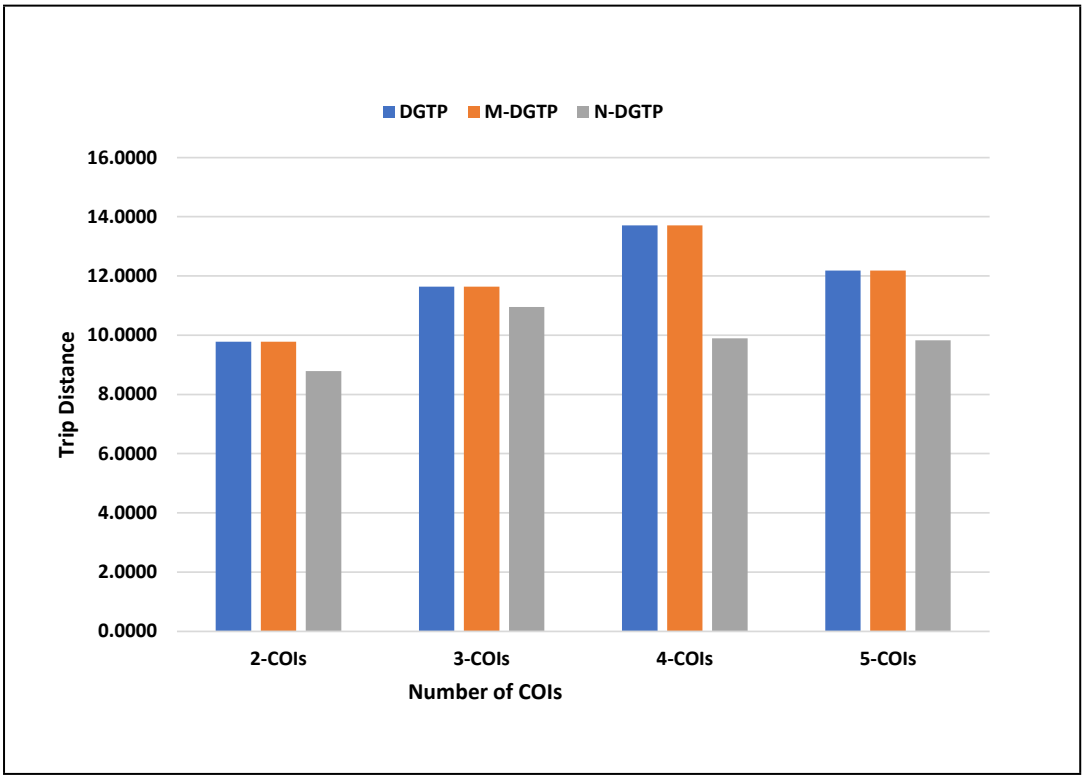

Figure 4.4: Effect of number of COIs on average trip distances

on an average of 15% longer trip result than N-DGTP in terms of total distance. Figure 4.4 shows the effect of the number of COIs on trip distances. Similar to earlier, due to optimality N-DGTP generates smaller trip distances than DGTP and M-DGTP. Both DGTP and M-DGTP still being greedy approaches generates an average of 20% longer trip result than N-DGTP in terms of trip distances where the generated maximum and minimum values of longer trip results are 39% and 6% respectively.

4.4.3 Effect of Group Size

Table 4.9 shows that processing time of all three approaches: DGTP, M-DGTP and N-DGTP increases with the increase of the group size. This is expected because the increase of group size will result in a larger number of shortest path computations which leads to higher processing time.

Table 4.9: Processing time in sec.

Group size	DGTP	M-DGTP	N-DGTP
2-members	0.0600	0.0600	6.0700
4-members	0.0760	0.0740	8.6980
8-members	0.1000	0.1040	11.9720
16-members	0.1600	0.1620	21.2100
32-members	0.2900	0.2960	41.2040

From the results, it is also observed that N-DGTP has the worst processing time which is approximately 120 times slower than DGTP and M-DGTP both of which have almost the same processing time. This is because N-DGTP computes too many shortest paths and trip re-calculation operations than our proposed approaches.

Figure 4.5 shows the effect of group size on the three approaches and as expected with the increase in group size the increase of total distances is an approximate multiple of the number of participants. As explained earlier, for static group DGTP and M-DGTP calculates the same total distances for all group trip combinations.

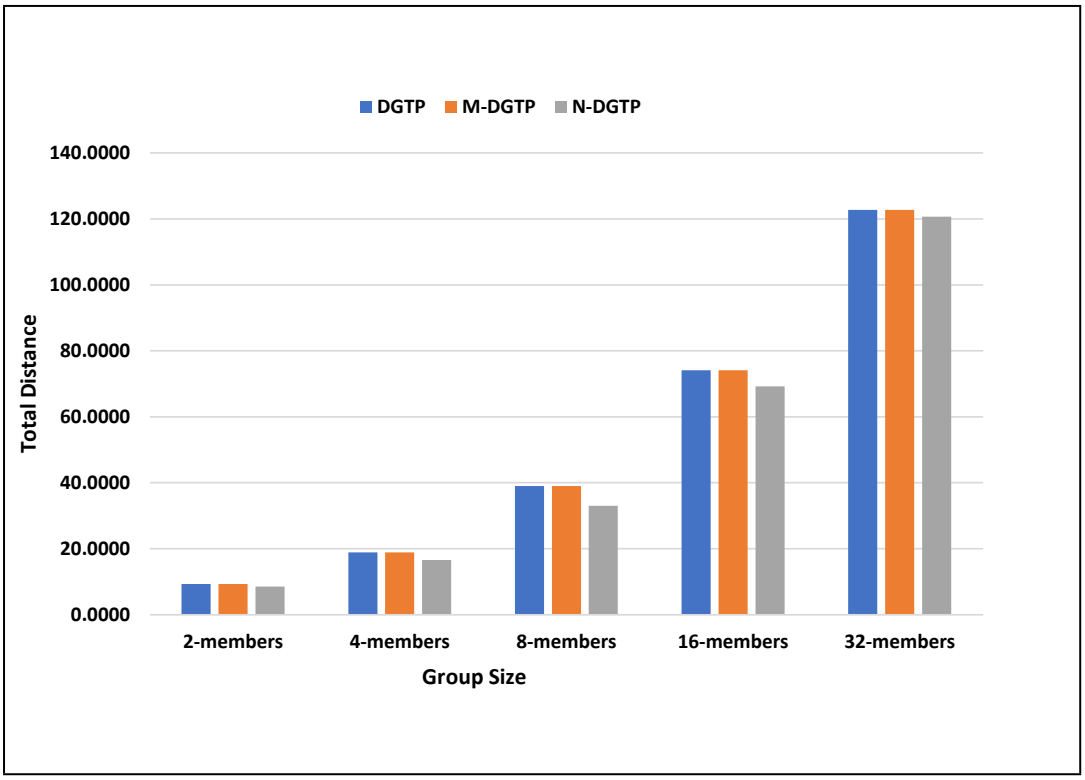

Figure 4.5: Effect of group size on average total distances

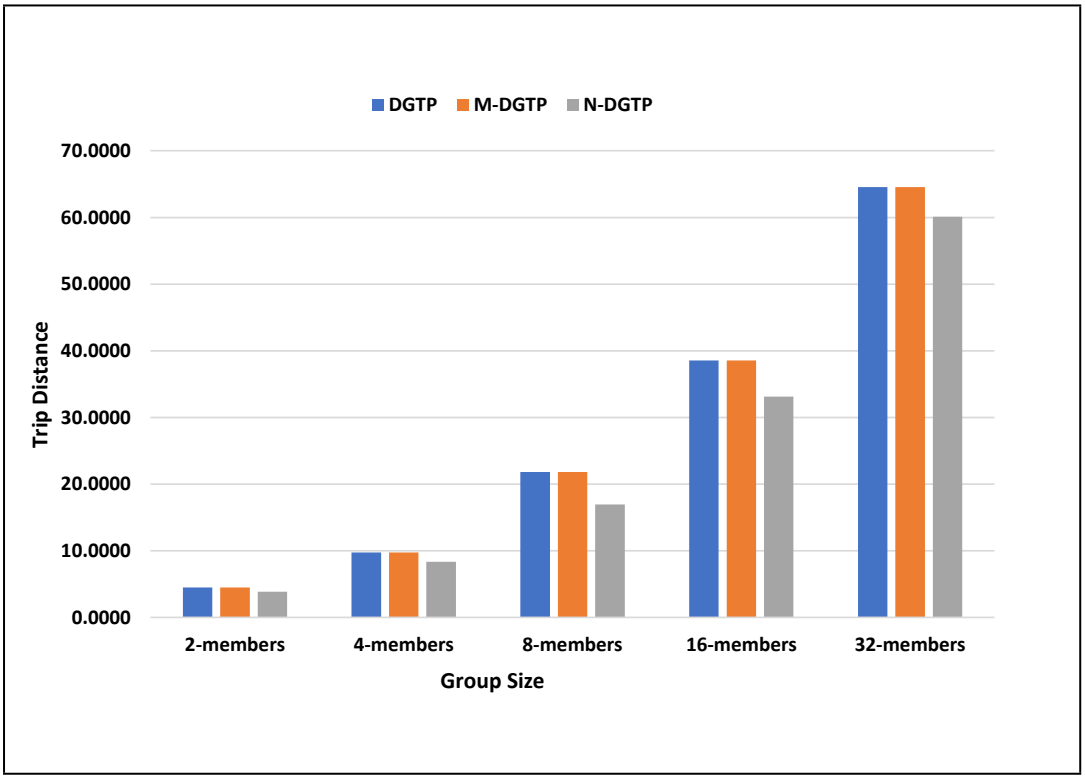

Figure 4.6: Effect of group size on average trip distances

The results also show that, N-DGTP being an optimal approach generates an average of only a 10% shorter trip than DGTP and M-DGTP where the maximum and minimum values of shorter trips generated by N-DGTP approach are 18% and 2% respectively.

Figure 4.6 shows the effect of group size on trip distances and all three approaches show an increase in trip distance with an increase of group size where trip distance is an approximate multiple of the number of participants. It is also expected that N-DGTP generates shorter trip distances than our approaches as it is an optimal approach. But with the increase of group size N-DGTP on average generates only 17% shorter trip than our proposed approaches. The maximum and minimum values of shorter trips generated by N-DGTP is 29% and 7% respectively.

4.5 Dynamic Group Results

This section also compares the performance of DGTP, M-DGTP and N-DGTP in terms of total distance, trip distance and processing time. This time, however, the effect of adding new participants, departing participants and random adding and departing participants on a group trip is studied. Section 4.5.1 discusses the effect of adding new participants, Section 4.5.2 discuss the effect of departing participants from trip and Section 4.5.3 discusses the effect of randomly adding and departing participants.

4.5.1 Effect of Adding New Participants

Table 4.10 presents the effect of adding new participants during a trip using DGTP, M-DGTP and N-DGTP in terms of processing time. To study the effect of adding new participants we consider four group combinations presented earlier in Table 4.4. From the results, it is observed that adding new members to the trip in any combination has almost no effect on processing time for DGTP and M-DGTP.

However, the processing time of N-DGTP varies significantly with different group combinations and on how many steps we are adding our new participants. This is because

Table 4.10: Processing time in sec.

Groups	DGTP	M-DGTP	N-DGTP
Group-1	0.1900	0.1900	14.8980
Group-2	0.1900	0.1900	27.2200
Group-3	0.1900	0.1900	21.4740
Group-4	0.1900	0.1900	10.1140

adding new participants will result in an increase in the search space which leads to more shortest path computations and more trip re-calculation operations. Adding new members to the trip will also increase the search space for DGTP and M-DGTP but this is where we can see a significant difference in processing time with N-DGTP. Our proposed DGTP and M-DGTP perform an average of 98 times faster in terms of processing time.

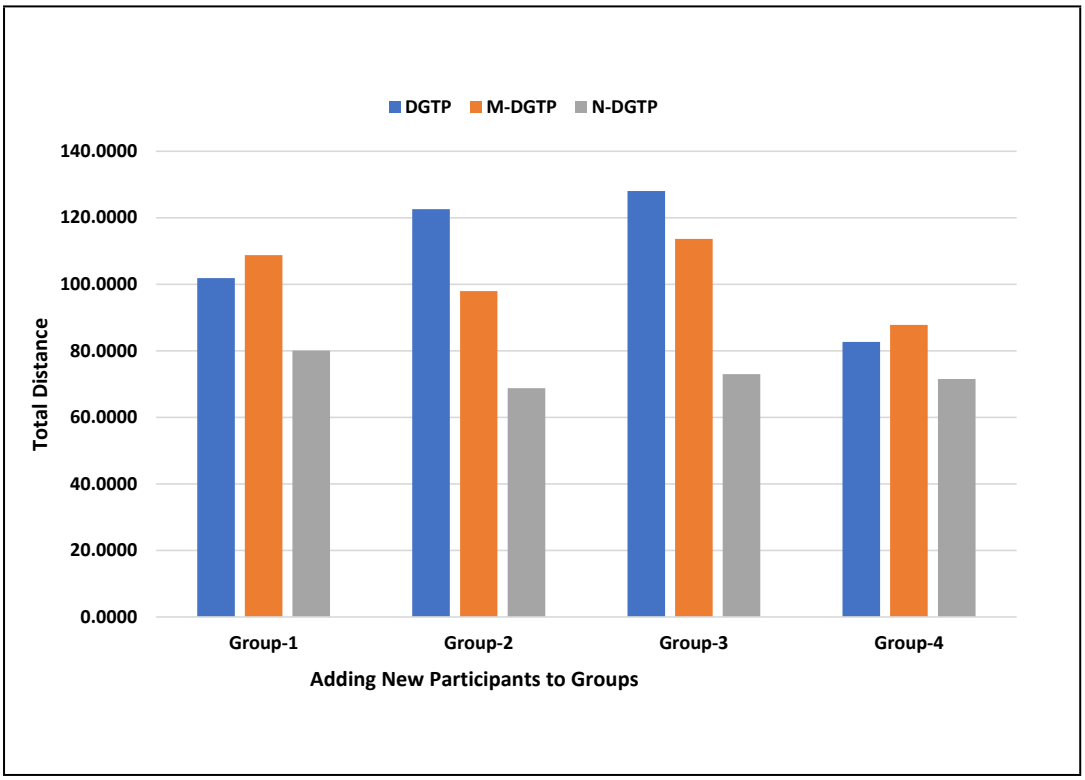

Figure 4.7: Effect of adding new participants on total distance

Figure 4.7 illustrates the effect of adding new participants on the total distance for all the approaches. For Group-1 and Group-4 M-DGTP generates a small longer trip result than DGTP but for Group-2 and Group-3 M-DGTP generates much shorter route than DGTP.

This is because, M-DGTP gives a better trip result for larger number of group member additions on the beginning steps. For Group-2 and Group-3 more people are added to the initial steps and this is why M-DGTP is able to optimize overall trip result for the group trips. N-DGTP generates smaller trip results than both DGTP and M-DGTP because it is an optimal method. M-DGTP stands in second position generating an average of 10% better trip result than DGTP and it occurs specially when more people are added in the initial COIs where the M-DGTP approach consider the actual group size.

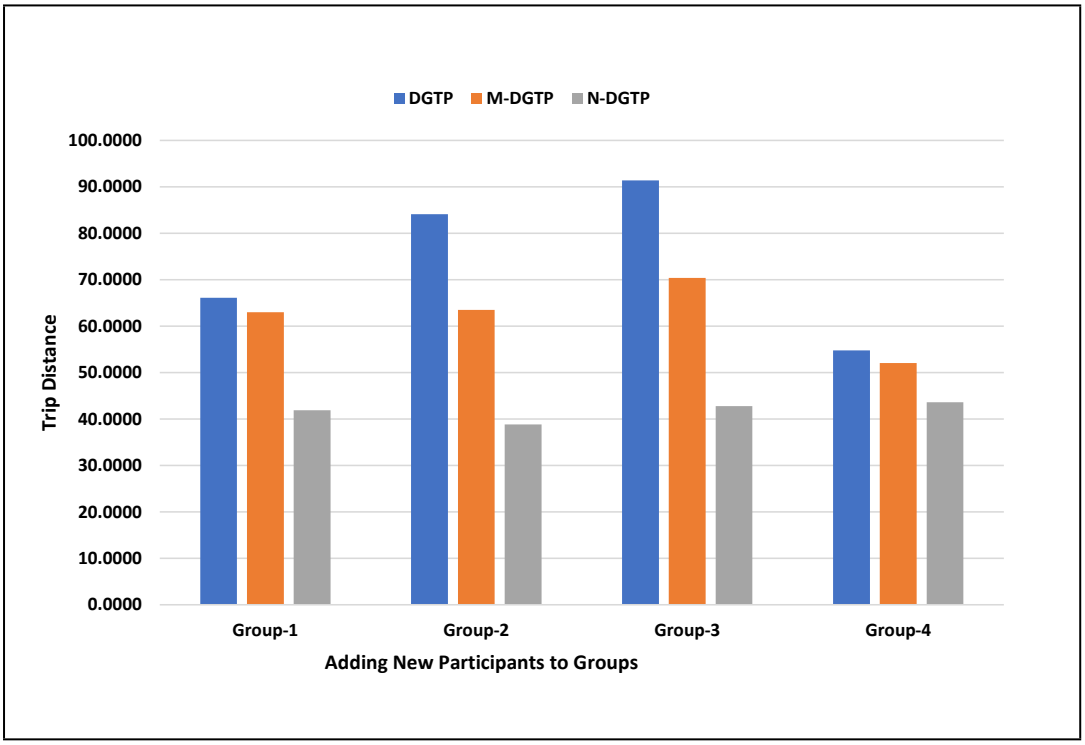

Figure 4.8: Effect of adding new participants on trip distance

Figure 4.8 illustrates the effect of adding new participants on trip distances. The results show that M-DGTP outperforms DGTP for all group combinations by reducing trip distances. N-DGTP generates the best trip result for all group combinations and M-DGTP standing at second place in case of trip distances. For group-2 and group-3 the performance of DGTP is worst because DGTP approach does not consider actual group size in case of aggregate distance calculation (3.3.2). From the results, it is also observed that M-DGTP produces an average of 30% better trip result than DGTP as the M-DGTP approach consid-

ers actual group size in the case of aggregate distance calculation. Thus, it generates better trip result than DGTP.

4.5.2 Effect of Having Participants Depart

Table 4.11 shows the effect of having participants depart on total distances for DGTP, M-DGTP and N-DGTP. From the results, it is seen that DGTP shows a slightly better performance than M-DGTP in terms of processing time and N-DGTP is significantly worse. DGTP shows slightly better processing time because it performs fewer calculations calculations than M-DGTP. N-DGTP shows a very high processing time because it computes a large number of shortest paths and trip re-calculation operations. It is seen that both M-DGTP and N-DGTP performs an average of 90 times better than N-DGTP.

Table 4.11: Processing time in sec.

Groups	DGTP	M-DGTP	N-DGTP
Group-1	0.1880	0.1900	13.3240
Group-2	0.1900	0.1900	19.8820
Group-3	0.1880	0.1900	14.0660
Group-4	0.1900	0.1880	20.1000

Figure 4.9 illustrates the effect of departing participants during the trip in terms of total distance. Both DGTP and M-DGTP show exactly same total distances for all the group combinations because no new members are added during the trip. In other words, only departing participants during the whole trip has no effect on M-DGTP over DGTP. As we have seen earlier, N-DGTP generates smaller trip results than both DGTP and M-DGTP and our proposed DGTP and M-DGTP generates an average of 22% longer trip than N-DGTP in terms of total distance.

Figure 4.10 shows the effect of departing participants on trip distances. As explained before, both the DGTP and M-DGTP approaches generate same trip results and N-DGTP generates smaller trip results than both the approaches.

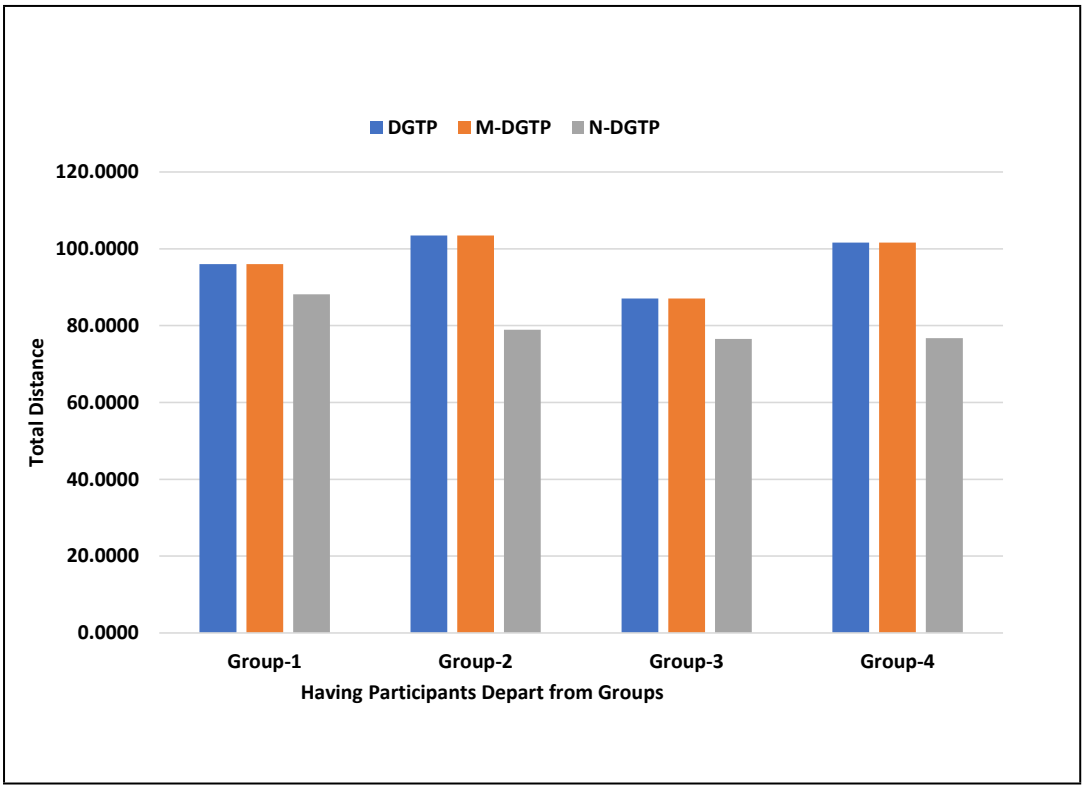

Figure 4.9: Effect of having participants depart on total distance

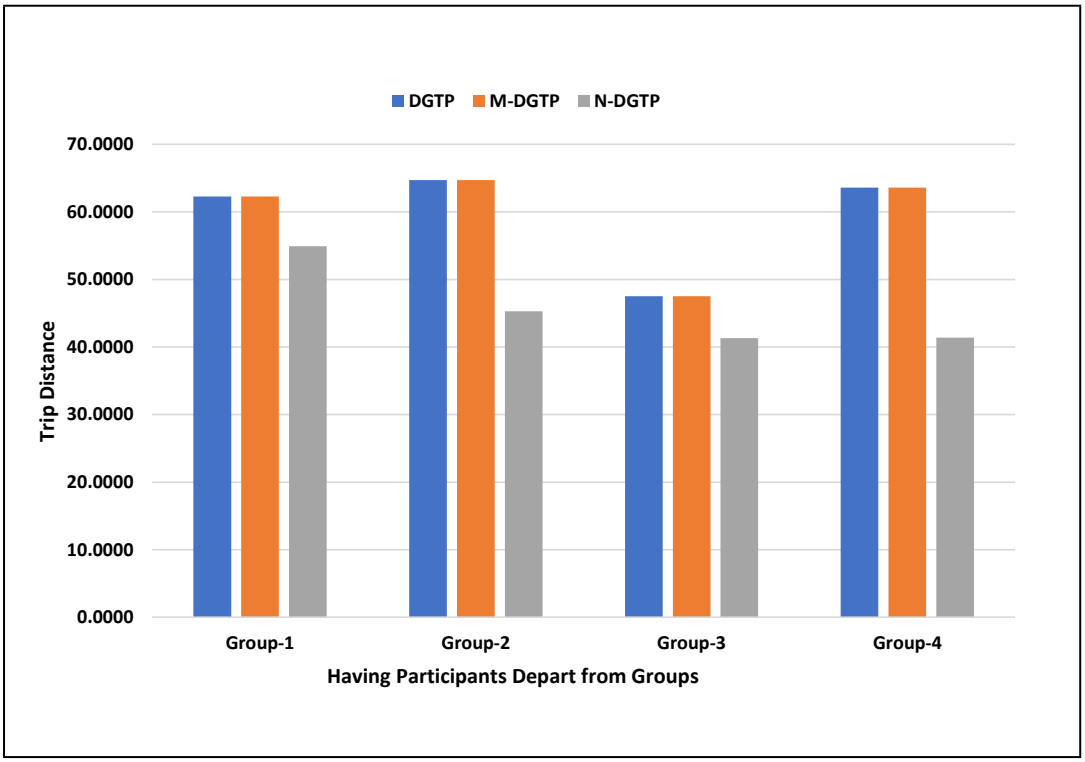

Figure 4.10: Effect of having participants depart on trip distance

N-DGTP generates smaller trip results than both DGTP and M-DGTP because it performs a guided search. N-DGTP computes an initial trip first and the initial trip is recalculated if changes (i.e. during-trip additions and departures) are made. But this strategy, proposed by Tabassum et al [33], is also a drawback for N-DGTP because all COIs must be pre-specified. The order and type of the COIs cannot be changed during the trip. Both of our proposed approaches, DGTP and M-DGTP do not require pre-specified COIs. From the results, DGTP and M-DGTP generates an average of 31% longer trip results than N-DGTP in terms of trip distance.

4.5.3 Effect of Addition and Departure of Participants

Table 4.12 shows the effect of addition and departure of participants on a trip on DGTP, M-DGTP and N-DGTP in terms of processing time. From results, DGTP shows slightly better processing time than M-DGTP because M-DGTP performs a few more operations than DGTP. N-DGTP shows the worst processing time and it performs an average of 274 times slower than both DGTP and M-DGTP.

Table 4.12: Processing time in sec.

Groups	DGTP	M-DGTP	N-DGTP
Group-1	0.3020	0.3060	62.1540
Group-2	0.3060	0.3080	67.7120
Group-3	0.3080	0.3100	116.7400
Group-4	0.3080	0.3100	91.4000

Figure 4.11 illustrates the effect of random addition and departure of participants on DGTP, M-DGTP and N-DGTP in terms of total distance. From the results, M-DGTP shows better trip results than DGTP in terms of total distance for all group combinations. N-DGTP has the smallest trip results than others because it performs guided search as explained before.

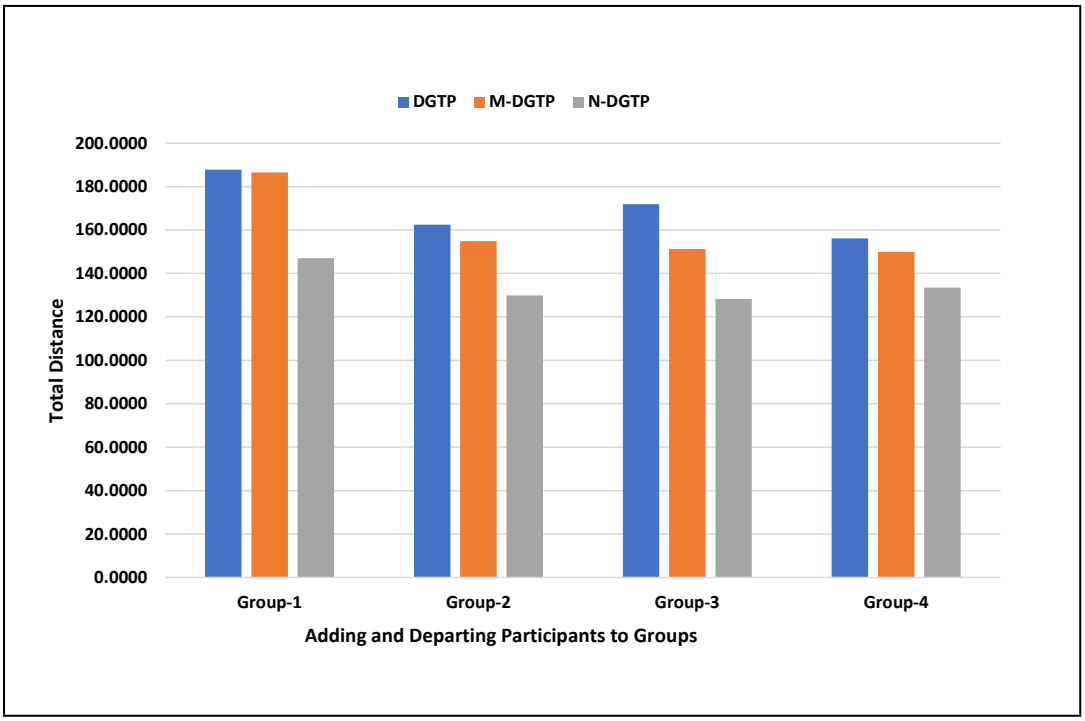

Figure 4.11: Effect of adding and departing participants on total distance

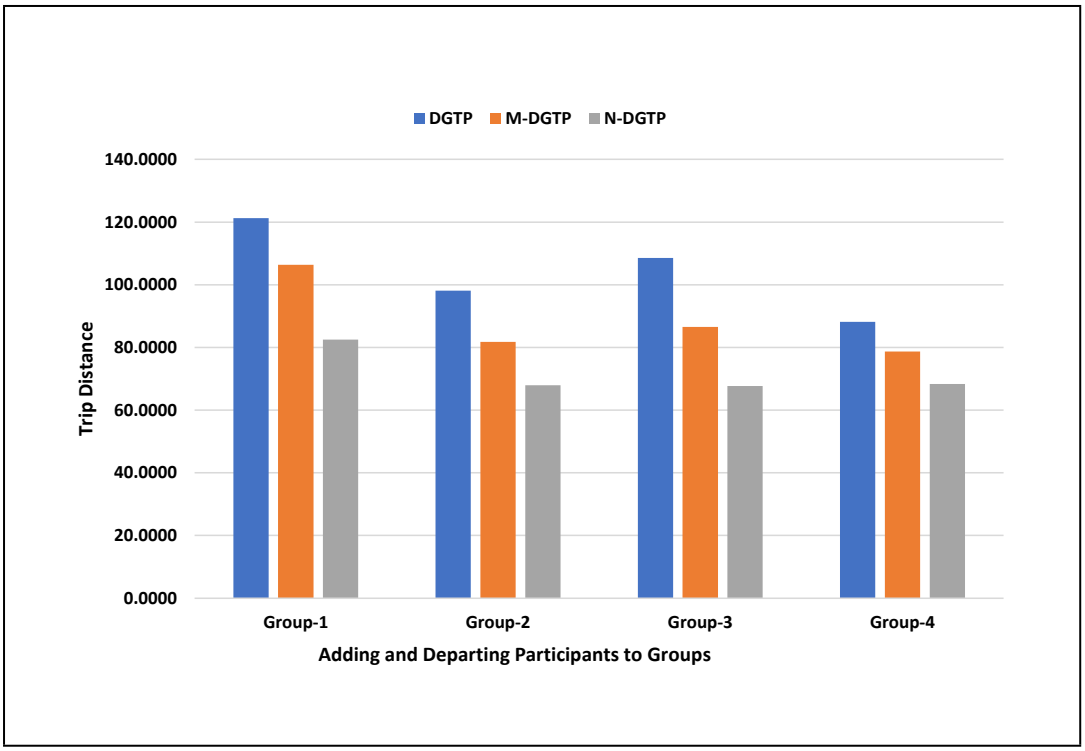

Figure 4.12: Effect of adding and departing participants on trip distance

From the results, it is also observed that DGTP generates on average 26% and M-DGTP generates 19% longer trip results than N-DGTP in terms of total distance Which means M-DGTP generates on average 7% better trip results than DGTP.

Figure 4.12 shows the effect of random addition and departure of participants on the all three approaches in terms of trip distances. For all group combinations, M-DGTP shows better trip results than DGTP. This is because M-DGTP choose shortest route prioritizing the location of the group. So, with the increase of group add-depart transaction performance of M-DGTP generating better trip result also increases. N-DGTP generates the smallest trip in terms of trip distance. From the results, it is also observed that DGTP generates an average of 45% and M-DGTP generates 23% longer trip results than N-DGTP in terms of trip distances. So, M-DGTP generates an average of 22% better trip result than DGTP in terms of trip distances.

4.6 Discussion

In this section, we discuss our conducted experiments and evaluations and discuss some generalities. We conduct experiments for different experimental setup and for each setup we conduct our experiments 5x times. In our experiments, rather than considering all the COIs, we only consider the COIs having less than 1000 POIs. This is because, the N-DGTP approach produces a very long running time which becomes worse as the number of POIs increases. To overcome this for experimental purposes, we reduce the searching criteria by considering COIs having less than 1000 POIs.

N-DGTP generates smaller trip results (i.e. trip and total distances) than both DGTP and M-DGTP because it performs an optimal guided search. N-DGTP computes an initial trip first and the initial trip is re-calculated if changes (i.e. during-trip additions and departures) are made. But this strategy, proposed by Tabassum et al. [33], is also a drawback for N-DGTP because all the COIs must be pre-specified and cannot be changed during the trip. Although both DGTP and M-DGTP do not require pre-specified COIs. In addition, both

generate only a slightly longer route than those generated by N-DGTP. In all cases, DGTP and M-DGTP significantly outperforms N-DGTP in terms of processing time especially for very high number of group transactions. This is because, N-DGTP must perform a very high number of POI search operations and partial trip re-calculations, whereas both DGTP and M-DGTP approaches do not require any partial trip re-calculation.

Both the DGTP and M-DGTP approaches produce exactly the same trip and total distance for the static group scenarios and the dynamic group scenarios where only participants depart. However, M-DGTP optimizes the route in the adding group members and both addition and departure of group members scenarios. From the results, it is observed that M-DGTP chooses significantly optimized trips when both the addition and departure transactions are high in the trip.

When we compare our static group results with the dynamic group results for DGTP, M-DGTP and N-DGTP, in almost every case the differences in the proposed (i.e. DGTP and M-DGTP) vs adapted Tabassum et al. [33] (i.e. N-DGTP) is smaller in the static groups than those found in the dynamic groups. The static group scenarios assume all the group members start and depart at the same time, which means there is no group transaction during the trip. On the other hand, the dynamic group scenario assumes any member(s) can join at any point of the trip which means there is the occurrence of group transactions during the trip. From the results, our proposed M-DGTP approach contributes significantly by optimizing the route for dynamic groups when the group transactions are high than no group transactions of static groups.

4.7 Summary

In this chapter, experiments and evaluations of our two proposed approaches: DGTP and M-DGTP are discussed and compared with adapted N-DGTP approach which has been recently proposed by Tabassum et al. [33]. The next chapter concludes the thesis with some future research directions.

Chapter 5

Conclusion

5.1 Our Contribution

We propose two strategies to process SGTPQs in spatial databases. Both approaches consider dynamic groups where, any member can leave or join at any POI of the trip. In addition, both approaches do not require that the number and sequence of COIs be specified in advance. The first proposed DGTP approach prioritizes all source locations equally to find the desired POI without considering the current group size of the source locations. The first approach optimally selects each POI that is closest with respect to the source locations. Though the first approach selects the POIs that are closest to the respective source locations the total and trip distance calculated from this approach may not be best.

The proposed M-DGTP approach prioritizes the location of the group over the new member joining locations when selecting POIs of intended COIs to generate the resultant trip. The main objective of M-DGTP approach is to reduce the total and trip distances.

From experimental results, it is found that for joining new group members in the middle of the trip M-DGTP achieves on average of 10% and 30% better trip results than DGTP in terms of total distance and trip distance respectively. In the case of random addition and departure of group members during the trip, it is found from the results that M-DGTP generates on average of 7% and 22% better trip results than DGTP in terms of total and trip distnace respectively.

In the case of static group and the case where members only leave, M-DGTP has no improvement over DGTP. This is because in a static group it is assumed that all the group

members start at the same time and no new member joins or departs during the trip, which means the group only moves from one POI to another. In the case of members only leaving in between trips M-DGTP also has no effect over DGTP.

From experimental results, it is also found that when our proposed approaches are compared with N-DGTP in terms of processing time in every case, our proposed approaches significantly outperform N-DGTP. When our proposed approaches are compared with N-DGTP in terms of total and trip distance it is found that N-DGTP generates the smallest trip in all cases.

Though N-DGTP uses an optimal method and generates small trip results it is very costly in terms of processing time, whereas our proposed approaches still being a greedy method generates a small margin of longer trips than N-DGTP. In addition, N-DGTP requires that the intended list of COIs that the members want to travel to be pre-determined before the start of the trip, which means the COIs are static and can not be changed during the trip.

5.2 Future Work

In our proposed strategies we find optimal POI locations for the group and for some cases it is observed that the home distance (i.e. the distance from last visited POI to respective destination) is larger than the individual trip distance of the member. In the future, we plan to work on optimizing this home distance to achieve a better overall trip result.

In [33] an optimized algorithm is also presented that uses elliptical properties to refine the POI search region. In the future, we also plan to apply elliptical properties to our proposed approaches and perform a comparison to the improved approach proposed in [33].

We also plan to work on reducing total and trip distances for both the approaches by calculating an optimal start and departure point for the group. Calculating optimal meeting and departure locations will help further to improve our existing results.

In our work, each POI exactly belongs to one COI. But, a POI may belong to multiple

COIs depending on dataset. We plan to extend our approaches so that it works for the same POI belonging multiple COIs.

In future, we also plan to extend our strategies in terms of time, which means the group can visit all POIs in smallest possible time in such a way that the total and trip distances are minimized.

Bibliography

[1] Tenindra Abeywickrama, Muhammad Aamir Cheema, and David Taniar. K-nearest neighbors on road networks: a journey in experimentation and in-memory implementation. *Proceedings of the VLDB Endowment*, 9(6):492–503, 2016.

[2] Elham Ahmadi and Mario A Nascimento. A mixed breadth-depth first search strategy for sequenced group trip planning queries. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 24–33. IEEE, 2015.

[3] Elham Ahmadi and Mario A Nascimento. Ibs: An efficient stateful algorithm for optimal sequenced group trip planning queries. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pages 212–221. IEEE, 2017.

[4] Rudolf Bayer and Edward McCreight. Organization and maintenance of large ordered indexes. In *Software pioneers*, pages 245–262. Springer, 2002.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *Acm Sigmod Record*, volume 19, pages 322–331. ACM, 1990.

[6] Jae-Woo Chang and Yong-Ki Kim. Materialization-based range and k-nearest neighbor query processing algorithms. In *International Conference on Flexible Query Answering Systems*, pages 65–74. Springer, 2006.

[7] Haiquan Chen, Wei-Shinn Ku, Min-Te Sun, and Roger Zimmermann. The multi-rule partial sequenced route query. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 10:1–10:10. ACM, 2008.

[8] Ying-Ju Chen, Kun-Ta Chuang, and Ming-Syan Chen. Spatial-temporal query homogeneity for knn object search on road networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1019–1028. ACM, 2013.

[9] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[10] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.

[11] Viviana E Ferraggine, Jorge H Doorn, and Laura C Rivero. *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends*. IGI Global, 2009.

[12] Yunjun Gao and Baihua Zheng. Continuous obstructed nearest neighbor queries in spatial databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 577–590. ACM, 2009.

[13] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[14] Tanzima Hashem, Sukarna Barua, Mohammed Eunus Ali, Lars Kulik, and Egemen Tanin. Efficient computation of trips with friends and families. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 931–940. ACM, 2015.

[15] Tanzima Hashem, Tahrima Hashem, Mohammed Eunus Ali, and Lars Kulik. Group trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases*, pages 259–276. Springer, 2013.

[16] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 840–851. VLDB Endowment, 2004.

[17] Mohammad R Kolahdouzan and Cyrus Shahabi. Alternative solutions for continuous k nearest neighbor queries in spatial network databases. *GeoInformatica*, 9(4):321–341, 2005.

[18] Ken CK Lee, Wang-Chien Lee, and Baihua Zheng. Fast object search on road networks. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 1018–1029. ACM, 2009.

[19] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. On trip planning queries in spatial databases. In *International symposium on spatial and temporal databases*, pages 273–290. Springer, 2005.

[20] Kyriakos Mouratidis, Man Lung Yiu, Dimitris Papadias, and Nikos Mamoulis. Continuous nearest neighbor monitoring in road networks. In *Proceedings of the 32nd international conference on Very large data bases*, pages 43–54. VLDB Endowment, 2006.

[21] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. Group nearest neighbor queries. In *Proceedings. 20th International Conference on Data Engineering*, pages 301–312. IEEE, 2004.

[22] Dimitris Papadias, Yufei Tao, Kyriakos Mouratidis, and Chun Kit Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database Systems (TODS)*, 30(2):529–576, 2005.

[23] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 802–813. VLDB Endowment, 2003.

[24] Maytham Safar. Group k-nearest neighbors queries in spatial network databases. *Journal of geographical systems*, 10(4):407–416, 2008.

[25] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. Scalable network distance browsing in spatial databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 43–54. ACM, 2008.

[26] Samiha Samrose, Tanzima Hashem, Sukarna Barua, Mohammed Eunus Ali, Mohammad Hafiz Uddin, and Md Iftekhar Mahmud. Efficient computation of group optimal sequenced routes in road networks. In *2015 16th IEEE international conference on mobile data management*, volume 1, pages 122–127. IEEE, 2015.

[27] Jochen Schiller and Agnès Voisard. *Location-Based Services*. Elsevier, 2004.

[28] Mehdi Sharifzadeh, Mohammad Kolahdouzan, and Cyrus Shahabi. The optimal sequenced route query. *The VLDB Journal—The International Journal on Very Large Data Bases*, 17(4):765–787, 2008.

[29] Mehdi Sharifzadeh and Cyrus Shahabi. Processing optimal sequenced route queries using voronoi diagrams. *GeoInformatica*, 12(4):411–433, 2008.

[30] Shashi Shekhar and Sanjay Chawla. *Spatial Databases: A Tour*. Pearson, 2003.

[31] Nusrat Sultana, Tanzima Hashem, and Lars Kulik. Group nearest neighbor queries in the presence of obstacles. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 481–484. ACM, 2014.

[32] Guang-Zhong Sun, Zhong Zhang, and Jing Yuan. An efficient pre-computation technique for approximation knn search in road networks. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, pages 41–44. ACM, 2009.

[33] Anika Tabassum, Sukarna Barua, Tanzima Hashem, and Tasmin Chowdhury. Dynamic group trip planning queries in spatial databases. In *Proceedings of the 29th international conference on scientific and statistical database management*, pages 38:1–38:6. ACM, 2017.

[34] Man Lung Yiu, Nikos Mamoulis, and Dimitris Papadias. Aggregate nearest neighbor queries in road networks. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):820–833, 2005.

[35] Jun Zhang, Nikos Mamoulis, Dimitris Papadias, and Yufei Tao. All-nearest-neighbors queries in spatial databases. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management*, pages 297–306. IEEE, 2004.

[36] Ruicheng Zhong, Guoliang Li, Kian-Lee Tan, Lizhu Zhou, and Zhiguo Gong. G-tree: An efficient and scalable index for spatial search on road networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2175–2189, 2015.